

# A PROJECT REPORT ON

## *Face recognition Attendance system*

Submitted to:

Bankura Sammilani College, Bankura

Affiliated to:

Bankura University, Bankura

Submitted by:

1. Anup Mandal	UID:21023115018
2. Biplab Pal	UID:21023115012
3. Bristi Dey	UID:21023115002
4. Pintu Karmakar	UID:21023115019
5. Pradip Bauri	UID:21023115028
6. Sougata Das Modak	UID:21023115027
7. Suman Mal	UID:21023115039
8. Susanta Malgope	UID:21023115009

Guide Details:

Internal:

1.Mr. Sanjoy sen

External:

1. \_\_\_\_\_

# INDEX

Sr. No	Contents	Page No:
1.	Preface	3
2.	Acknowledgment	4
3.	Project Profile	5
4.	Introduction	6
5.	Feasibility Study	7
6.	System Profile	13
7.	Tools & Technology	13
	i.    Software Requirement	14
	ii.   Hardware Requirement	14
8.	System Design	15
	i.    Data Flow Diagram	
9.	Overview	17
10.	Scope of our System	18
11.	Modules	19
12.	Roles and Responsibility	20
13.	Future Implementation	21
14.	Form Layout	22
15.	Code	26
16.	Bibliography	53

# PREFACE

In today's rapidly advancing technological landscape, the integration of biometric identification systems has become increasingly prevalent across various sectors, including security, healthcare, finance, and social media. Among these systems, face recognition technology stands out for its non-invasive nature and high accuracy, enabling seamless and secure user authentication and identification processes.

We have done the project “FACE RECOGNITION ATTENDANCE SYSTEM”. This documentation aims to provide a comprehensive guide to the face recognition system, offering detailed insights into its architecture, functionalities, and implementation strategies.

# ACKNOWLEDGEMENT

We, the students of Computer Science, Bankura Sammilani College, Bankura, declare that the work entitled 'FACE RECOGNITION BASED ATTENDANCE SYSTEM' has been successfully completed under the guidance of Prof. Sanjoy Sen, Computer Science Department, Bankura Sammilani College, Bankura. We would like to express our sincere thanks to him/her for his/her special guidance throughout the project.

We would like to express our heartfelt thanks to Prof. Tapas Ghosh (H.O.D.) for their constant support and encouragement during this phase and for give us valuable suggestions time to time.

Last but not least we are thankful to our family members and friends for their silent co-operation and passion, and to all others known and unknown person who helped us in this duration.

# PROJECT PROFILE

## **System information**

**System definition:** FACE RECOGNITION ATTENDANCE SYSTEM

**Type of the application:** Desktop Application

**Time duration:** April 2024 – June 2024

**Internal system guide:**

**Submitted by:**

Anup Mandal(21023115018)

Biplab Pal(21023115012)

Bristi Dey(21023115002)

Pintu Karmakar(21023115019)

Pradip Bauri(21023115028)

Sougata Das Modak(21023115027)

Suman Mal(21023115039)

Susanta Malgope(21023115009)

**Submitted to:** Department of computer Science,  
Bankura Sammilani College, Bankura

# INTRODUCTION

Face recognition attendance systems use facial technology to identify and verify a person by comparing their facial features with stored images in a database. When an individual stands in front of a camera, the system captures their image and analyzes unique facial characteristics such as the distance between the eyes, the shape of the cheekbones, and the contour of the lips, jaw, and chin. This data is then matched with pre-recorded data to confirm identity. Here we have used the haar cascade and LBPH algorithm to implement the face recognition.

# FEASIBILITY STUDY

## **Feasibility Study in Face Recognition Attendance Systems**

### **1. Introduction:**

- Maintaining accurate attendance records is essential in various domains, including educational institutions, workplaces, and events.
- Traditional methods are time-consuming and prone to errors.
- Face recognition technology offers an automated and efficient solution to track attendance.

### **2. Project Overview:**

- The feasibility study focuses on creating a real-time face recognition attendance system.
- The system aims to replace manual attendance processes with an automated, reliable, and secure method.

### 3 .Key Components :

- **Hardware:** The feasibility study explores the integration of low-cost hardware with face recognition technology.
- **Software:** The system relies on machine learning algorithms (implemented in Python) for face recognition.
- **Camera:** The input image can be captured using a laptop camera.

