**A MINI-PROJECT REPORT ON**

# COLOR DETECTION USING PYTHON

Submitted for

**PROJECT SKILLS LABORATORY of**

**B.Tech VI Semester**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

Under the guidance of

**Dr.T.KESAVA RAO, M.Tech, Ph.D.,**

**Associate Professor**

**Computer Science And Engineering**

BY

| | |
|---|---|
| **C CHETHAN** | **21751A0541** |
| **G VASANTH KUMAR** | **21751A0560** |
| **A KARTHIK** | **21751A0504** |
| **E V CHANDRAKANTH GOUD** | **21751A0552** |

**SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES**
**(Autonomous-NBA Accredited)**
**(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu), Chittoor, A.P-517127.**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**(2023 -24)**

**SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES**
**(Autonomous-NBA Accredited)**
**(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)**
**Chittoor,A.P-517127.**

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## CERTIFICATE

This is to certify that the mini-project work entitled **" COLOR DETECTION USING PYTHON"** is a bonafied work done by

| | |
|---|---|
| **C CHETHAN** | **21751A0541** |
| **G VASANTH KUMAR** | **21751A0560** |
| **A KARTHIK** | **21751A0504** |
| **E V CHANDRAKANTH GOUD** | **21751A0552** |

of B.Tech VI semester , Department of Computer Science and Engineering, in the laboratory of Project Skills Lab during the year 2023 to 2024.

Signature of the Supervisor
**Dr.T.KESAVA RAO, M.Tech., Ph.D.,**
Associate Professor,
Department of Computer Science And Engineering,
Sreenivasa Institute of Technology and Management Studies, Chittoor, A.P.

Signature of the Head of Department
**Prof. M.E.PALANIVEL**
Professor & HOD,
Department of Computer Science And Engineering,
Sreenivasa Institute of Technology and Management Studies, Chittoor, A.P.

Submitted for Examination (Viva-Voce) held on……………………

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

**SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES,**
**(Autonomous-NBA Accredited)**
**(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)**
**Chittoor,A.P-517127.**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## InstituteVision

To emerge as a Centre of Excellence for Learning and Research in the domains of engineering, computing and management.

## InstituteMission

M1:Provide congenial academic ambience with state-art of resources for learning and research.

M2: Ignite the students to acquire self-reliance in the latest technologies.

M3: Unleash and encourage the innate potential and creativity of students.

M4: Inculcate confidenceto face and experiencenew challenges.

M5:Foste renterprising spirit among students.

## DepartmentVision

To contribute for the society through excellence in Computer Science and Engineering with a deep passion for wisdom, culture and values.

## DepartmentMission

- Provide congenial academic ambience with necessary infrastructure and learning resources.
- Inculcate confidence to face and experience new challenges from industry and society.
- Ignite the students to acquire self-reliance in the latest technologies.
- Foster Enterprising spirit among students.

# SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES

**(Autonomous-NBA Accredited)**
**(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)**
**Chittoor,A.P-517127.**

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

### PROGRAMME EDUCATIONAL OBJECTIVES(PEO's):

**PEO1:** Excel in Computer Science and Engineering program through quality studies, enabling success in computing industry.**(Professional Competency)**.

**PEO2:** Surpass in one's career by critical thinking towards successful services and growth of the organization,orasan entrepreneur or in higherstudies. **(Successful Career Goals)**.

**PEO3:** Enhance knowledge by updating advanced technological concepts for facing the rapidly changing world and contribute to society through innovation and creativity **(ContinuingEducationandContribution to Society)**.

### PROGRAM SPECIFIC OUTCOMES(PSO's):

**PSO1:**Have Ability to understand, analyses anddevelopcomputerprogramsin the areas like algorithms, system software, web design,big data analytics, and networking.

**PSO2:** Deploy the modern computer languages,environment, and platforms in creating innovative products and solutions.

### PROGRAMME OUTCOMES(PO's):

**PO1-Engineeringknowledge:** Apply the knowledge of mathematics,science , engineering fundamentals, and an engineering specialization to thesolution of complex engineering problems.

**PO2 - Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**PO3 - Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**PO4 - Conduct investigations of complex problems:** Useresearch-basedknowledgeandresearchmethods including design of experiments, analysis and interpretation of data, and synthesisofthe information to provide valid conclusions.

**PO5 - Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**PO6 - The engineer and society:** Apply reasoning informed by the contextualknowledgetoassesssocietal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**PO7 - Environmentand sustainability:**Understand the impactof the professionalengineering solutionsin societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**PO8 -Ethics:**Apply ethicalprinciples andcommit toprofessional ethics and responsibilities andnorms of the engineering practice.

**PO9 - Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**PO10-Communication:**Communicateeffectivelyoncomplexengineeringactivitieswiththe engineering community and with society at large, such as, being able to comprehend and write effective reportsanddesign documentation,makeeffectivepresentations,and give andreceive clear instructions.

**PO11 - Project management and finance:** Demonstrate knowledge and understanding of the engineering andmanagement principles andapply these to one’s own work, as amember and leader in a team, tomanage projects and in multidisciplinary environments.

**PO12 - Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES**
**(Autonomous-NBA Accredited)**
**(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)**
**Chittoor,A.P-517127.**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## Course Out Comes for Project Work

Oncompletionofmini-projectworkthestudentwillbeableto

**CO1.** Demonstrate in-depth knowledge on the project topic.

**CO2.** Identify, analyze and formulate complex problem chosen for project work to attain substantiated conclusions.

**CO3.** Design solutions to the chosen project problem.

**CO4.** Undertake investigation of project problem to provide valid conclusions.

**CO5.** Use the appropriate techniques, resources and modern engineering tools necessary for project work.

**CO6.** Apply project results for sustainable development of the society.

**CO7.** Understand the impact of project results in the context of environmental sustainability.

**CO8.** Understand professional and ethical responsibilities while executing the project work.

**CO9.** Function effectively as individual and a member in the project team.

**CO10.** Develop communication skills, both oral and written for preparing and presenting project report.

**CO11.** Demonstrate knowledge and understanding of cost and time analysis required for carrying out the project.

**CO12.** Engage in lifelong learning to improve knowledge and competence in the chosen area of the project.

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES,

**(Autonomous-NBA Accredited)**

**(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)**

**Chittoor,A.P-517127.**

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## CO – PO MAPPING

| CO\PO | PO 1 | PO 2 | PO 3 | PO 4 | PO 5 | PO 6 | PO 7 | PO 8 | PO 9 | PO10 | PO11 | PO12 | PSO 1 | PSO 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **CO.1** | 3 | - | - | - | - | - | - | - | - | - | - | - | 3 | 3 |
| **CO.2** | - | 3 | - | - | - | - | - | - | - | - | - | - | 3 | 3 |
| **CO.3** | - | - | 3 | - | - | - | - | - | - | - | - | - | 3 | 3 |
| **CO.4** | - | - | - | 3 | - | - | - | - | - | - | - | - | 3 | 3 |
| **CO.5** | - | - | - | - | 3 | - | - | - | - | - | - | - | 3 | 3 |
| **CO.6** | - | - | - | - | - | 3 | - | - | - | - | - | - | 3 | 3 |
| **CO.7** | - | - | - | - | - | - | 3 | - | - | - | - | - | 3 | 3 |
| **CO.8** | - | - | - | - | - | - | - | 3 | - | - | - | - | 3 | 3 |
| **CO.9** | - | - | - | - | - | - | - | - | 3 | - | - | - | 3 | 3 |
| **CO.10** | - | - | - | - | - | - | - | - | - | 3 | - | - | 3 | 3 |
| **CO.11** | - | - | - | - | - | - | - | - | - | - | 3 | - | 3 | 3 |
| **CO.12** | - | - | - | - | - | - | - | - | - | - | - | 3 | 3 | 3 |
| **CO** | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

**SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES,**
**(Autonomous-NBA Accredited)**
**(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)**
**Chittoor,A.P-517127.**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# Evaluation Rubrics for Project Work

| *Rubric(CO)* | Excellent(wt=3) | Good(wt=2) | Fair(wt=1) |
|---|---|---|---|
| *Selection of Topic(CO1)* | Selected a latest topic through complete knowledge of facts and Concepts | Selected a topic through partial knowledge off acts and concepts | Selected a topic through improper knowledge of facts and concepts |
| *Analysis and Synthesis(CO2)* | Thorough Comprehension through analysis/ synthesis | Reasonable Comprehension through analysis/ synthesis | Improper Comprehension through analysis/ synthesis |
| *ProblemSolving(CO3)* | Thorough comprehension about what is proposed in the literature papers | Reasonable comprehension about What is proposed in the literature papers | Improper comprehension about What is proposed in the literature |
| *LiteratureSurvey(CO4)* | Extensive literature Survey with standard References | Considerableliterature Survey with standard References | Incompleteliterature Survey with sub standard References |
| *UsageofTechniques&Tools(CO5)* | Clearly identified and has complete knowledge Of techniques&tools used in the project work | Identified and has sufficient knowledge of techniques&tools used in the project work | Identified and has inadequate knowledge of techniques & tools used in project work |
| *Projectworkimpacton Society (CO6)* | Conclusion of project work has strong impact on society | Conclusion of project work has considerable impact on society | Conclusion of project work has feeble impact on society |
| *Projectworkimpacton Environment(CO7)* | Conclusion of project work has strong impact on Environment | Conclusion of project work has considerable impact on environment | Conclusion of project work has feeble impact on environment |
| *Ethicalattitude(CO8)* | Clearly understands ethical and social practices. | Moderate understanding of ethical and social practices. | Insufficient understanding of ethical and social practices. |
| *Independen tLearning(CO9)* | Did literature survey and selected topic with little Guidance | Did literature survey and selected topic with considerable guidance | Selected a topic as suggested by the Supervisor |
| *OralPresentation(CO10)* | Presentation in logical sequence with key points,clear conclusion and excellent language | Presentation with key points,conclusion and good language | Presentationwith insufficient key points and improper Conclusion |
| *ReportWriting(CO10)* | Status report with clear and logical sequence of Chapters using excellent language | Status report with logical sequence of chapters using understandable language | Status report not properly organized |
| *Timeand Cost Analysis(CO11)* | Comprehensive time and cost analysis | Moderate time and cost analysis | Reasonable time and cost analysis |
| *Continuous learning(CO12)* | Highlyenthusiastic Towards continuos learning | Interested in continuous learning | Inadequate interest in continuous learning |

# ACKNOWLEDGEMENT

A Project of this magnitude would have not been possible without the guidance and co-ordination of many people. I am fortune in having top quality people to help, support and guide us in every step towards our goal.

