

A Mini Project Report On

OBJECT TACKING

A project report submitted to
**GURU NANAK INSTITUTIONS TECHNICAL CAMPUS,
IBRAHIMPATNAM, HYDERABAD.**
In partial fulfilment of the requirement for the award of the degree of
BACHELOR OF TECHNOLOGY

Submitted by

**G.VASUKI NANDAN
P.SARATH SURYA BHARADWAJ
V. HARISH**



**SCHOOL OF ENGINEERING & TECHNOLOGY
GURU NANAK INSTITUTIONS TECHNICAL CAMPUS
(An UGC Autonomous Institution - Hyderabad)
Ibrahimpatnam, Ranga Reddy District -501506
Telangana, India. 2019-2020**

INDEX

S.No.	TOPIC	PAGE No.
1	INTRODUCTION	
	1.1 EXISTING SYSTEM	01
	1.2 PROPOSED SYSTEM	01
2	LITERATURE SURVEY	
	2.1 PROJECT LITERATURE	02
	2.2 INTRODUCTION TO PYTHON	02
	2.2.1 PYTHON TECHNOLOGY	03
	2.2.2 MVC ARCHITECTURE	04
	2.2.3 TINTER	05
	2.3 LIBRARIES SPECIFIC TO PROJECT	06
	2.3.1 IMGUTILS	06
	2.3.2 OPEN CV	06
3	SYSTEM ANALYSIS AND REQUIREMENTS	
	3.1 FEASIBILITY STUDY	07
	3.1.1 ECONOMIC FEASIBILITY	07
	3.1.2 TECHNICAL FEASIBILITY	08
	3.1.3 SOCIAL FESIBILITY	08
	3.2 SOFTWARE AND HARDWARE	08-09
	REQUIREMENTS	
	3.3 PERFORMANCE REQUIREMENTS	09

S.NO.	TOPIC	PAGE No.
4	SOFTWARE DESIGN	
	4.1 USE CASE DIAGRAM	10
	4.2 CLASS DIAGRAM	10
	4.3 ACTIVITY DIAGRAM	11
	4.4 ACTIVITY DIAGRAM	12
	4.5 ACTIVITY DIAGRAM	12
	4.6 ACTIVITY DIAGRAM	12
	4.7 ACTIVITY DIAGRAM	12
5	CODE TEMPLATES	
	Folder View	13
	5.1 APP CODE	14
	5.2 AuthModel.py	14
	5.3 AuthController.py	15
	5.4.1 AuthModel.py	16
	5.4.2 Details_view.py	19
	5.4.3 Home.py	21
	5.5.1 ColorDetect.py	23
	5.5.2 Db.py	25
6	OUTPUTS	28-30
7	SYSTEM TESTING	
	7.1 INTRODUCTION	31
	7.2 TYPES OF TEST	31

	7.2.1 UNIT TESTING	31
	7.2.2 INTEGRATION TESTING	32
	7.2.3 FUNCTIONAL TEST	32
	7.2.4 SYSTEM TEST	33
	7.2.5 WHITE BOX TEST	33
	7.2.6 BLACK BOX TEST	33
	7.2.7 ACCEPATANCE TESTING	34
	7.3 TEST APPROACH	34
8	8.1 CONCLUSION	35
	8.2 REFERENCES	35

CHAPTER 1

INTRODUCTION

Tracking objects in both structured and unstructured environments is one of the most challenging tasks in computer vision and artificial intelligence research. This project introduces a new computer vision-based object tracking for various applications. Object tracking has been driven by an increase in power available in both hardware and software. The method uses a webcam camera that performs in real-time and also provides an image. In this method, the system keeps learning about the appearance of the object during real-time.

It is related to many real-time applications like vehicle perception, video surveillance, and so on. To overcome the issue of detection, tracking related to object movement, and appearance. Most of the algorithms focus on the tracking algorithm to smoothen the video sequence. On the other hand, few methods use the prior available information about object shape, color, texture, and so on.

1.1 EXISTING SYSTEM

The existing system of object tracking is limited when storage of a video or an image. The object is being tracked for the images or videos that are stored.

1.2 PROPOSED SYSTEM

The system is more efficient and is capable of detecting the objects live. Unlike the images, it also tracks the object in a video and the direction of the object in which it is moved.

CHAPTER 2

LITERATURE SURVEY

2.1 PROJECT LITERATURE

In various fields, there is a necessity to detect the target object and also track them effectively while handling occlusions and other included complexities. Many researchers (Almeida and Guting 2004, Hsiao-Ping Tsai 2011, Nicolas Papadakis and Aure lie Bugeau 2010) attempted for various approaches in object tracking. The nature of the techniques largely depends on the application domain. Some of the research works which made the evolution of proposed work in the field of object tracking are depicted.

2.2 INTRODUCTION TO PYTHON

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Python is developed by Guido van Rossum. Guido van Rossum started implementing Python in 1989. The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation. The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well.

2.2.1 PYTHON TECHNOLOGY

Python is known for its general-purpose nature that makes it applicable in almost every domain of software development. Python as a whole can be used in any sphere of development. Here, we are specifying application areas where python can be applied.

1.WEB APPLICATION

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautiful soup, Feedparser, etc. It also provides Frameworks such as Django, Pyramid, Flask, etc to design and develop web-based applications. Some important developments are PythonWikiEngines, Pocoo, PythonBlogSoftware, etc.

2. DESKTOP GUI APPLICATIONS

Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

3. SOFTWARE DEVELOPMENT

Python is helpful for the software development process. It works as a support language and can be used to build control and management, testing, etc.

4. SCIENTIFIC AND NUMERIC

Python is popular and widely used in scientific and numeric computing. Some useful libraries and packages are SciPy, Pandas, IPython, etc. SciPy is a group of packages of engineering, science, and mathematics.