### 4 .Methodology :

- **Face Detection:** The camera locates and detects faces within the captured image.
- **Feature Extraction:** Facial features (such as eyes, nose, and mouth) are analyzed to create a unique representation of each face.
- **Data Conversion:** The face information is transformed into digital data (vectors or descriptors).
- **Matching Process:** The system compares the data from the captured face with stored data of known individuals.
- **Attendance Record:** If a match is found, the person's identity is revealed, and their attendance is recorded.



## 5.Challenges and Considerations:

- **Lighting and Pose Variations:** The system should work under different lighting conditions and angles.
- **Privacy and Ethical Concerns:** Balancing security with privacy rights is crucial.
- **Accuracy:** Achieving high accuracy while minimizing false positives and negatives.

## 6.Applications:

- **Education:** Automated attendance in schools, colleges, and universities.
- **Workplaces:** Employee attendance tracking.
- **Events and Conferences:** Streamlined registration and attendance management.

## 7.References:

- A mini project report titled “FACE RECOGNITION ATTENDANCE SYSTEM” provides insights into the implementation details and challenges faced during the project.

## ❖ **Technical Feasibility:**

- Availability of technology and infrastructure
- Compatibility with existing infrastructure
- Algorithm selection and performance
- Database management
- Live video processing
- Privacy and security
- Testing and validations
- Resource constraints

The technical needs of the system may include:

### **Front-end selection:**

1. It must have a graphical user interface that assists employees that are not from IT background.
2. Scalability and extensibility.
3. Flexibility.
4. According to the organization requirement and the culture.
5. Platform independent.
6. Easy to debug and maintain.
7. Event driven programming facility.

### **Back-end Selection:**

1. Multiple user support.
2. Efficient data handling.
3. Face recognition algorithm.
4. Attendance module.
5. Excel sheet integration.
6. Communication with front-end.
7. Operating System compatible.

### ❖ **Economical feasibility:**

#### 1. Cost Estimation:

- Development cost
- Implementation cost
- Maintenance cost

#### 2. Benefits:

- Accuracy
- Efficiency
- Security
- Automated Reporting

#### 3. Cost-Benefit Analysis:

#### 4. Return on Investment:

#### 5. Considerations:

- Scalability
- Integration
- User Acceptance

### ❖ **Operational Feasibility:**

- User acceptance
- Integration with existing process
- Scalability
- Resource availability
- Support and maintenance

Operational feasibility involves assessing whether the face recognition attendance system can smoothly integrate into the educational institution's daily operations.

❖ **Schedule feasibility:**

# SYSTEM PROFILE

- OVERVIEW:
- SCOPE OF SYSTEM:
- MODULES:
- MODULES AFTER LOGIN BY USER:
- MODULES AFTER LOGIN BY ADMINISTRATOR:

## TOOLS AND TECHNOLOGY

- ❑ **Programming Languages:** Python
- ❑ **Libraries and Frameworks:** OpenCV, tkinter, tkcalendar, numPy
- ❑ **Database Systems:** SQLite3
- ❑ **Operating Systems:** Windows

# SOFTWARE AND HARDWARE REQUIREMENTS

## 1. Hardware components:

- **Camera:** High-resolution cameras are required to capture clear images of faces.
- **Computer/Server:** To process the images and run the face recognition algorithms.
- **Storage:** Database to store images and attendance records.