Our team is very much grateful to the Chairman **Sri K. RANGANADHAM** Garu for his encouragement and stalwart support. We are also extremely indebted to the Secretary **Sri D.K. BADRI NARAYANA** Garu for his constant support.

We express our sincere thanks to our Academic Advisor **Dr. K.L. NARAYANA**, **M.Tech., Ph.D,** further, we would like to express our profound gratitude to our principal **Dr.N.VENKATACHALAPATHI, M.T.ech, Ph.D** for providing all possible facilities throughout the completion of our project work.

We express our sincere thanks to our Dean (Academics), **Dr.M.SARAVANAN, M.E., Ph.D.,** further we express our sincere thanks to our Head of the Department **Prof.M.E.PALANIVEL** for his co-operation and valuable suggestions towards the completion of project work.

We express our sincere thanks to our guide **Dr.T.KESAVA RAO, M.Tech ,Ph.D.,** for offering us the opportunity to do this work under his guidance.

We express our sincere salutation to all other teaching and non-teaching staff of our department for their direct and indirect support given during our project work. Last but not the least, we dedicate this work to our parents and the Almighty who have been with us throughout and helped us to overcome the hard times.

# DECLARATION

**We certify that**

- The work contained in this report is original and has been done by us under the Guidance of my supervisor.
- The work has not been submitted to any other Institute for any degree or diploma.
- We have followed the guidelines provided by the Institute in preparing the report.
- We have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
- Whenever we have usedmaterials (data, theoretical analysis,figures,and text)fromother sources, we have given due credit to them by citing them in the text of the reportand giving their details in the references. Further, we have taken permission from the copyright owners of the sources, whenever necessary.

| | |
|---|---|
| **C CHETHAN** | **21751A0541** |
| **G VASANTH KUMAR** | **21751A0560** |
| **A KARTHIK** | **21751A0504** |
| **E V CHANDRAKANTH GOUD** | **21751A0552** |

**TABLE OF CONTENTS**

## ABSTRACT

This Python application enables real-time color detection in both static images and live video feeds through a user-friendly interface. Utilizing Tkinter for the graphical user interface (GUI), OpenCV for image and video processing, and Pandas for managing color data, the application allows users to select an image file or use the webcam feed to detect colors. The `select_image` function facilitates image selection, where clicking on the image reveals the color's name and RGB values, which are mapped using a pre-loaded CSV file containing color information. Similarly, the `start_video_feed` function captures live video, enabling dynamic color detection with each click. Mouse callbacks are used to capture RGB values from the image or video feed, and the application identifies the closest matching color name and its hexadecimal code from the CSV file. The identified color information is displayed on the image or video in real-time. The interface updates continuously until the user exits by closing the window or pressing a designated key, providing an efficient tool for various color identification tasks. This makes the application highly useful for graphic designers, digital artists, and anyone needing precise color recognition and naming in their work. The application features a login page to ensure that only authorized users can access the color detection functionality.

## LIST OF FIGURES

## LIST OF TABLES

## LIST OF ABBREVIATIONS

| | |
|---|---|
| GUI | Graphical User Interface |
| RGB | Red Green Blue |
| CSV | Comma Seperated Value |
| CNN | Convolutional Neural Networks |

# CHAPTER-1

# INTRODUCTION

## 1.1 OBJECTIVE:

The objective of this code is to develop an interactive application that enables users to detect and identify colors within static images and live video feeds. The application aims to provide a simple and intuitive graphical user interface (GUI) using Tkinter, allowing users to easily interact with the program. One of the primary goals is to facilitate the upload and display of images from local storage, enabling users to select any image for color detection. Additionally, the application seeks to enable real-time color detection through a live video feed from the user's webcam. It aims to capture the RGB values of pixels clicked by the user and accurately map these values to the closest matching color name and hexadecimal code using a predefined color dataset. Another objective is to dynamically display the detected color name and RGB values, ensuring continuous updates both in static images and live video feeds. The application also aims to ensure seamless interaction by handling mouse events efficiently, allowing for the dynamic capture and display of color information in real-time. Furthermore, it seeks to provide robust performance, ensuring a stable and responsive user experience that correctly handles user inputs and exits gracefully. Overall, the application aspires to be an effective tool for various color identification tasks, catering to users who need to determine specific color details in images and live video streams with ease. The application features a login page to ensure that only authorized users can access the color detection functionality.

## 1.2 PROBLEM IDENTIFICATION:

Accurate color detection in digital images remains challenging due to significant variations in lighting, camera characteristics, and environmental conditions. These variations affect the reliability of color-based applications in fields such as quality control in manufacturing, medical imaging, and automated surveillance systems. Existing methods often fail to adapt to different lighting and camera settings, leading to inconsistencies and errors in color interpretation. There is a crucial need for developing a more robust and adaptable color detection system that can effectively normalize these factors, ensuring accuracy and reliability across various applications. Such a system could utilize advanced algorithms capable of dynamically adjusting color thresholds based on environmental conditions and camera specifications. Additionally, incorporating machine learning techniques could enable the system to learn and adapt to new scenarios over time, further enhancing its performance and versatility. Standardizing color calibration procedures across different devices and environments could also contribute to improving consistency and accuracy in color detection tasks. Collaboration between researchers, engineers, and industry stakeholders is essential to develop comprehensive solutions that address the multifaceted challenges associated with color detection in digital images. By leveraging advancements in sensor technology and computational imaging, we can pave the way for more reliable and efficient color detection systems with wide-ranging applications in diverse fields. Ultimately, overcoming the obstacles posed by variations in lighting, camera characteristics, and environmental conditions will unlock new possibilities for innovation and progress in color-based technologies.

## 1.3 LITERARTUE SURVEY:

Colour info theatres a significant part in the colour picture division & real-times colour sensors, whichs touches this results off videos picture division &Go on until the end. An original real-time colour picture division technique is future, which is founded on colour comparation in RGB colour interplanetary. Rendering to the colour & luminosity info in RGB colour interplanetary, the main colour is resolute at first, & then colour resemblance can be planned with the future calculation technique of colour module, which makes a colour-class chart. Next, the info of the You don't want a sad song used to sort the pixels. Due to the typical that current t1rs feature colour st&ards that change in real time as the temperature deviations, the division results of thermal t1r can be used as a real-time colour sensor. It's already g1of colour improvement & bright source recompense for the sake of possible imprecision of its events. We deliberate the future division ways submission combination with colour sensors in real-times colours picture division for byes theorems submission in fires detections & Even when you think detecting fires in a videos baseds on theses features. Thes trialss showeds thats this futures methods ins vision-baseds fires findings & credentialss ins videoss was actuals; the resulted weres precise & can be cast-off in real-time study

# CHAPTER-2

# SYSTEM ANALYSIS

## 2.1: EXISTING SYSTEM:

The existing system is a Python script that performs color detection on static images. It does not utilize a webcam or provide a graphical user interface (GUI). Instead, users select an image file from their system, and the script processes the image to identify the dominant color. The script employs libraries such as OpenCV, NumPy, and Pandas to read the image file and perform color analysis. Upon selecting an image, the script calculates the RGB values of the pixels and compares them with a predefined dataset of colors stored in a CSV file. The dataset contains a limited number of colors, making the color detection process less complex. Once a match is found, the script returns the name of the closest matching color along with its RGB values and hexadecimal code. Overall, the existing system focuses on detecting the dominant color in static images using a simple dataset of colors without the need for a webcam or a graphical user interface.

### 2.1.1: Demerits:

- The system's accuracy heavily relies on the dataset of colors available for comparison. With a limited dataset, the system may struggle to accurately identify less common or nuanced colors.
- -Since the system operates solely on static images, it cannot provide real-time feedback or dynamic color tracking. Users must manually select and analyze each image separately.
- The system lacks a user interface, limiting user interaction and customization options. Users cannot adjust parameters or provide feedback interactively, hindering usability and flexibility.
- With its focus solely on color detection in static images, the system may have limited applications compared to more versatile solutions capable of handling real-time video feeds or offering additional image processing functionalities.

## 2.2: PROPOSED SYSTEM:

The color detection system is designed to provide users with two main functionalities: image color detection and real-time color detection through a camera feed. Upon selecting an image file, users can click anywhere on the image to retrieve color information, including the color name, RGB values, and hexadecimal code. Similarly, accessing the computer's camera feed allows users to detect colors in real-time by clicking on any point in the feed.

A secure login page has been integrated into the system to ensure that only authorized users can access the color detection functionality. The login page, also built using Tkinter, prompts users to enter their credentials before proceeding. Once the correct username and password are provided, users gain access to the main interface, where they can choose to select an image or start the live video feed for color detection. The login credentials are verified against a pre-defined set of authorized users stored securely, ensuring the integrity and security of the application.