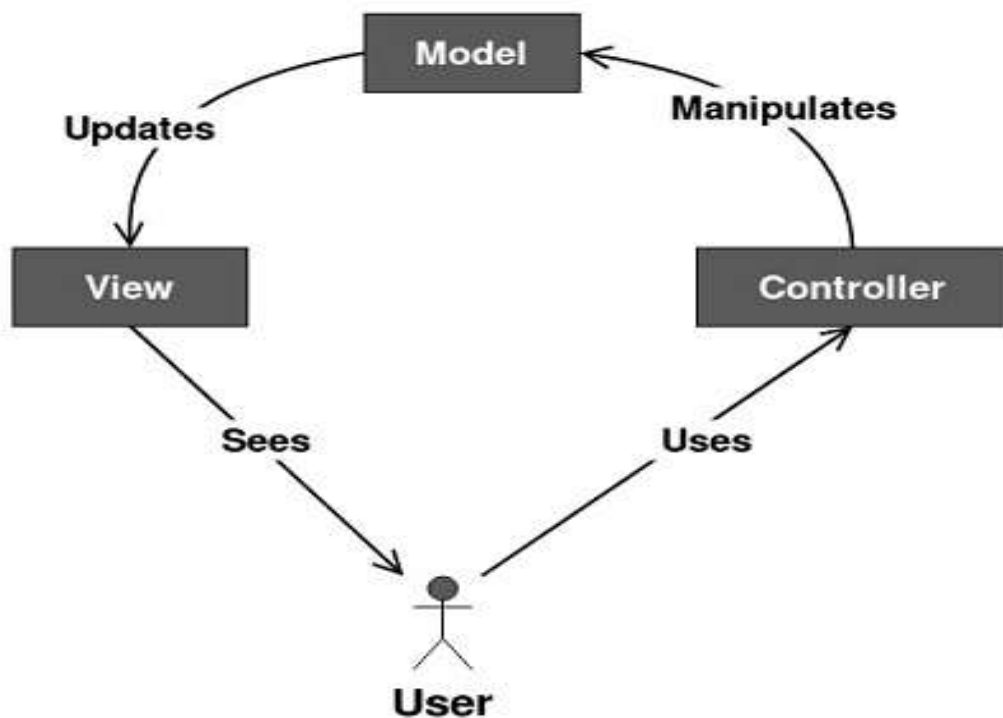
5.BUSINESS APPLICATIONS

Python is used to build Bussiness applications like ERP and e-commerce systems. Tryton is a high-level application platform.

2.2.2 MVC ARCHITECTURE

Model View Controller is the most commonly used design pattern. Developers find it easy to implement this design pattern.

Following is a basic architecture of the Model View Controller –



Let us now see how the structure works.

Model

It consists of pure application logic, which interacts with the database. It includes all the information to represent data to the end-user.

View

The view represents the HTML files, which interact with the end-user. It represents the model's data to the user.

Controller

It acts as an intermediary between view and model. It listens to the events triggered by the view and queries model for the same.

The three components of the MVC pattern are **decoupled** and they are responsible for different things:

The **model** manages the data and defines rules and behaviors. It represents the business logic of the application. The data can be stored in the Model itself or in a database (only the Model has access to the database).

The **view** presents the data to the user. A View can be any kind of output representation: an HTML page, a chart, a table, or even a simple text output. A View should never call its own methods; only a Controller should do it.

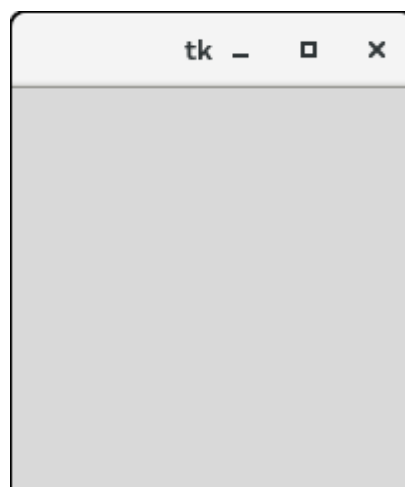
The **controller** accepts the user's inputs and delegates data representation to a View and data handling to a Model.

Since Model, View, and Controller are **decoupled**, each one of the three can be extended, modified, and replaced without having to rewrite the other two.

2.2.3 Tkinter

Tkinter tutorial provides basic and advanced concepts of Python Tkinter. Our Tkinter tutorial is designed for beginners and professionals. Python provides the standard library Tkinter for creating the graphical user interface for desktop-based applications. Developing desktop-based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.

1. Import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.
4. Call the main event loop so that the actions can take place on the user's computer screen.



2.3 LIBRARIES SPECIFIC TO PROJECT

2.3.1 IMGUTILS

This module provides read, write, and resize functions for images. The backends of these functions are automatically changed, depending on the user's environment. The priority of the backends is as below (upper is higher priority):

- OpenCV (cv2)
- scikit-image (skimage)
- numpy (Image dimensions)
- pillow (PIL) (need to be installed)

At least one of these modules needs to be installed to use this module.

2.3.2 Open CV

OpenCV was started at Intel in 1999 by **Gary Bradsky** and the first release came out in 2000. **Vadim Pisarevsky** joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won the 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day.

Currently, OpenCV supports a wide variety of programming languages like C++, Python, Java, etc, and is available on different platforms including Windows, Linux, OS X, Android, iOS, etc. Also, interfaces based on CUDA and OpenCL are under active development for high-speed GPU operations.

OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

CHAPTER 3

SYSTEM ANALYSIS AND REQUIREMENTS

3.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and the business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ◆ **ECONOMICAL FEASIBILITY**
- ◆ **TECHNICAL FEASIBILITY**
- ◆ **SOCIAL FEASIBILITY**

3.1.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.1.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.1.3 SOCIAL FEASIBILITY

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead they must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.2 SOFTWARE AND HARDWARE REQUIREMENTS

3.2.1 Hardware Requirements

Processors : Intel Atom® processor or Intel® Core™ i3 processor.
Disk space : 1 GB.

3.2.2 Software Requirements

Operating systems: Windows* 7 or later, macOS, and Linux.

Front End : python

BackEnd : Tkinter, Sqlite3

3.3 PERFORMANCE REQUIREMENTS

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into the required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use. The requirement specification for any system can be broadly stated as given below:

The system should be able to interface with the existing system

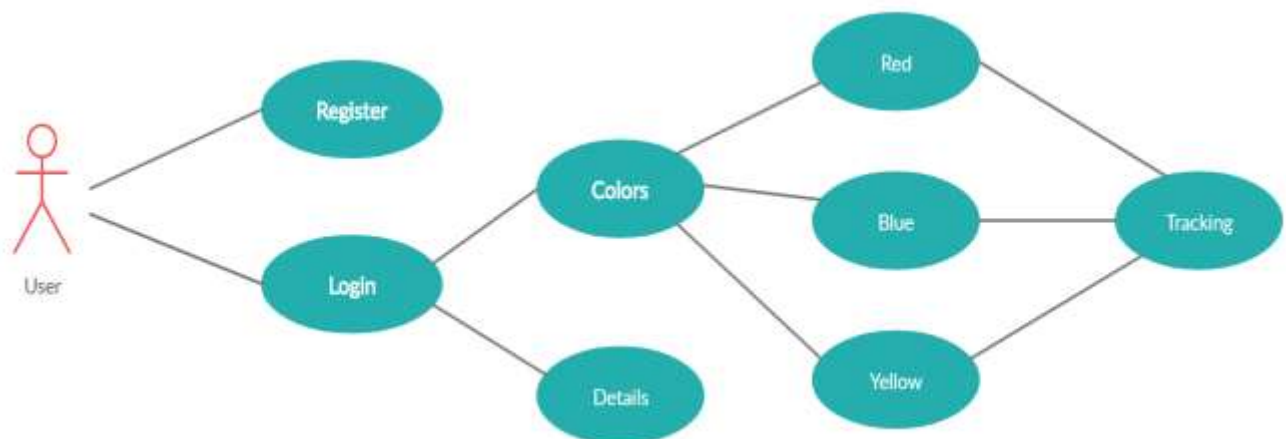
- The system should be accurate
- The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