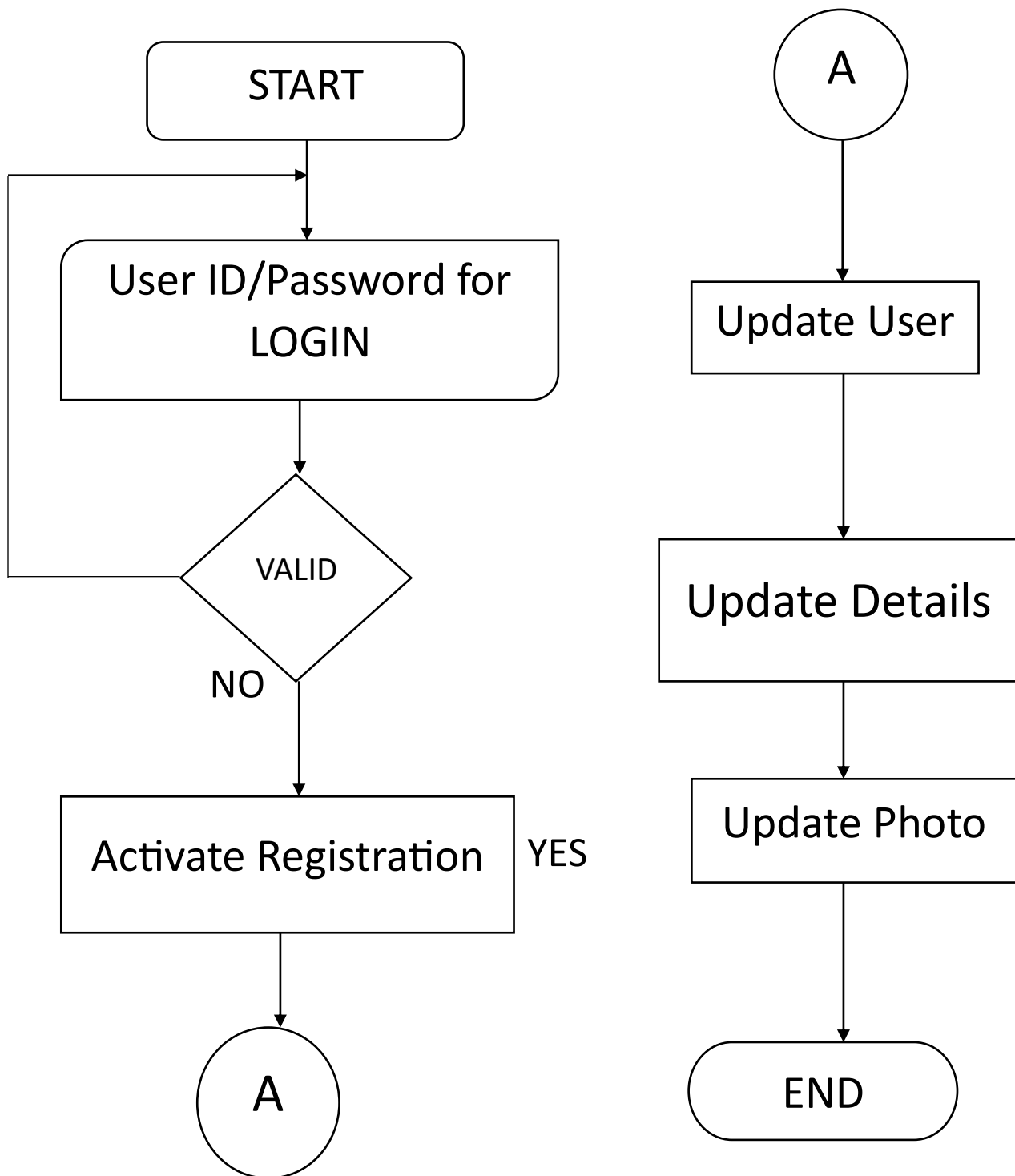
## 2. Software Components:

- **Face Detection Module:** Identifies and locates faces in the captured images.
- **Face Recognition Module:** Matches detected faces against a pre-registered database to identify individuals.
- **Database Management System:** Stores facial data, personal details, and attendance logs.