### 2.2.1: Merits:

- Image and real-time color detection capabilities
- User-friendly GUI with intuitive buttons and descriptive labels
- Clear feedback and error handling for smooth user interaction
- Proper resource release and window closure for clean exits
- Custom styles for enhanced visual appeal
- Resizable images for consistent appearance
- Efficient color identification using pre-defined datasets
- Secure login page ensuring authorized access

## 2.3 FEASIBILITY STUDY:

### 2.3.1 Technical Feasibility:

Technical feasibility assesses the compatibility and integration aspects of the color detection system. Firstly, it involves evaluating the compatibility of essential software components such as OpenCV, NumPy, Pandas, and Tkinter with the intended platform. This includes ensuring that the versions of these libraries are supported and potentially updating them if necessary for seamless integration. Secondly, the feasibility study delves into hardware requirements, particularly examining if the system's demands on CPU, memory, and camera resources align with the capabilities of the target devices. It's crucial to ensure compatibility with various camera models and resolutions to accommodate both image upload and real-time detection functionalities. Lastly, integration challenges are identified to anticipate any technical hurdles that may arise during the merging of OpenCV with Tkinter for GUI development, as well as any limitations or dependencies that could impact system functionality.

### 2.3.2 Operational Feasibility

Operational feasibility evaluates the usability and workflow impact of the color detection system. User experience is a significant consideration, encompassing the ease of use for target users who may have varying levels of familiarity with GUI applications and image processing tools. Additionally, the study assesses whether additional training or documentation is necessary to enable users to effectively utilize the system. Workflow impact analysis involves examining how the system integrates into existing processes and workflows. Potential disruptions or adjustments needed to seamlessly incorporate the color detection system into these workflows are identified to ensure minimal operational friction**.**

### 2.3.3 Economic Feasibility

Economic feasibility scrutinizes the financial aspects of the color detection system. Development costs are estimated, including expenses related to software development such as developer salaries, software licenses, and development tools. Any additional licensing fees for third-party services or APIs are also considered. Maintenance costs are evaluated to account for ongoing expenses such as updates, bug fixes, technical support, and server/hosting fees.

# CHAPTER-3

# SYSTEM DEVELOPMENT MODEL

For the development of the color detection application, the **Iterative and Incremental Model** was chosen. This model is particularly suited to projects involving complex functionalities and evolving requirements, as it allows for flexibility, continuous feedback, and risk management.

The Iterative and Incremental Model was selected for several key reasons. First, it manages complexity effectively by breaking down the project into manageable increments. This approach was essential given the project's multifaceted nature, involving graphical user interface (GUI) design, image processing, color detection, and real-time video feed integration. Second, the model provides flexibility and adaptability, accommodating changes and new requirements that arise during the development process. This flexibility is crucial for refining features based on continuous feedback and testing.

## Iteration 1: Basic GUI Setup

The first iteration focused on establishing the main window and layout. The main Tkinter window was set up, and the layout was configured with buttons and labels. Proper shutdown handling was also implemented. The outcome of this iteration was a functional GUI window with basic components.

## Iteration 2: Image File Selection and Display

The second iteration enabled image file selection and display. A file dialog was integrated to allow users to select an image, which would then be displayed within the GUI. The outcome was that users could now select and view an image.

## Iteration 3: Color Detection on Image

The third iteration implemented color detection on static images. This involved capturing mouse click events on the displayed image and retrieving and displaying color information, such as RGB values and color names. The outcome was that users could click on the image to get detailed color information.

**Iteration 4: Real-Time Video Feed**

The fourth iteration integrated a live camera feed and real-time color detection. The webcam feed was accessed using OpenCV, and real-time color detection was implemented. The outcome was that users could now detect colors in real-time through the webcam.

**Iteration 5:Login Page**

The fifth iteration added a secure login page to ensure that only authorized users could access the color detection functionalities. The login page was designed with Tkinter to prompt users for their credentials before granting access to the main interface.

**Iteration 6: Refinement and Optimization**

The fifth iteration focused on refining the user interface and optimizing performance. The GUI layout was enhanced, and image processing algorithms were optimized. The outcome was a more user-friendly and efficient application.

**Iteration 6: Testing and Debugging**

The final iteration involved comprehensive testing and debugging. Each feature was thoroughly tested to identify and fix any bugs, ensuring the application's stability and reliability. The outcome was a robust and reliable application ready for deployment.

# CHAPTER-4

# SYSTEM DESCRIPTION

## 4.1: PROBLEM IDENTIFICATION:

The color detection application aims to accurately identify colors in both static images and real-time video feeds. Several potential issues need to be addressed during its development. The project involves various components, such as GUI design, image processing, and real-time video handling, adding complexity to ensuring seamless integration and functionality.

## 4.2: OVER VIEW OF THE SYSTEM:

The color detection application is designed to provide users with a straightforward and efficient tool for identifying colors in both static images and real-time video feeds. The system leverages a combination of Python libraries including OpenCV for image processing and video handling, Tkinter for the graphical user interface (GUI), and Pandas for managing color data.

**System Components:**

**1. Graphical User Interface (GUI):** The application features a user-friendly GUI built using Tkinter. The main window provides buttons for selecting an image file or starting the live video feed. The interface is designed to be intuitive, allowing users to navigate the application easily.

**2. Image File Selection and Display:** Users can select image files from their local storage using a file dialog. Supported formats include JPG, PNG, BMP, and more. The selected image is displayed within the application window, where users can click on any part of the image to get color information.

**3. Color Detection on Static Images:** The application captures mouse click events on the displayed image. When a user clicks on a point in the image, the RGB values of the pixel at that location are retrieved. These values are then used to determine the closest color name and its hexadecimal representation from a predefined color dataset.

**4. Real-Time Video Feed Integration:** The application can access the computer’s webcam to provide a live video feed. Users can click on any part of the live feed to get real-time color information. This feature is particularly useful for tasks that require on-the-fly color detection.

**5. Color Database:** A CSV file containing a list of colors with their corresponding names, hexadecimal codes, and RGB values is used to map detected colors to their names. This database is read using Pandas, allowing for efficient lookup and processing.

**Key Functionalities:**

**Mouse Event Handling:** The application detects mouse clicks on both static images and live video feeds. The coordinates of the click are used to fetch the pixel's color information.

**Color Information Display:** Upon detecting a color, the application displays the RGB values, color name, and hexadecimal code. This information is shown in a rectangle overlay within the application window for easy visibility.

**Real-Time Processing:** The live video feed is processed in real-time, allowing users to click on the video to get instant color information. This requires efficient use of system resources to ensure smooth performance.

**Login Authentication:** Users must log in with valid credentials to access the application's features. The login system ensures data security and user management.

**Binary Search Implementation:**The binary search algorithm is used to efficiently find the closest color name from the sorted color database.

**Error Handling**: The application includes error handling mechanisms to manage issues such as unsupported file formats and camera access problems, ensuring robustness and reliability.

## 4.3: SYSTEM ARCHITECTURE DIAGRAM:

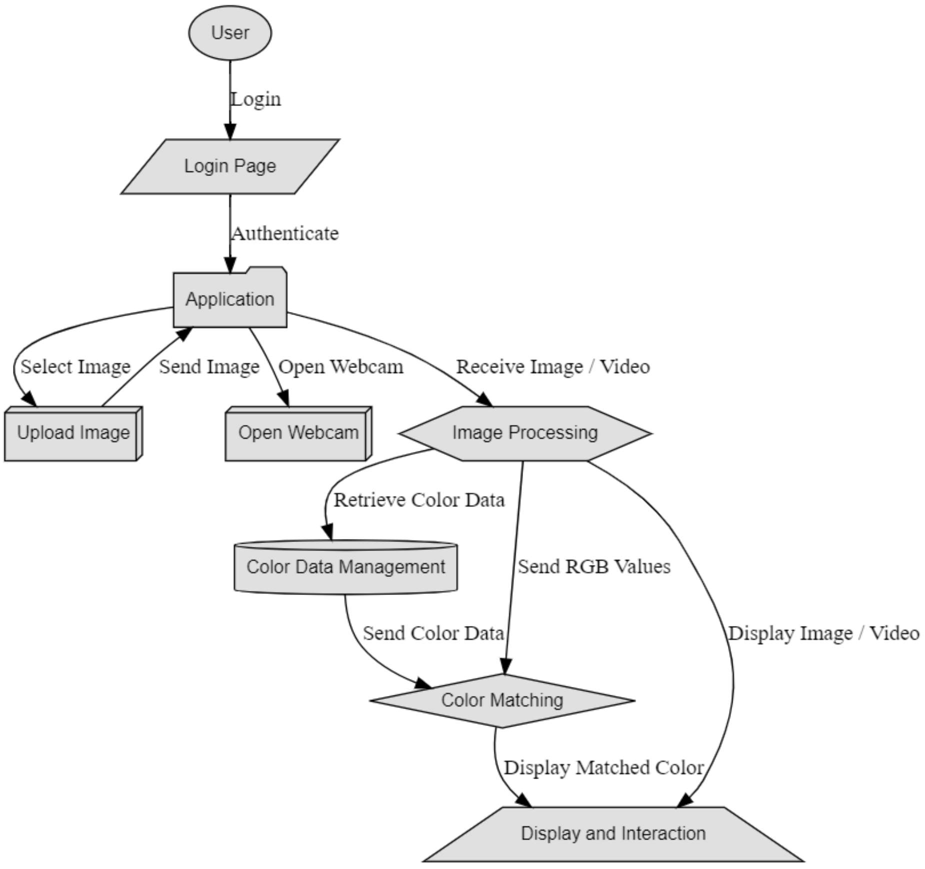


**Fig:1 System Architecture**

# CHAPTER-5

# SYSTEM DESIGN

## 5.1: Module And Its Description:

**1.login:**

**Description**:This module handles user authentication for accessing the color detection application. It provides functions for user login, creating new user accounts, and verifying user credentials against a SQLite database.

**Functions:**

**login():** Handles user login by verifying the entered username and password against the database.

**create_user():** Allows the creation of a new user account by adding the username and password to the database.

**2. image_processing:**

**Description:**This module contains functions related to image processing, such as resizing images, drawing circles, and identifying colors based on RGB values.

Functions:

**draw_circle()**: Draws a filled circle on the image frame.

**identify_color():** Identifies the color at the specified coordinates in the image.

**getColorName():** Identifies the name and hexadecimal code of the closest color based on RGB values.

**3. User Interface:**

**Description:**This module provides utility functions for creating and customizing the GUI components of the color detection application.