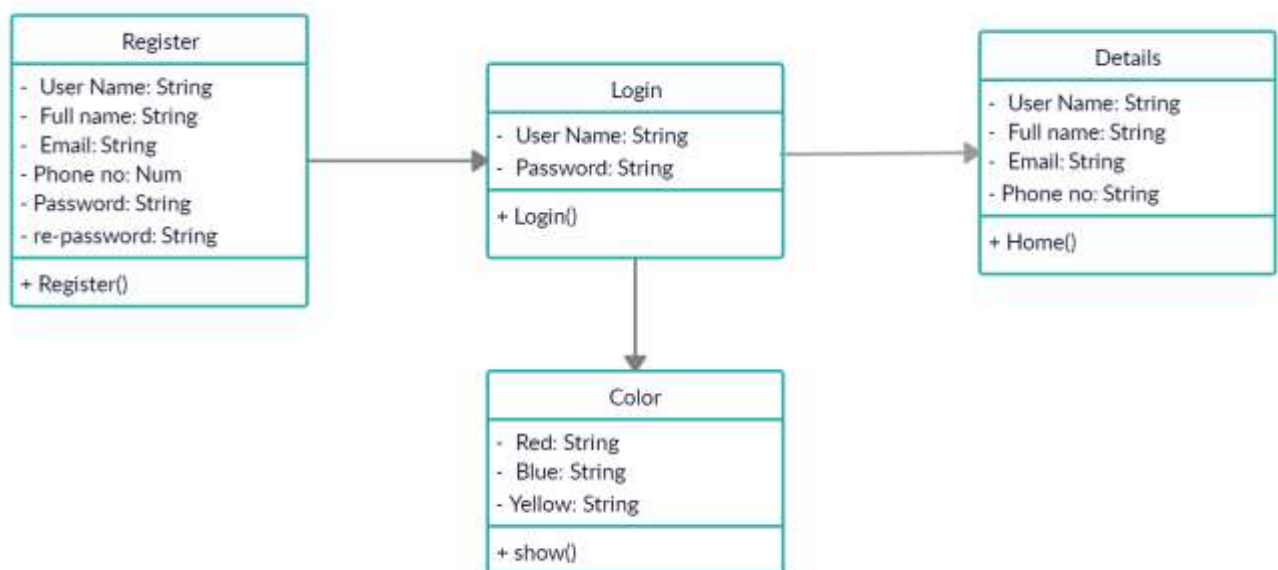
CHAPTER 4

SOFTWARE DESIGN

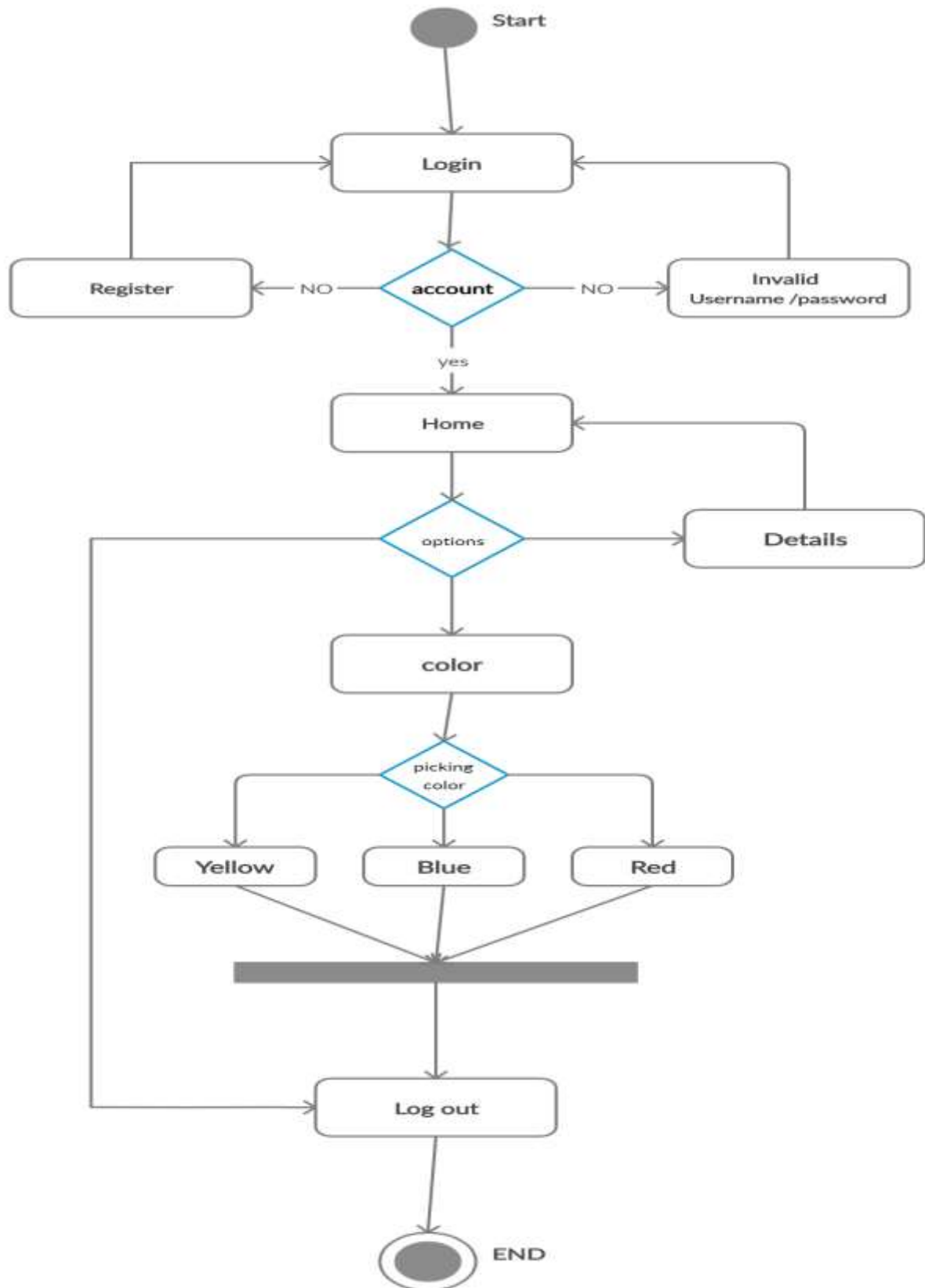
4.1 Use Case Diagram



4.2 Class Diagram



4.3 Activity Diagram



4.4 Login:

Where the user is used to login to the existing account to operate the program functionality. If the account does not exist alert will be sent.