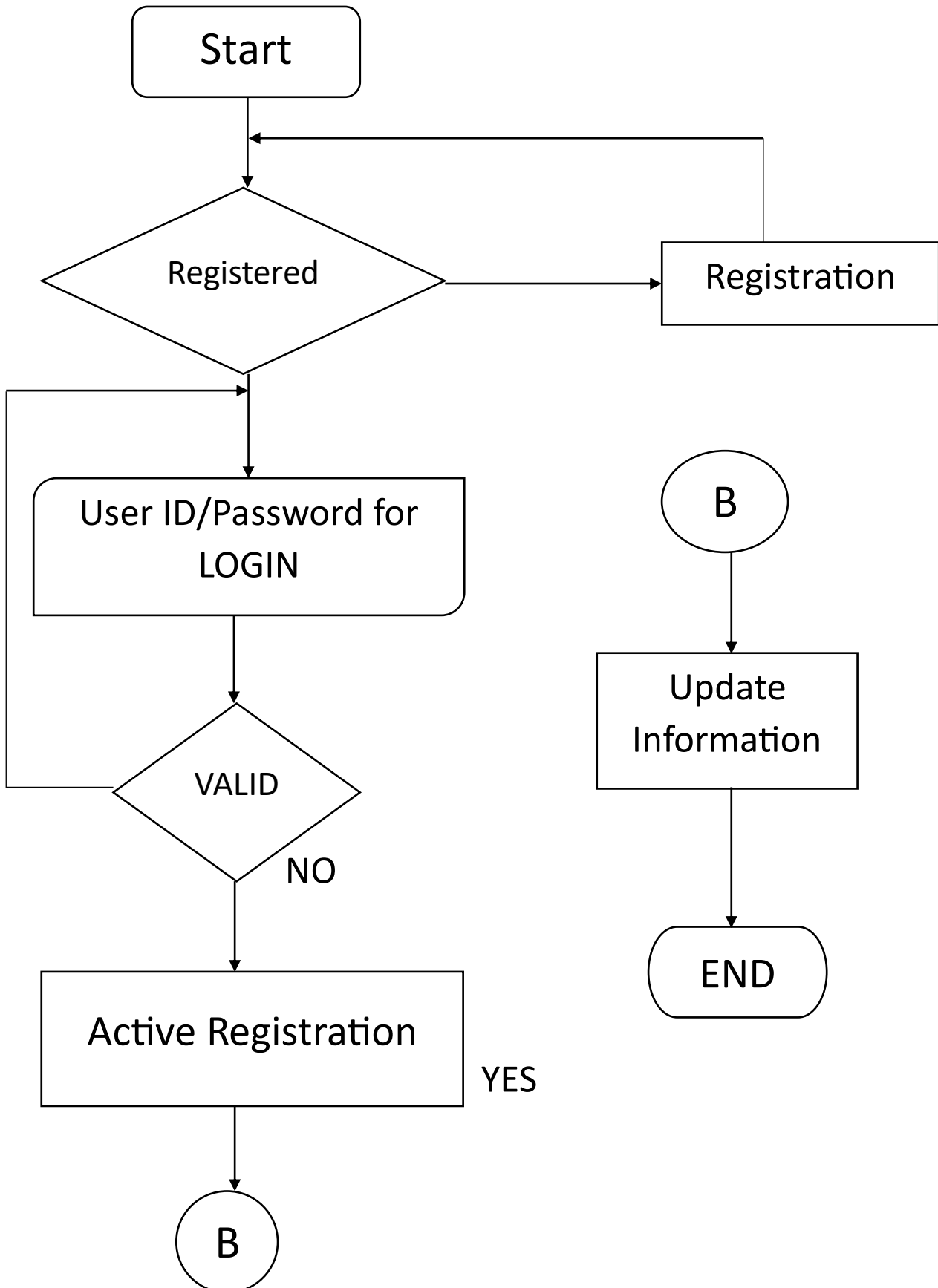
**User Interface:** For administration and monitoring, including functionalities for adding/removing users, viewing attendance reports, and system settings.

# SYSTEM FLOW DIAGRAM

## SYSTEM FLOW FOR ADMIN



## SYSTEM FLOW FOR ADMIN/CLIENT





# OVERVIEW

A face recognition attendance system is an automated solution that leverages facial recognition technology to record the presence and attendance of individuals, typically in settings such as workplaces, educational institutions, and events

# SCOPE OF SYSTEM

The scope of face recognition attendance systems is broad and multifaceted, impacting various sectors and offering numerous advantages. Here are the key aspects and potential applications:

1. It can be highly useful in Educational Institutions for student attendance, examination security & campus security.
2. For employee attendance, access control & payroll integration, it is used in Corporate and Workplaces.
3. In Healthcare Facilities, it can be utilized for staff attendance & patient monitoring.
4. It can be applied in Event & Conferences to keep track on participants.
5. It can be very helpful for Security & Law Enforcement for surveillance and border control(Verifying identities at border crossings and immigration points).
6. In Government & Public Sector, it is used for workforce management & public safety.
7. To manage passengers and to monitor driver and staffs, it can be helpful in Transportation.

# MODULES

Creating a face recognition attendance system involves several modules, each with distinct functionalities. Here are the key modules typically involved in such a project:

1. User Interface Module: Frontend development, dashboard. User enrolment.
2. Database Management Module: Database design, Database connectivity.
3. Image Acquisition Module: Camera integration, Image preprocessing
4. Face Detection & recognition Module: Using Haar cascade algorithm for detection & LBPH algorithm for recognition.
5. Security & Authentication Module: Data encryption, Access control
6. Logging & Monitoring Module: Activity logs like auditing & troubleshooting, System monitoring.