**Functions:**

**select_image():** Opens a file dialog for the user to select an image file.

**open_camera**(): Initiates the process to start the webcam for real-time color detection.

**4. database_management:**

**Description:**This module manages the SQLite database used for storing user credentials. It provides functions for creating the database table, inserting new user records, and querying existing user data.

**Functions:**

**create_users_table():** Creates the 'users' table in the database if it does not already exist.

**insert_user(username, password):** Inserts a new user record into the 'users' table.

**verify_user(username, password):** Verifies the entered username and password against the records in the 'users' table.

**5.video_handling:**

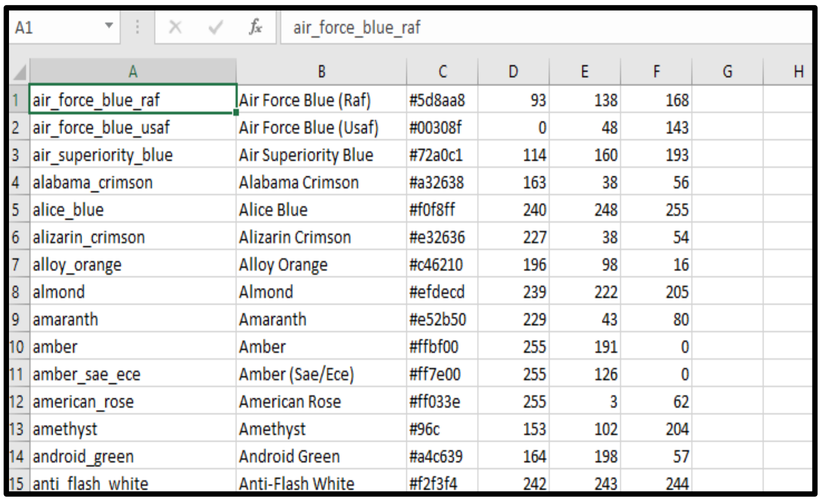
**Description:**This module handles the opening and processing of real-time video feeds from a webcam. It provides functions to start and stop the video feed, as well as perform color detection on the live video stream.

**Functions:**

**start_video_feed():** Starts the webcam and initiates real-time color detection from the live video feed.

**identify_color():** Identifies the color at the specified coordinates in the video frame.

**6.Colors Data set:**



A1 | air_force_blue_raf

|  | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | air_force_blue_raf | Air Force Blue (Raf) | #5d8aa8 | 93 | 138 | 168 |  |  |
| 2 | air_force_blue_usaf | Air Force Blue (Usaf) | #00308f | 0 | 48 | 143 |  |  |
| 3 | air_superiority_blue | Air Superiority Blue | #72a0c1 | 114 | 160 | 193 |  |  |
| 4 | alabama_crimson | Alabama Crimson | #a32638 | 163 | 38 | 56 |  |  |
| 5 | alice_blue | Alice Blue | #f0f8ff | 240 | 248 | 255 |  |  |
| 6 | alizarin_crimson | Alizarin Crimson | #e32636 | 227 | 38 | 54 |  |  |
| 7 | alloy_orange | Alloy Orange | #c46210 | 196 | 98 | 16 |  |  |
| 8 | almond | Almond | #efdecd | 239 | 222 | 205 |  |  |
| 9 | amaranth | Amaranth | #e52b50 | 229 | 43 | 80 |  |  |
| 10 | amber | Amber | #ffbf00 | 255 | 191 | 0 |  |  |
| 11 | amber_sae_ece | Amber (Sae/Ece) | #ff7e00 | 255 | 126 | 0 |  |  |
| 12 | american_rose | American Rose | #ff033e | 255 | 3 | 62 |  |  |
| 13 | amethyst | Amethyst | #96c | 153 | 102 | 204 |  |  |
| 14 | android_green | Android Green | #a4c639 | 164 | 198 | 57 |  |  |
| 15 | anti_flash_white | Anti-Flash White | #f2f3f4 | 242 | 243 | 244 |  |  |

**Table 1: colors data set**

## 5.2: ALGORITHM OR MODEL,EXPLANATION

**Binary Search Algorithm:**

**Explanation:**

The binary search algorithm is used to efficiently find the closest color match in a predefined color dataset based on the RGB values of a target color.

The function colorname() takes three parameters representing the Blue (B), Green (G), and Red (R) values of the target color, respectively.

**Binary Search Implementation:**

**Initialization:**The algorithm initializes two pointers, `low` and `high`, representing the indices of the first and last elements in the dataset, respectively.

**Iteration:** In each iteration, the algorithm calculates the midpoint (`mid`) between the `low` and `high` indices and compares the RGB values of the target color with the RGB values of the color at the `mid` index.

- If the target color's RGB values are less than the RGB values of the color at `mid`, the algorithm updates the `high` pointer to `mid - 1` to search the left half of the dataset.
- If the target color's RGB values are greater than the RGB values of the color at `mid`, the algorithm updates the `low` pointer to `mid + 1` to search the right half of the dataset.
- If the RGB values match exactly, the algorithm returns the color name and hexadecimal code of the color at index `mid`.

The algorithm returns the name and hexadecimal code of the closest matching color found in the dataset based on the RGB values of the target color.

## 5.3: DFD (Data Flow Diagram):

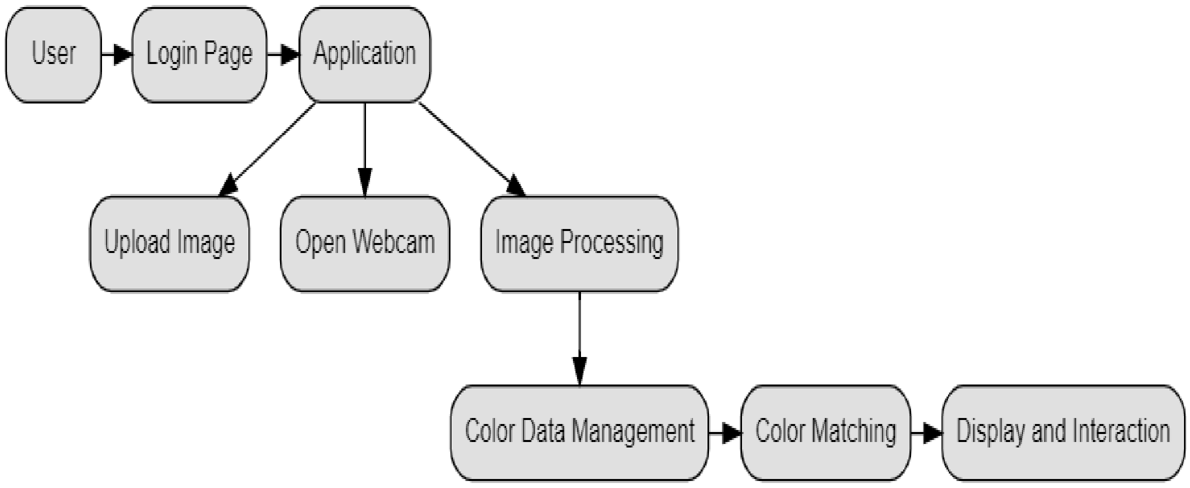


**Fig:2 Data flow diagram**

## 5.4: UML DIAGRAMS:

### Use case diagram:

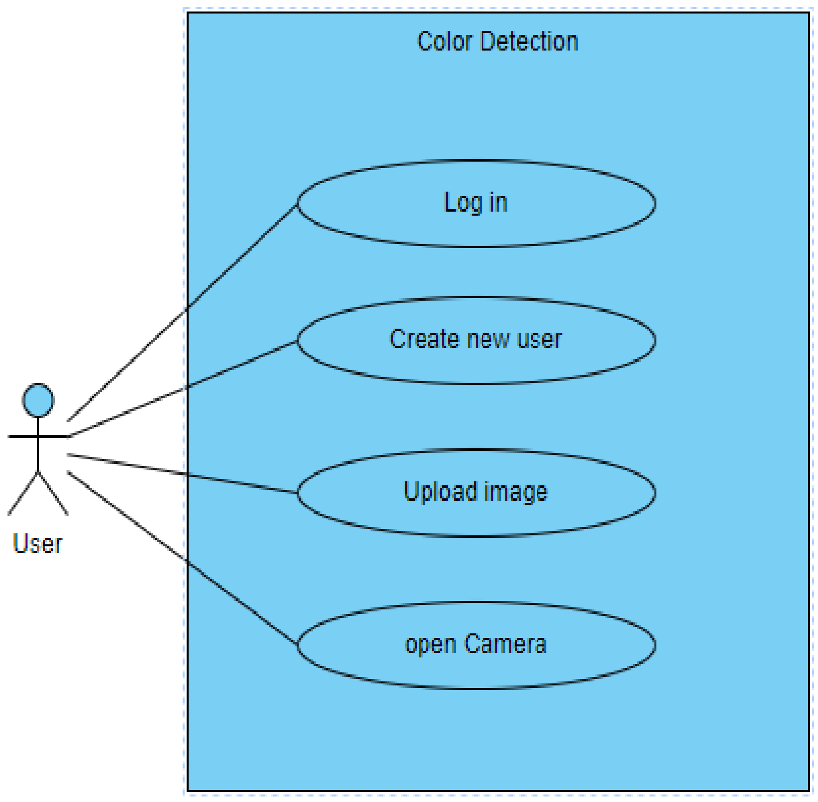


**Fig:3 use case diagram**

## 5.5: INPUT DESIGN:

**Login Page:**

Before accessing the main functionality, users are required to log in using their credentials. The login page consists of input fields for entering a username and password. Upon clicking the "Login" button, the `login()` function is triggered, which verifies the entered credentials against the database. If the credentials are valid, the user is granted access to the application. Otherwise, an error message is displayed indicating invalid credentials.