4.5 Register:

When the user opened the application for the first time or to create the new account Register is used. Which creates an Account to the Data-Base

4.5 Details:

The user clicks the Details button on the page it will show the user full name, email, phone no.

4.6 Logout:

When the user clicks the logout button. It will retrieve back to the Login page

4.7 Tracking:

Externally It will show the options to choose the color to track.

Internally It will take the color which to select. That particular color selects the HSV color lower range and upper range.

When the webcam is opened by Cv2. It will detect the color between HSV lower range and upper range is present or not. If present that color it will round up the color and shows the direction. In Cv2 each frame is transferred by the millisecond. It will Detect automatically. Until we press the ESC button it will continuously Detect. If we press ESC it will Terminate

CHAPTER 5

CODE TEMPLATES

Folder View:

5.1.App.py

5.2.Models –

5.2.1 Authmodel.py

5.3.Controllers –

5.3.1 AuthController.py

5.4.Views –

5.4.1 Authview.py

5.4.2 Details_view.py

5.4.3 home.py

5.5.Lib –

5.5.1 Colordetect.py

5.5.2 db.py

5.1 APP CODE

```
from Views.AuthView import AuthView
from Views.Home import Home
class App:
    # Starting
    def run(self):
        Auth = AuthView()
        Auth.transfer_control = self.ho
        Auth.run()

    # signout to goback run
    def ho(self):
        home = Home()
        home.tab_so = self.run
        home.hme()

# calling
app = App()
app.run()
```

5.2.1 Authmodel.py

```
from lib.db import *
class AuthModel:

    def __init__(self):
        # connectong the Database
        self.con =connect("db/app.db")

    # adding the database
    def ad(self,un,na,pa,em,ph):
        query = f'INSERT INTO log (un,na,pa,em,ph) VALUES("{un}","{na}","{pa}"
,{em}","{ph});'
        se = insert(self.con,query)
        return se

    # checking user name and password
    def show(self,un,pa):
        query = f'SELECT * FROM log WHERE un= "{un}" and pa= "{pa}";'
```

```
se = fetchone(self.con,query)
return se
```

5.3.1 AuthController.py

```
# importing
from models.AuthModel import AuthModel

# class
class AuthController:

    # login recieved login entry from view sending to db
    def login(self,un,pa):

        # checking field is empty or not
        if len(un) ==0:
            mes ="No user Name"
            return mes

        if len(pa) ==0:
            mes ="No password"
            return mes

        am = AuthModel()

        # sending db
        r=am.show(un,pa)

        # printing all
        print(r)

        # for home and details
        global dn
        dn = r

        # if r has value is true
        if r:

            # user found
            mes = 1
            # pop up
            return mes

        else:

            # user not found
            mes = "user not found"
            # pop up
```

```
        return mes

# login recieved register entry from view sending to db
def register(self,un,na,pa,em,ph):

    # checking the filed is empty or not
    if len(un) == 0 or len(na) == 0 or len(pa) == 0 or len(em) == 0 or len
(ph) == 0 :
        mes = "some thing is null"
        return mes

    am = AuthModel()
    # sending to db
    r = am.ad(un,na,pa,em,ph)
    print(r)
    mes = 'Registered'

    # pop up
    return mes

# sending to details_view
def det(self):
    return dn
```

5.4.1 Authmodel.py

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from Controllers.AuthController import AuthController
class AuthView:

    def run(self):
        # creating tk window
        self.window = Tk()

        # Renaming the Title
        self.window.title("Object Tracking")

        # geometry
        #self.window.geometry("250x150")

        # Notebook
        tab_control = ttk.Notebook(self.window)
```

```

# Adding frames
Login_frame = Frame(tab_control, padx = "200", pady = "100")
Register_frame = Frame(tab_control, padx = "200", pady = "100")

# connect Frames
tab_control.add(Login_frame, text = "Login")
tab_control.add(Register_frame, text = "Register")

# presting Frames

tab_control.grid()

# calling Login
self.Login(Login_frame)

# calling Register
self.Register(Register_frame)
self.na = 'temp'

self.window.mainloop()

# creating the Login view
def Login(self, Login_frame):
    print("This is Login")

# creating username
un = Label(Login_frame, text="User Name: ")
un.grid(row=0, column=0)

ue = Entry(Login_frame, width="20")
ue.grid(row=0, column=1)

# creating password
pn = Label(Login_frame, text="Password: ")
pn.grid(row=1, column=0)

pe = Entry(Login_frame, show = "*", width="20")
pe.grid(row=1, column=1)

# login button
lb = Button(Login_frame, text="Login", command=lambda: self.logincontroller( ue.get(), pe.get() ), fg='blue', width=7, height=1)
lb.grid(row=2, column=1, pady="5")
self.na = ue.get()

# sending the login entry to controller
def logincontroller(self, un1, pa1):

```

```
A = AuthController()
mes = A.login(un1,pa1)
global na
na = un1

# pop up
if mes == 1:
    self.window.destroy()
    #print(self.na)
    self.transfer_control()
else:
    messagebox.showinfo('message: ', mes)

def nam(self):
    cra = na
    return cra

# creating Register
def Register(self, Register_frame):
    print("This is Register")

    #creating user name
    run = Label(Register_frame, text="User Name: ")
    run.grid(row=0, column=0)

    rue = Entry(Register_frame, width="22")
    rue.grid(row=0,column=1)

    #full name
    rfn = Label(Register_frame, text="Full Name: ")
    rfn.grid(row=1, column=0)

    rfe = Entry(Register_frame, width="22")
    rfe.grid(row=1,column=1)