# ROLES AND RESPONSIBILITY

**1.Administrator:** The administrator oversees the overall management, security, and maintenance of the system, ensuring smooth operation and compliance with regulations.

**2.Authorized User:** Authorized users interact with the system to mark their attendance, verify their records, and report any issues.

**3.Unauthorized User:** Unauthorized users are restricted from accessing the system, with security measures in place to prevent and respond to unauthorized attempts.

# **FUTURE IMPLEMENTATION**

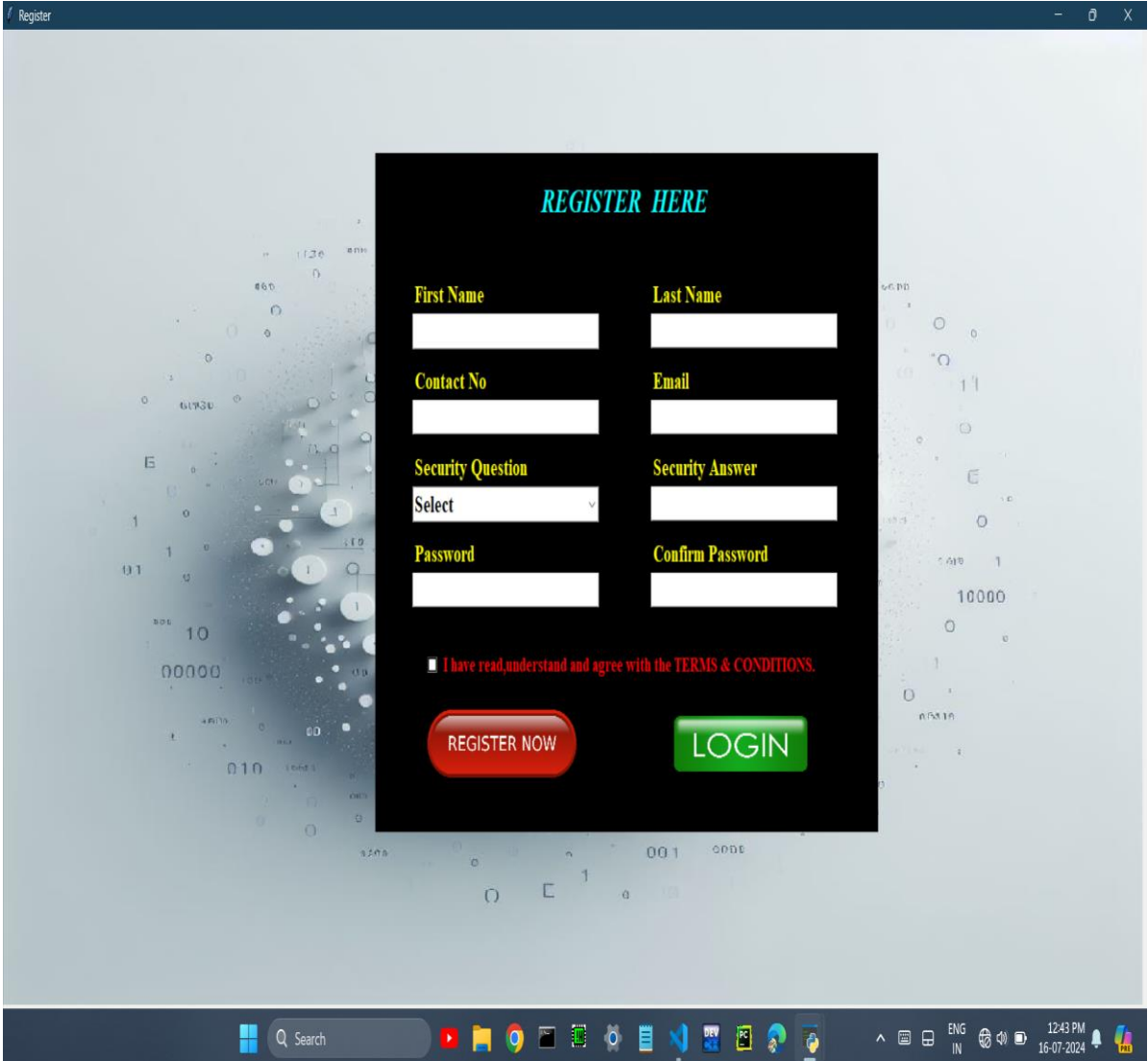
We are hereby confessing that, it is a mini project and it is not completed successfully. We will work more on it make it significant and useful. There are some future implementations we've thought