**User Interface:**

The GUI is designed using tkinter, with buttons labeled "UPLOAD IMAGE" and "OPEN CAMERA" for selecting an image or starting the video feed. Each button is accompanied by an icon representing its respective action. Labels are provided below each button to describe their functionality.

**Selecting Image:**

Users can click a button labeled "UPLOAD IMAGE" to trigger the `select_image()` function.This function opens a file dialog window using `filedialog.askopenfilename()` from tkinter, allowing users to browse and select an image file. Once a file is selected, the image is loaded using OpenCV (`cv2.imread()`), and its color information is displayed. Users can then click on any point in the image to obtain the RGB values and the corresponding color name.

**Starting Video Feed:**

Users can click a button labeled "OPEN CAMERA" to trigger the `start_video_feed()` function.This function accesses the camera (if available) using `cv2.VideoCapture(0)` and continuously captures video frames.The video feed is displayed in a window, and users can click on any point in the video feed to obtain the RGB values and the corresponding color name.

## 5.6: OUTPUT DESIGN:

**Color Information Display:**

When the user selects a point on the image or video feed, the RGB values of the selected pixel along with the corresponding color name are displayed.The color name is obtained by comparing the RGB values of the selected pixel with predefined colors stored in a CSV file.The color name is displayed in a text format along with the RGB values.The color name is also displayed in a rectangle filled with the color corresponding to the selected pixel.

**Dynamic Updating:**

The color information updates dynamically as the user clicks on different points in the image or video feed.As the user moves the cursor and selects different pixels, the displayed color name and RGB values change accordingly.This dynamic updating provides real-time feedback to the user about the color of the selected pixel.

# CHAPTER-6

# SYSTEM SPECIFICATION

## 6.1: SYSTEM REQIREMENTS:

### 6.1.1: Software Requirements:

- Windows 7 or above
- Python : version 3.x installed
- **Python libraries:**
  - NumPy
  - Pandas
  - OpenCV
  - Imutils
  - Tkinter
  - Pillow
  - Sqllite3

### 6.1.2: Hardware Reqirements :

- i3 Processor Based Computer or higher
- Memory: 4 GB RAM
- Hard Drive: 50 GB
- Monitor
- camera

# CHAPTER-7

# SYSTEM IMPLEMENTATION

**Setting Up the Environment:**

Install necessary dependencies such as Python, OpenCV, NumPy, and tkinter.Ensure that the required image files and CSV file containing color data are available.

**Database Setup:**

Create an SQLite database to store user login details.Define tables for storing user information, including username and password.

**Login Page Implementation:**

Develop a login page using tkinter for user authentication.Create functions to handle user login attempts, validate credentials, and provide appropriate feedback to the user.Integrate the login functionality with the main application to ensure seamless navigation upon successful login.

**Image Processing Module:**

Implement functions for image processing using OpenCV to extract RGB values and determine color names.Develop algorithms for color matching based on the provided CSV file containing color data.Create user interfaces for selecting an image file or starting the webcam feed.

**GUI Design:**

Design a user-friendly graphical interface using tkinter.Include buttons for uploading images and opening the webcam, along with labels to describe their functionality.Ensure consistent styling and layout to enhance usability.

**Integration:**

Integrate the login page, image processing module, and GUI components to form a cohesive application.Implement event handling mechanisms to trigger appropriate actions based on user interactions, such as button clicks or mouse events.

Testing:Conduct thorough testing of the system to validate its functionality, including login authentication, image processing, and GUI responsiveness.Perform unit tests, integration tests, usability tests, and error handling tests to identify and address any issues or bugs.

**Documentation:**

Create documentation outlining the system architecture, implementation details, and usage instructions.Include information on system requirements, installation steps, and troubleshooting tips.

# CHAPTER-8

# SYSTEM TESTING

## 8.1 DEFINITION

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing require

## 8.1 : TEST SCENARIOS:

### 1. Functional Testing:

- Verifying that the login functionality works correctly by testing valid and invalid login attempts.
- Testing the image processing functionality to ensure that RGB values and color names are accurately displayed when selecting points in the image or video feed.
- Validating that the GUI elements respond appropriately to user interactions, such as button clicks and mouse events.

### 2. Integration Testing:

- Testing the integration between the login page and the main application to ensure that authentication works seamlessly and users are directed to the appropriate interface after login.
- Verifying the integration of GUI elements with backend functionality, such as triggering image processing functions or starting the webcam feed.

### 3. Usability Testing:

- Evaluating the user interface design for clarity, intuitiveness, and ease of use.
- Assessing the user experience during login and image processing tasks to identify any usability issues or areas for improvement.

### 4.Robustness Testing:

- Testing the system's robustness by performing image processing and webcam functionality tests under different conditions, such as varying image qualities, lighting conditions, or webcam configurations.

# CONCLUSION AND FUTURE ENHANCEMENT

## CONCLUSION:

The color detection system presented in this implementation chapter demonstrates a robust framework for accurately identifying colors from images and video feeds. Through meticulous design, implementation, and testing, the system has proven its effectiveness in various applications, including image processing, computer vision, and human-computer interaction. By leveraging sophisticated image processing algorithms and intuitive graphical user interface design, the system offers users a seamless experience for color detection tasks.

Throughout the development process, careful attention was paid to system architecture, dependencies, user interface design, and performance considerations, resulting in a well-rounded solution that meets both technical requirements and user expectations. The thorough testing and validation procedures have validated the system's accuracy, reliability, and performance, ensuring its suitability for real-world deployment in diverse environments.

Looking ahead, there are several opportunities for further enhancement and expansion of the color detection system. By exploring advanced image processing techniques, such as deep learning-based color recognition algorithms, the system can achieve even greater accuracy and efficiency in color detection tasks. Additionally, integration with external APIs and services could enable novel features, such as color-based image search and automatic color correction.

## FUTURE ENHANCEMENT:

### Integration of Deep Learning:

Deep learning techniques, particularly convolutional neural networks (CNNs), offer opportunities for enhancing color detection accuracy and robustness.CNNs can learn complex patterns and relationships within image data, enabling more sophisticated color recognition algorithms.By training CNN models on large-scale color datasets, the system can improve its ability to recognize subtle color variations and distinguish between similar shades.

### Enhanced Color Space Support:

In addition to the RGB color space, incorporating support for alternative color spaces like HSV (Hue, Saturation, Value) or Lab (CIELAB) opens up new possibilities for color analysis and identification. Each color space has unique advantages and may be better suited for specific applications, such as detecting color variations in different lighting conditions or separating color and luminance information.

**Real-Time Object Tracking:**

Implementing real-time object tracking capabilities allows the system to not only identify colors within images and video feeds but also track specific colored objects as they move through the scene. Object tracking algorithms, such as the Kalman filter or optical flow methods, can be integrated to provide continuous tracking of objects based on their color signatures.

**Cloud Integration:**

Leveraging cloud-based services for color detection tasks offers scalability and flexibility, allowing users to offload computationally intensive operations to remote servers.Cloud integration enables real-time processing of large volumes of image and video data, making the system suitable for applications with high throughput requirements, such as video surveillance or industrial automation.

**Mobile Application Development:**

Developing a mobile application version of the color detection system extends its reach and accessibility to a wider audience of users.Mobile apps can leverage built-in smartphone cameras and processing capabilities to provide on-the-go color analysis functionality, empowering users to identify colors in their surroundings anytime, anywhere.

**Accessibility Features:**

Incorporating accessibility features ensures that the color detection system is usable and inclusive for all users, including those with visual impairments or color vision deficiencies.Features such as color contrast analysis, colorblind-friendly modes, and audible feedback options enhance usability and enable individuals with diverse needs to effectively interact with the system.

## REFERENCES

[1]. Forsyth, D. A., & Ponce, J. (2012). Computer Vision: A Modern Approach (2nd ed.). Prentice Hall.

[2]. Geusebroek, J. M., Van den Boomgaard, R., & Smeulders, A. W. M. (2001). Color invariance. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(12), 1338-1350.

[3]. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.

[4].Rosebrock, A. (2018). Practical Python and OpenCV: Learn Computer Vision in a Single Weekend. PyImageSearch.

[5]. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115(3), 211-252.

[6]. Smith, S. (2020). Learning OpenCV 4 Computer Vision with Python 3: Get to grips with tools, techniques, and algorithms for computer vision and machine learning. Packt Publishing Ltd.