    #Email
    ren = Label(Register_frame, text="Email: ")
    ren.grid(row=2, column=0)

    ree = Entry(Register_frame, width="22")
    ree.grid(row=2,column=1)

    #phone no
    rpn = Label(Register_frame, text="Phone No: ")
    rpn.grid(row=3, column=0)
```

```
rpe = Entry(Register_frame, width="22")
rpe.grid(row=3,column=1)

#password
rpan = Label(Register_frame, text="Password: ")
rpan.grid(row=4, column=0)

rpae = Entry(Register_frame, show="*", width="22")
rpae.grid(row=4,column=1)

#retype
rrpn = Label(Register_frame, text="Re-Password: ")
rrpn.grid(row=5, column=0)

rrpe = Entry(Register_frame, show="*", width="22")
rrpe.grid(row=5, column=1)

#button
rb = Button(Register_frame, text="Register", fg='blue',command=lambda:
self.registercontroller( rue.get(), rfe.get(), ree.get(), rpe.get(), rrpe.get
(), rpae.get()) ,width=7, height=1)
rb.grid(row=6, column=1, pady="5")

# sending the Register entry to controller
def registercontroller(self,rue,rfe,ree,rpe,rrpe,rpae):

    # checking the password and re-type password
    if rrpe != rpae:
        messagebox.showwarning(title="password", message="Password Incorre
ct")

    else:
        A = AuthController()
        mes = A.register(rue,rfe,rrpe,ree,rpe)

        # pop up
        messagebox.showinfo('message: ', mes)

        # destroy window
        self.window.destroy()
        self.run()

if __name__ == "__main__":
    A = AuthView()
```

5.4.2 Details_view.py

```
# importing
from tkinter import *
from tkinter import ttk
from Views.AuthView import AuthView
from Controllers.AuthController import AuthController

class details:
    def __init__(self):

        # to revieve from the controller names, email , ....
        C = AuthController()
        self.cd = C.det()

        # to desplay details
        def details(self):
            print("details")
            self.window = Tk()
            self.window.title("Object Tracking")
            self.window.geometry("600x400")

            self.window.configure(background='#f2efe6')

            l1 = Label(self.window,text = "DETAILS",fg="blue" ,font = "Helvetica 20 italic")
            l1.place(relx =0.5, rely = 0.15, anchor = "n")

            # name
            l2 = Label(self.window,text = "NAME:", font = "Helvetica 14 italic")
            l2.place(relx =0.35, rely = 0.30, anchor = "n")

            n1 = Label(self.window,text = self.cd[1], font = "Helvetica 14 italic")
            n1.place(relx =0.5, rely = 0.30, anchor = "n")

            # fullname
            l3 = Label(self.window,text = "FULL NAME:", font = "Helvetica 14 italic")
            l3.place(relx =0.31, rely = 0.40, anchor = "n")

            f1 = Label(self.window,text = self.cd[2], font = "Helvetica 14 italic")
```



```
f1.place(relx =0.5, rely = 0.40, anchor = "n")

# Email
l4 = Label(self.window,text = "EMAIL:", font = "Helvetica 14 italic")
l4.place(relx =0.35, rely = 0.50, anchor = "n")

el = Label(self.window,text = self.cd[4], font = "Helvetica 14 italic"
)
el.place(relx =0.6, rely = 0.50, anchor = "n")

# phone NO:
l4 = Label(self.window,text = "PHONE NO:", font = "Helvetica 14 italic"
")
l4.place(relx =0.31, rely = 0.60, anchor = "n")

pl = Label(self.window,text = self.cd[5], font = "Helvetica 14 italic"
)
pl.place(relx =0.5, rely = 0.60, anchor = "n")

# button

b1 = Button(self.window, text = "HOME",command=self.goto_home, fg="red
", bg="white", width=6, height=2)
b1.place(relx =0.5, rely = 0.7, anchor = "n")

self.window.mainloop()

# destroy the window
def goto_home(self):
    self.window.destroy()
```

5.4.3 home.py

```
# importing
from tkinter import *
from tkinter import ttk
from Controllers.AuthController import AuthController
from Views.AuthView import AuthView
from Views.Details_View import details
from lib.ColorDetect import ColorDetect
class Home:

    def __init__(self):
        A = AuthView()
        self.k = A.nam()
```

```
# home page
def hme(self):
    print("this is Home")

# tkinter
self.window = Tk()
self.window.title("Object Tracking")
self.window.geometry("600x400")

self.window.configure(background='#f2efe6')

# welcome name
l1 = Label(self.window,text = "welcome "+self.k+".", font = "Helvetica
16 italic")
l1.place(relx = 0.0,relly = 0.0)

# obj detect

l2 = Label(self.window,text = "Click any color to Object track", font
= "Helvetica 10 italic")
l2.place(relx =0.35, relly = 0.25)

# blue
l3 = Button(self.window,text = "Blue",command =lambda: self.ot(0), fg=
'blue',bg="white",width=7, height=2)
l3.place(relx = 0.5,relly = 0.38,anchor = 'center')

# red
l4 = Button(self.window,text = "Red", fg='red',bg="white",command = la
mbda: self.ot(1) , width=7, height=2)
l4.place(relx = 0.4,relly = 0.38,anchor = 'center')

# yellow
l5 = Button(self.window,text = "yellow", fg="#9b870c",command =lambda:
self.ot(2) ,bg="white", width=7, height=2)
l5.place(relx = 0.6,relly = 0.38,anchor = 'center')

# escape
ex = Label(self.window, text = "*Press <ESC> to exit the Camera", font
= "Helvetica 10 bold")
ex.place(relx = 0.5, relly = 0.5, anchor = 's')

# user details
l6 = Label(self.window, text = "Click here for User Details", font = "
Helvetica 10 italic")
l6.place(relx = 0.5, relly = 0.6, anchor = 's')
```

```
17 = Button(self.window,text = "User details", command = self.detail ,
fg="black",bg="white", width=11, height=2)
17.place(relx = 0.5, rely = 0.75,anchor = 's')

# sign out
18 = Button(self.window,text="Sign out", command = self.fp, fg="black"
, bg="white", width=6, height=2)
18.place(relx = 0.9, rely = 0.95, anchor='sw')

self.window.mainloop()

# sign out
def fp(self):
    print("Sign Out")
    self.window.destroy()
    self.tab_so()

# to go details view
def detail(self):
    print(self.k)
    self.A = details()
    self.A.details()

# button selection to send

def ot(self,c):
    global d
    d = c
    cd = ColorDetect()
    cd.otd(d)
```

5.5.1 Colordetect.py