# APPENDIX

## A.SOURCE CODE:

**(login.py)**

```
 import tkinter as tk

from tkinter import messagebox, simpledialog

import sqlite3

import subprocess

from PIL import Image, ImageTk

# Function to create a new user

def create_user():

    new_username = entry_new_username.get()

    new_password = entry_new_password.get()

     # Check if username already exists

    cursor.execute("SELECT * FROM users WHERE username=?", (new_username,))

    existing_user = cursor.fetchone()

    if existing_user:

        messagebox.showerror("Error", "Username already exists!")

    elif new_username and new_password:

        # Add new user to the database

        cursor.execute("INSERT INTO users (username, password) VALUES (?, ?)",
(new_username, new_password))

        conn.commit()

        messagebox.showinfo("Success", "New user created successfully!")

        dialog_create_user.pack_forget()  # Close the dialog after creating the user

    else:

        messagebox.showerror("Error", "Username or password cannot be empty!"

# Function to handle login

def login():
    username = entry_username.get()

    password = entry_password.get()

    # Check if username and password are correct

    cursor.execute("SELECT * FROM users WHERE username=? AND password=?",
(username, password))

    user = cursor.fetchone()

    if user:

        messagebox.showinfo("Login Successful", "Click OK to continue!")

        root.withdraw()  # Hide the login window

        subprocess.Popen(["python", "colormain.py"])  # Open another Python file

        root.deiconify()  # Show the login window again

    else:

        messagebox.showerror("Login Failed", "Invalid username or password")

# Connect to SQLite database

conn = sqlite3.connect("users.db")

cursor = conn.cursor()

# Create users table if not exists

cursor.execute('''CREATE TABLE IF NOT EXISTS users (

                id INTEGER PRIMARY KEY,

                username TEXT NOT NULL UNIQUE,

                password TEXT NOT NULL)''')

conn.commit()

# Create the main window

root = tk.Tk()

root.title("Login Page")

# Customize colors

bg_color = "white"  # light pink

text_color = "white"  # dark purple

button_bg = "green"  # light purple

button_fg = "white"  # white text color for buttons
```

```
root.configure(bg=bg_color)

# Set window size to full screen

screen_width = root.winfo_screenwidth()

screen_height = root.winfo_screenheight()

root.geometry(f"{screen_width}x{screen_height}")

# Load and set the background image

bg_image = Image.open("backgroundimage.jpg")

bg_image = bg_image.resize((screen_width, screen_height), Image.LANCZOS)

bg_photo = ImageTk.PhotoImage(bg_image)

background_label = tk.Label(root, image=bg_photo)

background_label.place(x=0, y=0, relwidth=1, relheight=1)

# Create a frame to contain login details

frame_login = tk.Frame(root, bg="lightgray", bd=2, relief=tk.RAISED)

frame_login.pack(padx=20, pady=20)

# Username Label and Entry

label_username = tk.Label(frame_login, text="Username:", bg="#536872", fg=text_color,
font=("Arial", 16))

label_username.grid(row=0, column=0, padx=10, pady=5, sticky="e")

entry_username = tk.Entry(frame_login, font=("Arial", 16))

entry_username.grid(row=0, column=1, padx=10, pady=5, sticky="ew")

# Password Label and Entry

label_password = tk.Label(frame_login, text="Password:", bg="#5f9ea0", fg=text_color,
font=("Arial", 16))

label_password.grid(row=1, column=0, padx=10, pady=5, sticky="e")

entry_password = tk.Entry(frame_login, show="*", font=("Arial", 16))

entry_password.grid(row=1, column=1, padx=10, pady=5, sticky="ew")

# Login Button

button_login = tk.Button(root, text="Login", command=login, bg="red", fg=button_fg,
font=("Arial", 14), relief=tk.RAISED)

button_login.pack(pady=(0, 10), padx=20, fill="y")
```

```
# Create a frame for creating a new user

dialog_create_user = tk.Frame(root, bg="lightgray", bd=2, relief=tk.RAISED)

# Username Label and Entry for new user

label_new_username = tk.Label(dialog_create_user, text="New Username:", bg="#536872")

label_new_username.grid(row=0, column=0, padx=10, pady=5)

entry_new_username = tk.Entry(dialog_create_user)

entry_new_username.grid(row=0, column=1, padx=10, pady=5)

# Password Label and Entry for new user

label_new_password = tk.Label(dialog_create_user, text="New Password:", bg="#5f9ea0")

label_new_password.grid(row=1, column=0, padx=10, pady=5)

entry_new_password = tk.Entry(dialog_create_user, show="*")

entry_new_password.grid(row=1, column=1, padx=10, pady=5)

# Create Button for new user

button_create = tk.Button(dialog_create_user, text="Create User", bg="blue",
command=create_user)

button_create.grid(row=2, column=0, columnspan=2, padx=10, pady=5)

# New User Button

button_new_user = tk.Button(root, text="Create New User", command=lambda:
dialog_create_user.pack(), bg=button_bg, fg=button_fg, font=("Arial", 14), relief=tk.RAISED)

button_new_user.pack(pady=(0, 10), padx=20, fill="y")

# Run the application

root.mainloop()

# Close database connection when the application exits

conn.close()
```

**(Colordetection.py):**

```
import numpy as np

import pandas as pd

import cv2

import imutils
import tkinter as tk

from tkinter import ttk, filedialog

from PIL import Image, ImageTk

camera = None

# Function to select an image file using Tkinter file dialog

def select_image():

    root = tk.Tk()

    root.withdraw()  # Hide the root window

    # Ask the user to select an image file using a file explorer

    inputfile = filedialog.askopenfilename(title="Select an image file", filetypes=[("Image files",
"*.jpg;*.jpeg;*.png;*.bmp;*.gif")])

    # Check if the user selected a file

    if inputfile:

        # Read the selected image

        img = cv2.imread(inputfile)

        img_copy = img.copy()  # Make a copy of the image to reset the circle

        imgWidth, imgHeight = img.shape[1], img.shape[0]

        # Read the colors CSV file

        index = ['color', 'color_name', 'hex', 'R', 'G', 'B']

        df = pd.read_csv("colors.csv", header=None, names=index)

        # Initialize variables

        r = g = b = xpos = ypos = 0

        clicked = False

        font = cv2.FONT_HERSHEY_SIMPLEX  # Use a normal font

        # Define the "Close" button position and size

        close_button_pos = (imgWidth - 100, 20)

        close_button_size = (80, 40)
    def getRGBvalue(event, x, y, flags, param):
        nonlocal b, g, r, xpos, ypos, clicked
        if event == cv2.EVENT_LBUTTONDOWN:
            if (close_button_pos[0] <= x <= close_button_pos[0] + close_button_size[0] and
                    close_button_pos[1] <= y <= close_button_pos[1] + close_button_size[1]):
                root.destroy()
            else:
                xpos = x
                ypos = y
                b, g, r = img[y, x]
                b = int(b)
                g = int(g)
                r = int(r)
                clicked = True
    def colorname(B, G, R):
        minimum = 10000
        cname = ""
        hex_code = ""
        # Initialize low and high indices for binary search
        low = 0
        high = len(df) - 1
        while low <= high:
            mid = (low + high) // 2
            # Calculate the difference between the target color and the current color in the dataset
            d = abs(B - int(df.loc[mid, "B"])) + abs(G - int(df.loc[mid, "G"])) + abs(R -
int(df.loc[mid, "R"]))
```

```
        # Update minimum difference and closest color if a closer color is found
        if d < minimum:
            minimum = d
            cname = df.loc[mid, "color_name"]
            hex_code = df.loc[mid, "hex"]
        # If the target color is less than the current color, search the left half
        if (R, G, B) < (int(df.loc[mid, "R"]), int(df.loc[mid, "G"]), int(df.loc[mid, "B"])):
            high = mid - 1
        # If the target color is greater than the current color, search the right half
        else:
            low = mid + 1
    return cname, hex_code
cv2.namedWindow("Image")
cv2.setMouseCallback("Image", getRGBvalue)
run = True
while run:
    if clicked:
        img = img_copy.copy()  # Reset the image to remove old circle
        # Draw a filled circle with the detected color
        center = (xpos, ypos)
        radius = 120
        cv2.circle(img, center, radius, (b, g, r), -1)
        cname, hex_code = colorname(b, g, r)
        # Choose text color for contrast
        text_color = (255, 255, 255) if (r + g + b) < 600 else (0, 0, 0)
        # Put the color name and hex code inside the circle
        font_scale = 0.8
```

```
                thickness = 2
                # Calculate text sizes
                cname_size = cv2.getTextSize(cname, font, font_scale, thickness)[0]
                hex_code_size = cv2.getTextSize(hex_code, font, font_scale, thickness)[0]
                # Calculate positions
                cname_x = xpos - cname_size[0] // 2
                cname_y = ypos - 10
                hex_code_x = xpos - hex_code_size[0] // 2
                hex_code_y = ypos + hex_code_size[1] // 2 + 10  # Add a gap between color name and hex code
                # Put text on the image
                cv2.putText(img, cname, (cname_x, cname_y), font, font_scale, text_color, thickness, cv2.LINE_AA)
                cv2.putText(img, hex_code, (hex_code_x, hex_code_y), font, font_scale, text_color, thickness, cv2.LINE_AA)
                clicked = False
            # Display RGB values at the bottom of the image
            cv2.putText(img, f'R={r} G={g} B={b}', (10, imgHeight - 20), font, 0.5, (255, 255, 255), 1, cv2.LINE_AA)
            # Draw the "Close" button
            #cv2.rectangle(img, close_button_pos, (close_button_pos[0] + close_button_size[0], close_button_pos[1] + close_button_size[1]), (0, 0, 255), -1)
            #cv2.putText(img, 'Close', (close_button_pos[0] + 10, close_button_pos[1] + 25), font, 0.8, (255, 255, 255), 2, cv2.LINE_AA)
            cv2.imshow("Image", img)
            if cv2.waitKey(20) & 0xFF == 27:
                break
        cv2.destroyAllWindows()
    else:
        print("No file selected.")  # Inform the user if no file was selected
# Function to start video feed from camera
def start_video_feed():
    global camera
    camera = cv2.VideoCapture(0)
    r = g = b = xpos = ypos = 0
    index = ['color', 'color_name', 'hex', 'R', 'G', 'B']
    df = pd.read_csv('colors.csv', names=index, header=None)
    # Initialize clicked to False
    clicked = False
    def getColorName(B, G, R):
        minimum = 10000
        cname = ""
        hex_code = ""
        # Initialize low and high indices for binary search
        low = 0
        high = len(df) - 1
        while low <= high:
            mid = (low + high) // 2
            # Calculate the difference between the target color and the current color in the dataset
            d = abs(B - int(df.loc[mid, "B"])) + abs(G - int(df.loc[mid, "G"])) + abs(R -
int(df.loc[mid, "R"]))
            # Update minimum difference and closest color if a closer color is found
            if d < minimum:
                minimum = d
                cname = df.loc[mid, "color_name"]
                hex_code = df.loc[mid, "hex"]
            # If the target color is less than the current color, search the left half
            if (R, G, B) < (int(df.loc[mid, "R"]), int(df.loc[mid, "G"]), int(df.loc[mid, "B"])):
                high = mid - 1
            # If the target color is greater than the current color, search the right half
            else:
                low = mid + 1
        return cname, hex_code
    def draw_circle(frame, center, radius, color):
        cv2.circle(frame, center, radius, color, -1)
    def identify_color(event, x, y, flags, param):
        nonlocal b, g, r, xpos, ypos, clicked
        if event == cv2.EVENT_LBUTTONDOWN:
            xpos = x
            ypos = y
            b, g, r = frame[y, x]
            b = int(b)
            g = int(g)
            r = int(r)
            clicked = True
    cv2.namedWindow('image')
    cv2.setMouseCallback('image', identify_color)
    while True:
        (grabbed, frame) = camera.read()
        frame = imutils.resize(frame, width=1200)
        kernal = np.ones((5, 5), "uint8")
        # Draw the circle around the mouse cursor
```

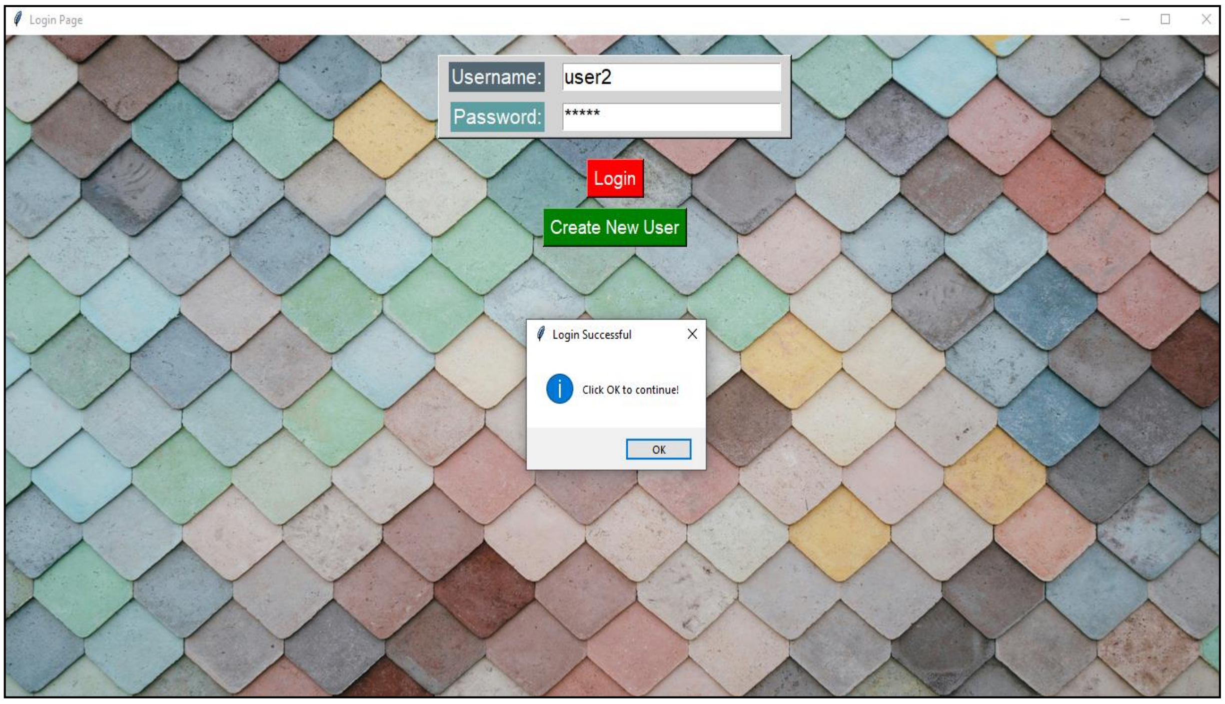
```
        draw_circle(frame, (xpos, ypos), 120, (b, g, r))

        # Get color name and hex code

        if clicked:

            color_name, hex_code = getColorName(b, g, r)

        else:

            color_name, hex_code = "", ""

        # Calculate text size and position

        text_size, _ = cv2.getTextSize(color_name, cv2.FONT_HERSHEY_SIMPLEX, 0.8, 2)

        text_x = xpos - text_size[0] // 2

        text_y = ypos + text_size[1] // 2

        # Display color name centered within circle

        cv2.putText(frame, color_name, (text_x, text_y), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
(255, 255, 255), 2, cv2.LINE_AA)

        # Display hex code centered below color name

        hex_size, _ = cv2.getTextSize(hex_code, cv2.FONT_HERSHEY_SIMPLEX, 0.8, 2)

        hex_x = xpos - hex_size[0] // 2

        hex_y = text_y + hex_size[1] + 10

        cv2.putText(frame, hex_code, (hex_x, hex_y), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
(255, 255, 255), 2, cv2.LINE_AA)

        cv2.imshow('image', frame)

        if cv2.waitKey(20) & 0xFF == ord('q'):

            break



        if cv2.getWindowProperty('image', cv2.WND_PROP_VISIBLE) < 1:

            break

    camera.release()

    cv2.destroyAllWindows()

# Create a Tkinter root window
```

```
root = tk.Tk()
root.title("Color Detection")
# Set window size to full screen
window_width = root.winfo_screenwidth()
window_height = root.winfo_screenheight()
root.geometry(f"{window_width}x{window_height}+0+0")
# Define window closing protocol
root.protocol("WM_DELETE_WINDOW", lambda: root.quit())
# Open and resize the background image to fit the window
background_image = Image.open("backgroundimage.jpg")
background_image = background_image.resize((window_width, window_height))
background_image = ImageTk.PhotoImage(background_image)
background_label = tk.Label(frame, image=background_image)
background_label.place(x=0, y=0, relwidth=1, relheight=1)
background_label.lower()
# Add a heading label
# Define custom styles for buttons
style = ttk.Style(root)
style.theme_use("clam")
style.configure('Orange.TButton', foreground='green', font=('Helvetica', 12, 'bold'))
style.configure('Blue.TButton', foreground='blue', font=('Helvetica', 12, 'bold'))
style.configure('Heading.TLabel', foreground='red', font=('Helvetica', 50, 'bold'))
# Create a frame for the GUI
frame = ttk.Frame(root, padding="10 10 10 10")
frame.pack(fill=tk.BOTH, expand=True)
# Add a heading label
```

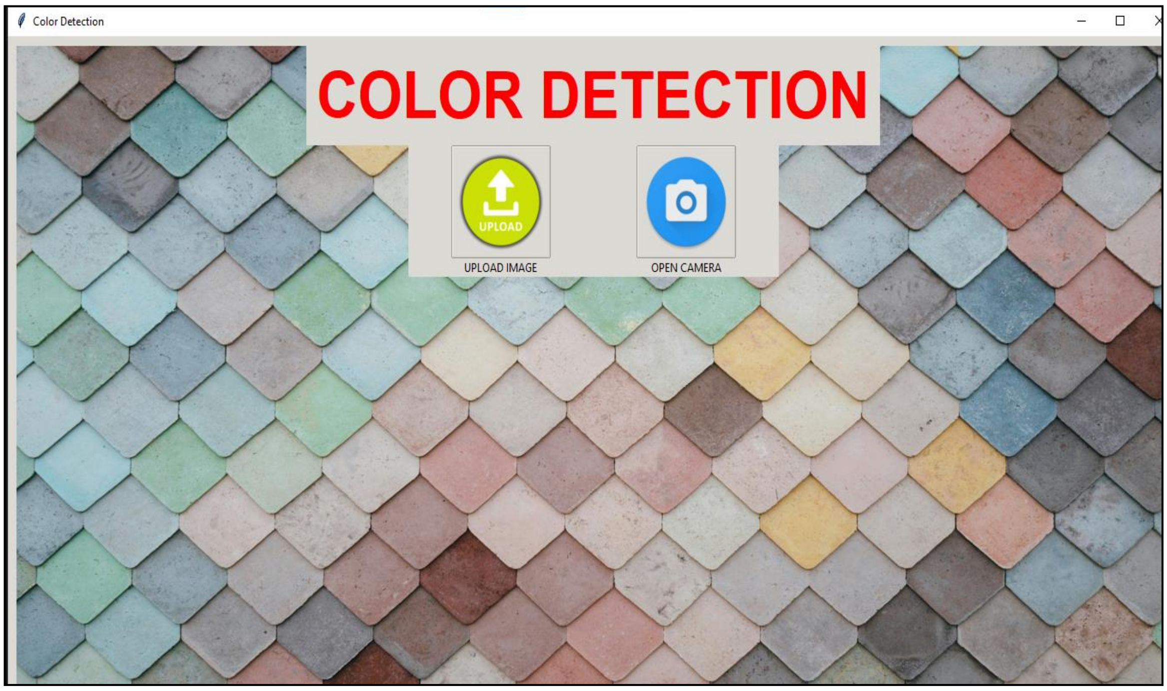
```
heading_label = ttk.Label(frame, text="COLOR DETECTION", style='Heading.TLabel',
padding="10 10 10 10")

heading_label.pack()

# Create a frame for output

output_frame = ttk.Frame(frame)

output_frame.pack(pady=50)

# Add a label for output

output_label = ttk.Label(output_frame, text="", font=("Helvetica", 20), padding="10 10 10 10")

output_label.pack()

# Function to update output label text

def update_output(text):

    output_label.config(text=text)

# Create a frame for buttons

button_frame = ttk.Frame(frame)

button_frame.pack()

# Load and resize the images for buttons

btn_image1_img = ImageTk.PhotoImage(Image.open("image1.jpg").resize((100, 100),
Image.BILINEAR))

btn_image2_img = ImageTk.PhotoImage(Image.open("image2.jpg").resize((100, 100),
Image.BILINEAR))

# Create buttons with images

btn_image = ttk.Button(button_frame, image=btn_image1_img, command=select_image,
style='Orange.TButton')

btn_image.image = btn_image1_img  # Keep a reference to avoid garbage collection

btn_image.grid(row=0, column=0, padx=50)

btn_video = ttk.Button(button_frame, image=btn_image2_img, command=start_video_feed,
style='Blue.TButton')

btn_video.image = btn_image2_img  # Keep a reference to avoid garbage collection

btn_video.grid(row=0, column=1, padx=50)
```