```
# importing
import cv2
import numpy as np
import time
from tkinter import ttk
from tkinter import *

class ColorDetect:

    # color detect
    def otd(self,col):
```

```
# blue
if col == 0:
    l = [110,50,50]
    u = [130,255,255]

# red
if col == 1:
    l = [0,50,50]
    u = [10,255,255]

# yellow
if col == 2:
    l = [20,110,110]
    u = [40,255,255]

# Connecting to the Camera
cap = cv2.VideoCapture(0)

# blue range
lower = np.array(l)
upper = np.array(u)

#repeating frames
while True:
    # Converting the Input Into HSV Format
    ret, frame = cap.read()
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Filtering Out Yellow Pixels
    mask = cv2.inRange(hsv, lower, upper)

    # Finding Contours
    contours, hierarchy = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    # Getting the Coordinates of the Rectangle Bounding the Contour
    for contour in contours:
        area = cv2.contourArea(contour)
        k = 0

        if(area > 800):
            x,y,w,h = cv2.boundingRect(contour)

            # circle size
            z = ((w+h)/2)
            z = int(z)
```

```
# direction
if x <= 200:
    xd="Left"
if x < 400 and x > 200:
    xd="Center"
if x > 400:
    xd = "Right"
if y > 150 and y < 275:
    yd="Middle"
if y < 150:
    yd="Top"
if y > 275:
    yd="Bottom"

# print
point=yd+" "+xd
print(point)

# text
frame =cv2.putText(frame,point,(0,50), cv2.FONT_HERSHEY_SIMPLEX, 1,(40, 116, 166 ))

# circle formation
frame = cv2.circle(frame, (x,y),z,(0,0,255),2)
cv2.putText(frame,point, (0,50), cv2.FONT_HERSHEY_SIMPLEX,
1, (0, 255, 255))

#showing
cv2.imshow("Object Tracking", frame)

# break
k = cv2.waitKey(1) & 0xFF
if k == 27:
    break

# destroy
cv2.destroyAllWindows()
cap.release()
```

5.5.2 db.py

```
import sqlite3 as lite

# connect to the database
def connect(database_name):
    try:
        connection = lite.connect(database_name)
        return connection
```

```
except lite.Error as error:
    print("Error while connecting to sqlite", error)

# print sqlite version
def version(connection):
    try:
        cursor = connection.cursor()
        query = "select sqlite_version();"
        cursor.execute(query)
        record = cursor.fetchall()
        print("SQLite Database Version is: ", record)
        cursor.close()
    except lite.Error as error:
        print("Failed to print database version", error)

# insert one record into table
def insert(connection,query):
    try:
        cursor = connection.cursor()
        cursor.execute(query)
        connection.commit()
        print("Successfully inserted")
        cursor.close()
    except lite.Error as error:
        print("Failed to insert data into table", error)

# fetch all records from table
def fetchall(connection,query):
    try:
        cursor = connection.cursor()
        cursor.execute(query)
        records = cursor.fetchall()
        cursor.close()
        return records
    except lite.Error as error:
        print("Failed to fetch records the data from table", error)

# fetch one record from table
def fetchone(connection,query):
    try:
        cursor = connection.cursor()
        cursor.execute(query)
        record = cursor.fetchone()
        cursor.close()
        return record
    except lite.Error as error:
        print("Failed to fetch one record from table", error)
```

```
# update one record in the table
def update(connection,query):
    try:
        cursor = connection.cursor()
        cursor.execute(query)
        connection.commit()
        print("Successfully updated one record")
    except lite.Error as error:
        print("Failed to update data into table", error)

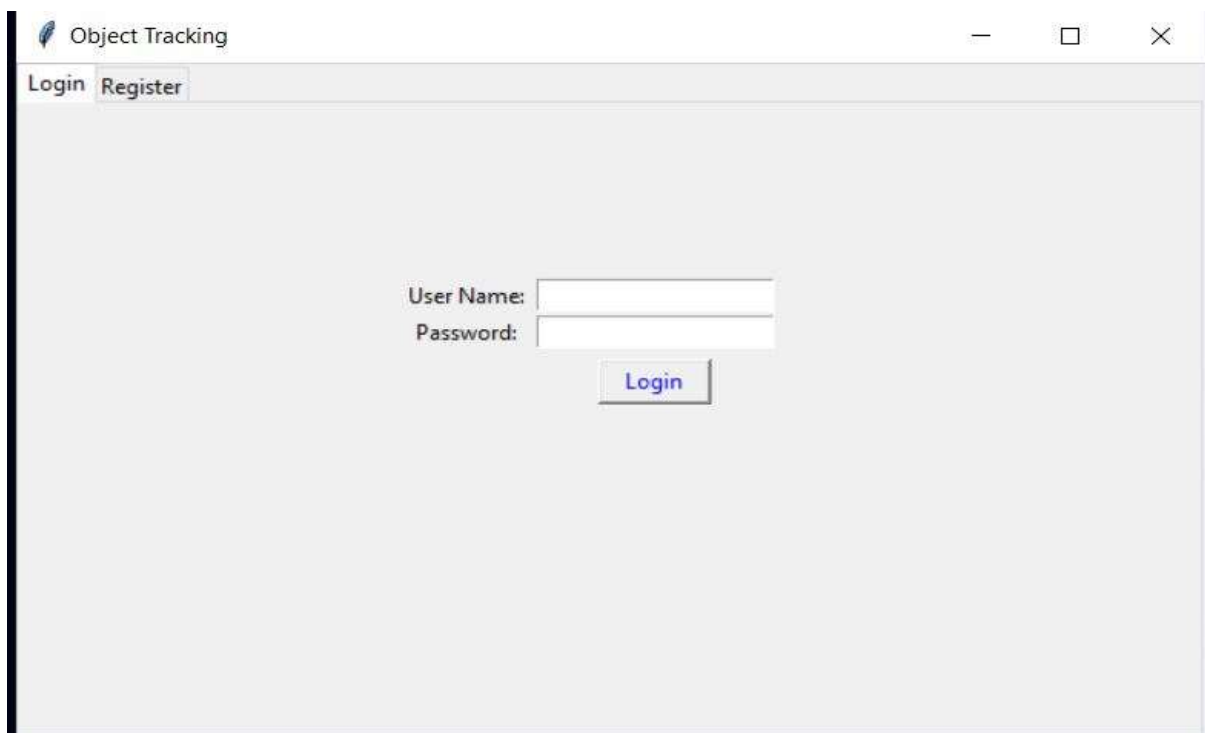
# delete one record from table
def delete(connection,query):
    try:
        cursor = connection.cursor()
        cursor.execute(query)
        connection.commit()
        print("Successfully deleted one record")
    except lite.Error as error:
        print("Failed to delete one record ", error)