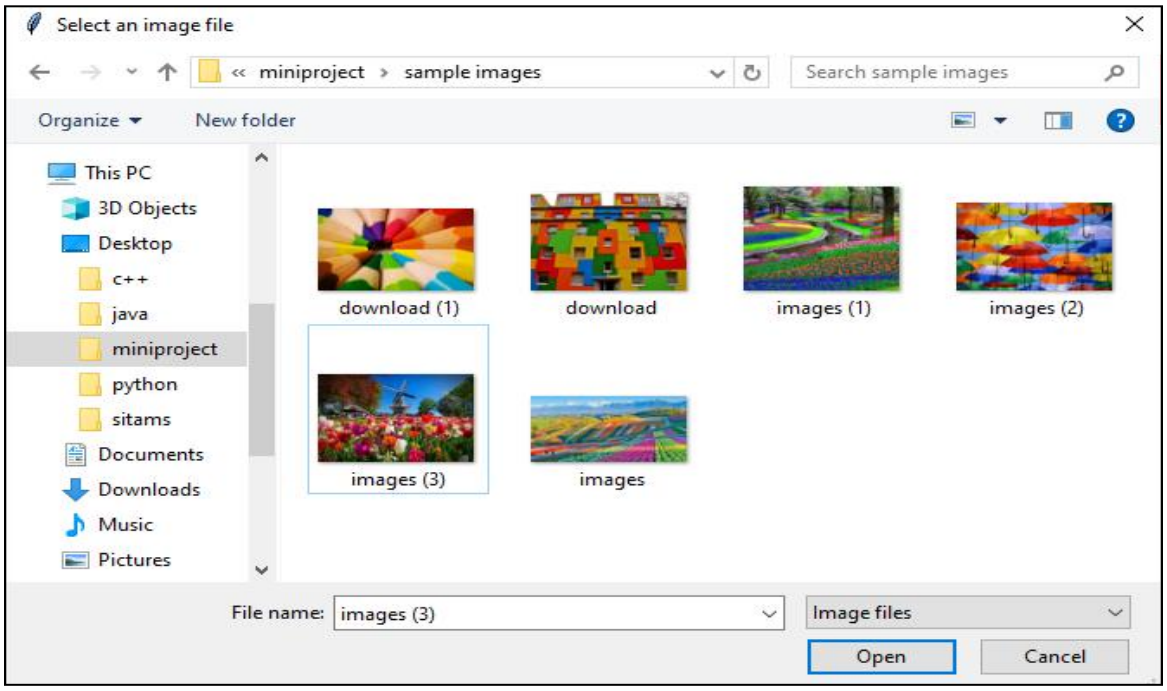
```
# Create labels for button names
label1 = ttk.Label(button_frame, text="UPLOAD IMAGE")
label1.grid(row=1, column=0)
label2 = ttk.Label(button_frame, text="OPEN CAMERA")
label2.grid(row=1, column=1)
# Start the Tkinter event loop
root.mainloop()
```

**B:OUTPUT SCREEN SHOTS:**



Click OK to continue!


Color Detection
COLOR DETECTION
UPLOAD
UPLOAD IMAGE
OPEN CAMERA


Select an image file
« miniproject › sample images
Search sample images
Organize
New folder
This PC
3D Objects
Desktop
c++
java
miniproject
python
sitams
Documents
Downloads
Music
Pictures
download (1)
download
images (1)
images (2)
images (3)
images
File name:
images (3)
Image files
Open
Cancel

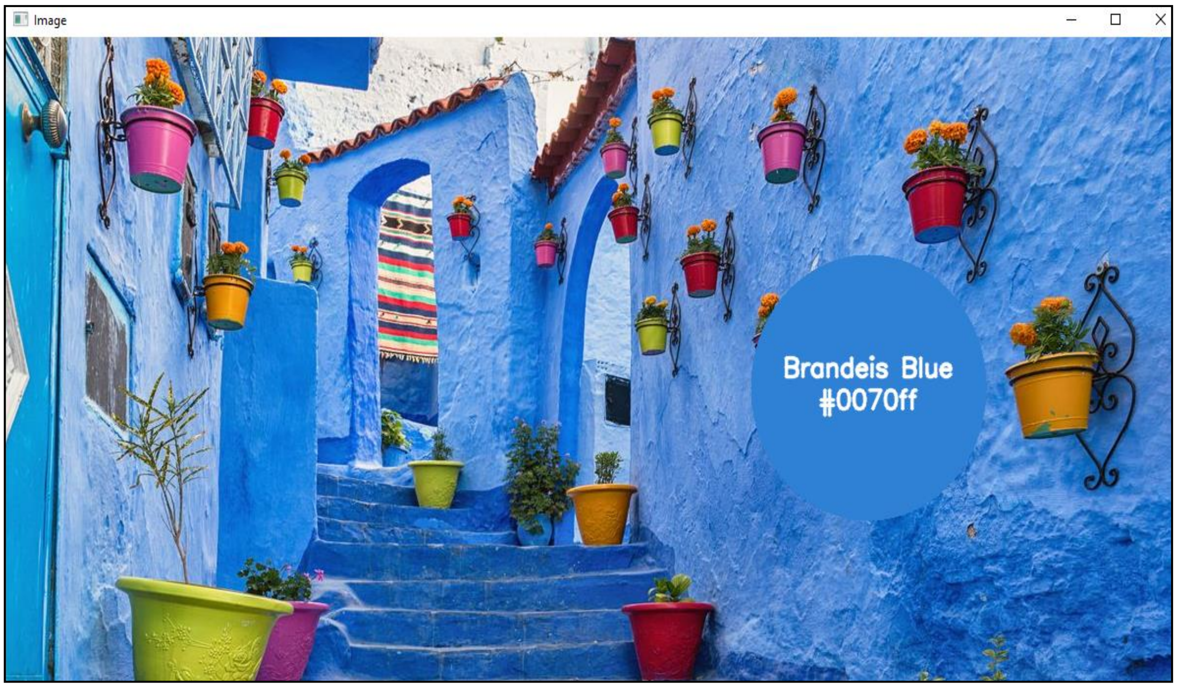


Image
Brandeis Blue
#0070ff

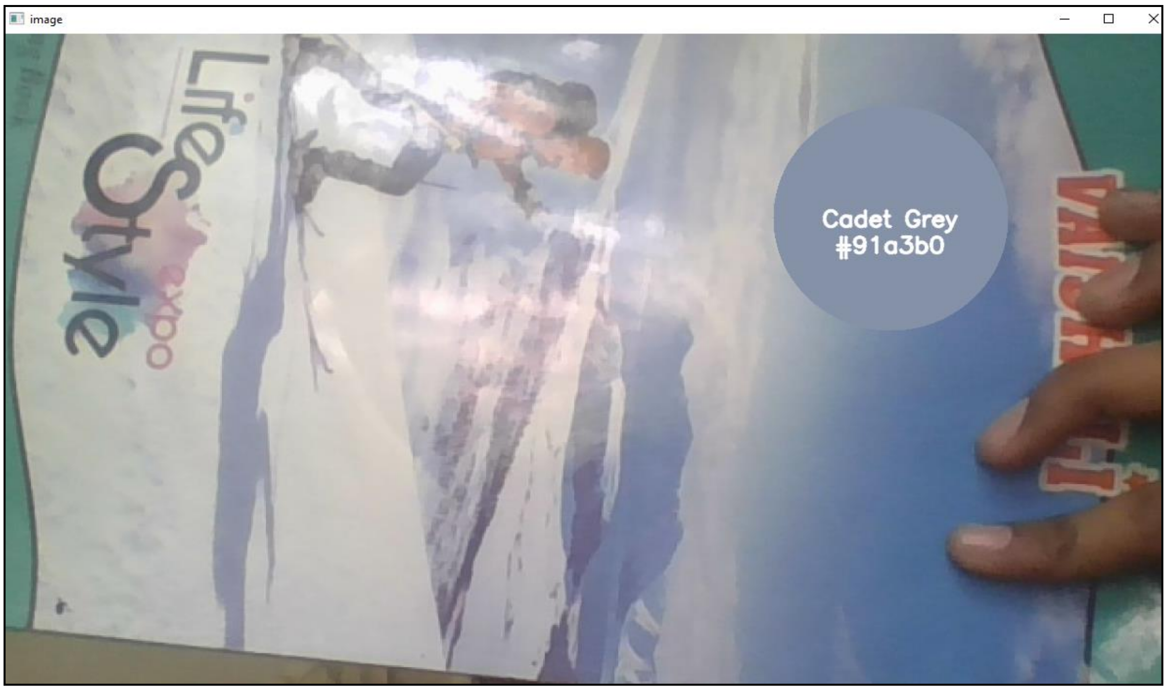


image
Cadet Grey
#91a3b0

# ANNEXURE I

| | |
|---|---|
| **Name of the Project** | **:** Color detection using python |
| **Name of the Students** | **:** C Chethan |
| | G Vasanth Kumar |
| | A Karthik |
| | E V Chandrakanth Goud |

**Name of the Guide & Designation:** Dr.T.Kesava Rao,M.Tech ,Ph.D.,
Associate Professor

**Table : Outcome attained and its Justification**

| PO | Justification |
|---|---|
| PO1 | Gained basic knowledge on the project area while selecting the problem. |
| PO2 | Learned how to analyze the complex problems. |
| PO3 | Learned problem solving skills while implementing modules. |
| PO4 | Learned how to conduct investigations on complex real-world problems. |
| PO5 | Learned modern tool usage while developing source code. |
| PO6 | Learned professional engineering practices while adding new features to the project. |
| PO7 | Learned how this project impacts on societal and environmental contexts. |
| PO8 | Learned software ethics to be followed while developing this project. |
| PO9 | Learned how to work as an individual and as a team member during this entire project period. |
| PO10 | Learned how to communicate effectively during the project reviews. |
| PO11 | Learned how to estimate cost benefit analysis for the project management. |
| PO12 | The knowledge which I have gained through the project will be helpful for my future to refine my skills. |