# close the connection
def close(connection):
    connection.close()
```

CHAPTER 6

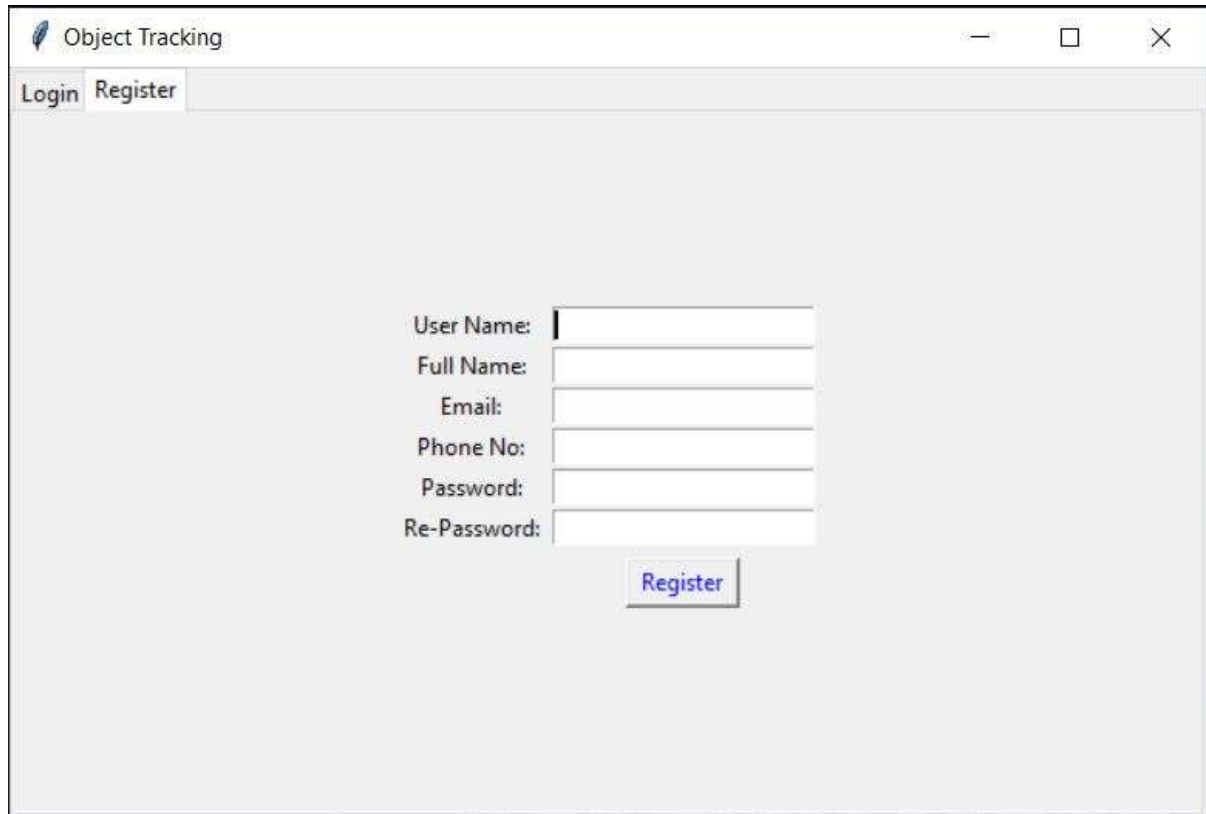
OUTPUTS

1.



The screenshot shows a web application window titled "Object Tracking". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Below the title bar, there is a navigation bar with two tabs: "Login" and "Register". The "Login" tab is currently selected. The main content area of the window is light gray and contains a login form. The form consists of two text input fields: "User Name:" and "Password:". Below these fields is a "Login" button with blue text. The "Register" tab is also visible but not selected.

2.

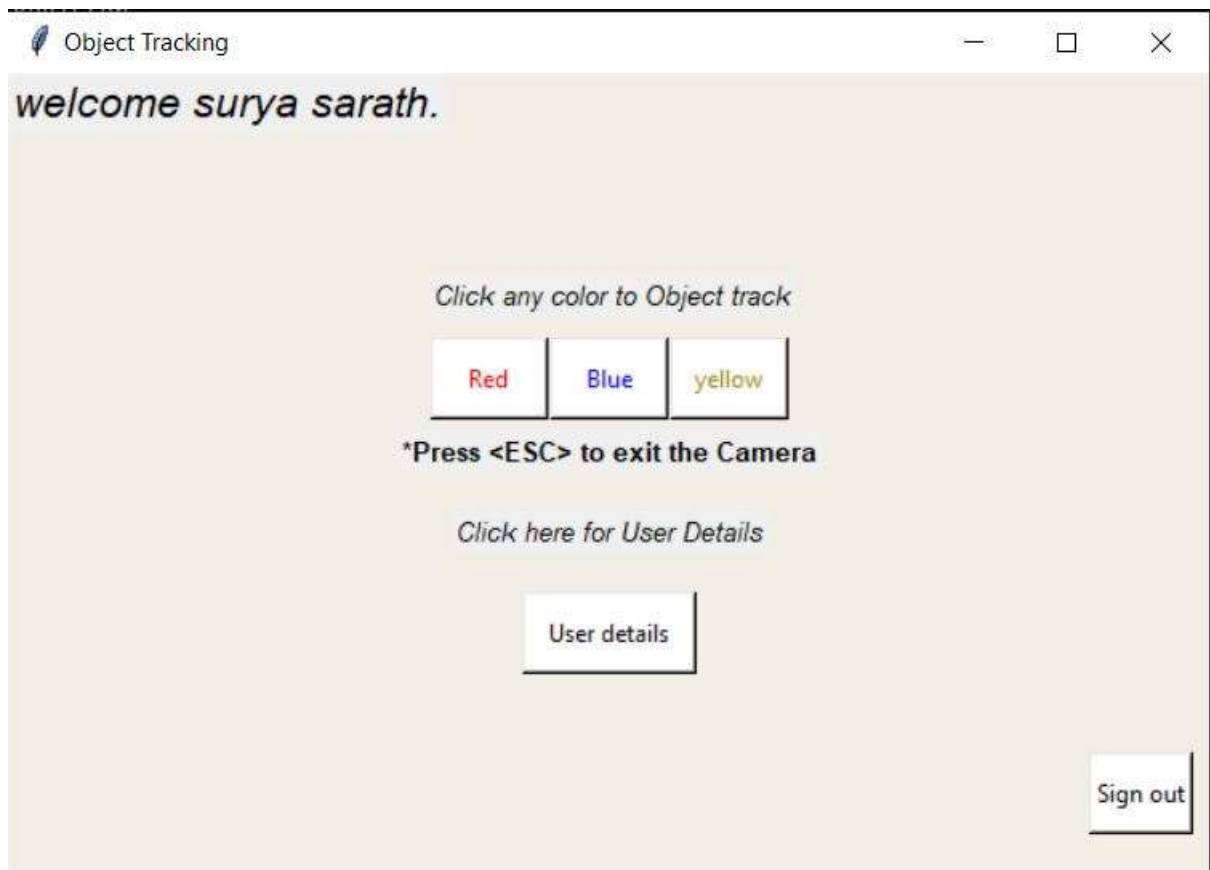


The screenshot shows a web application window titled "Object Tracking". At the top, there are two tabs: "Login" and "Register", with "Register" being the active tab. The main content area contains a registration form with the following fields and labels:

- User Name:
- Full Name:
- Email:
- Phone No:
- Password:
- Re-Password:

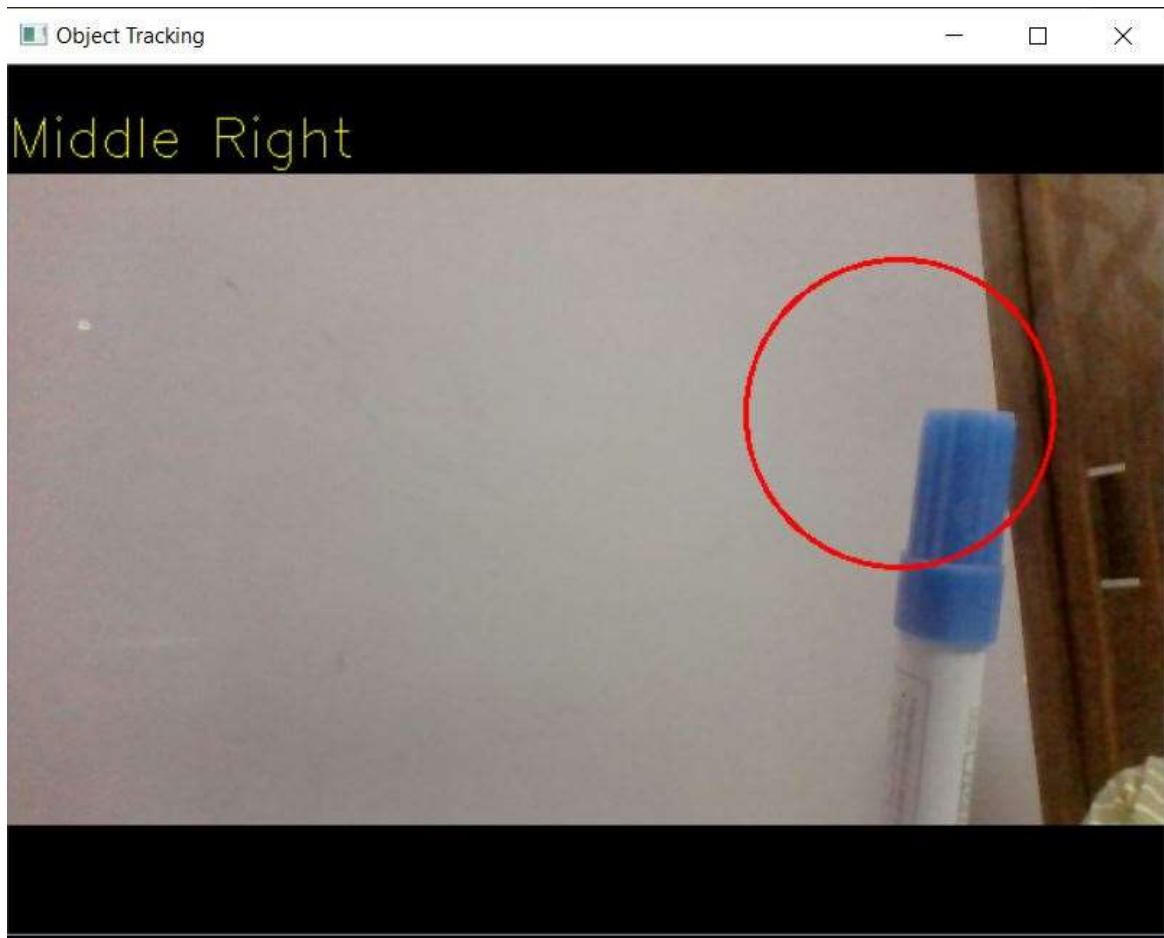
Below the form fields is a blue button labeled "Register".

3.



4.

Object Tracking



5.



CHAPTER 7

SYSTEM TESTING

7.1 INTRODUCTION

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

The software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

7.2 TYPES OF TESTS

7.2.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at the component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.2.2 Integration testing

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event-driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.2.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.2.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links, and integration points.

7.2.5 White Box Testing

White Box Testing is a testing in which the software tester knows the inner workings, structure, and language of the software, or at least its purpose. It is a purpose. It is used to test areas that cannot be reached from a black-box level.

7.2.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure, or language of the module being tested. Black box tests, like most other kinds of tests, must be written from a definitive source document, such as specification or requirements

document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7.2.7 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end-user. It also ensures that the system meets the functional requirements.

7.3 TEST APPROACH

Testing can be done in two ways

1. Bottom-up approach
2. Top-down approach

Bottom-up Approach

Testing can be performed starting from the smallest and lowest level modules and proceeding one at a time. For each module in bottom-up testing, a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower-level modules.

Top-down approach

This type of testing starts from upper-level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper-level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower-level module. 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

The software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

Test Results: All the test cases mentioned above passed successfully.
No defects encountered.

CHAPTER 8

8.1 CONCLUSION

It is related to many real-time applications like vehicle perception, video surveillance, and so on. In order to overcome the issue of detection, tracking related to object movement, and appearance. Most of the algorithms focus on the tracking algorithm to smoothen the video sequence. On the other hand, few methods use the prior available information about object shape, color, and texture, and so on.

8.2 REFERENCES

- 1.<https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking/>
- 2.<https://towardsdatascience.com/automatic-vision-object-tracking-347af1cc8a3b>
- 3.www.python.org