- Improved accuracy and speed
- Enhanced security and privacy
- Integration and other systems
- Scalability and cloud deployment
- User experience and accessibility
- Ethical and regulatory compliance
- Real-time analytics and reporting
- User feedback and continuous improvement

We will mainly focus on enhancing its accuracy, security and user experience.

# FORM LAYOUT

## REGISTRATION PAGE



The screenshot displays a web browser window with a dark blue title bar labeled "Register". The main content area features a registration form with a black background and white text. The form is titled "REGISTER HERE" in red. It contains several input fields for user registration, including First Name, Last Name, Contact No, Email, Security Question, Security Answer, Password, and Confirm Password. A checkbox for terms and conditions is present, followed by "REGISTER NOW" and "LOGIN" buttons. The browser's taskbar at the bottom shows various application icons and the system clock indicating 12:43 PM on 16-07-2024.

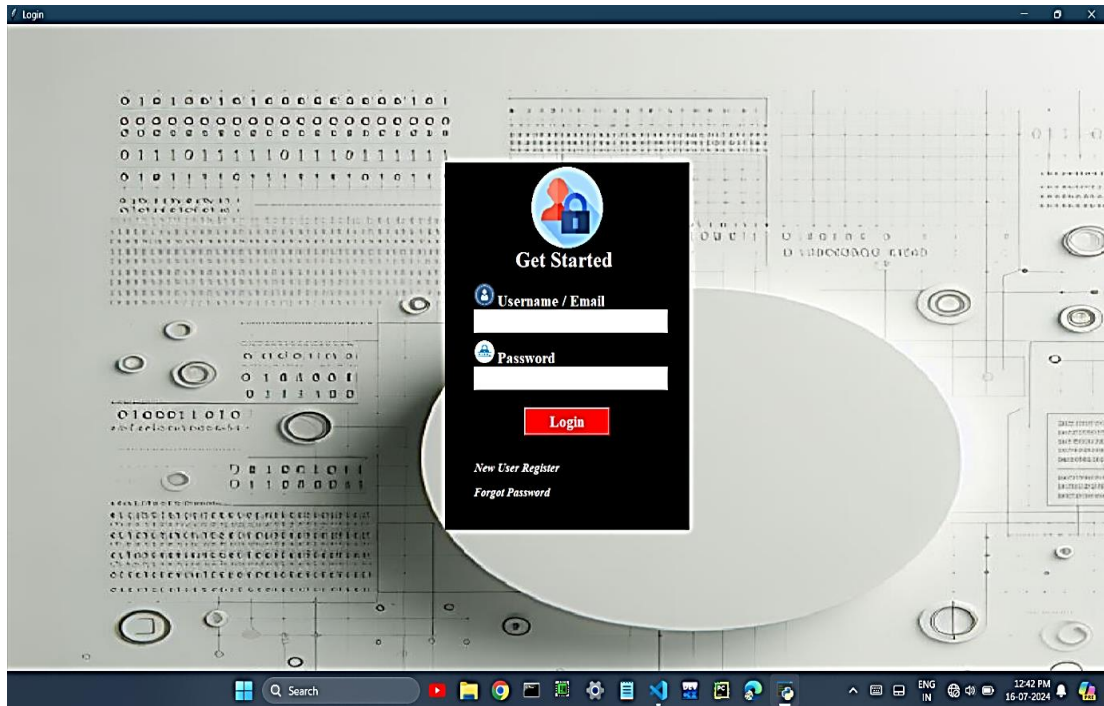
**REGISTER HERE**

<b>First Name</b>	<b>Last Name</b>
<input type="text"/>	<input type="text"/>
<b>Contact No</b>	<b>Email</b>
<input type="text"/>	<input type="text"/>
<b>Security Question</b>	<b>Security Answer</b>
Select <input type="text"/>	<input type="text"/>
<b>Password</b>	<b>Confirm Password</b>
<input type="text"/>	<input type="text"/>

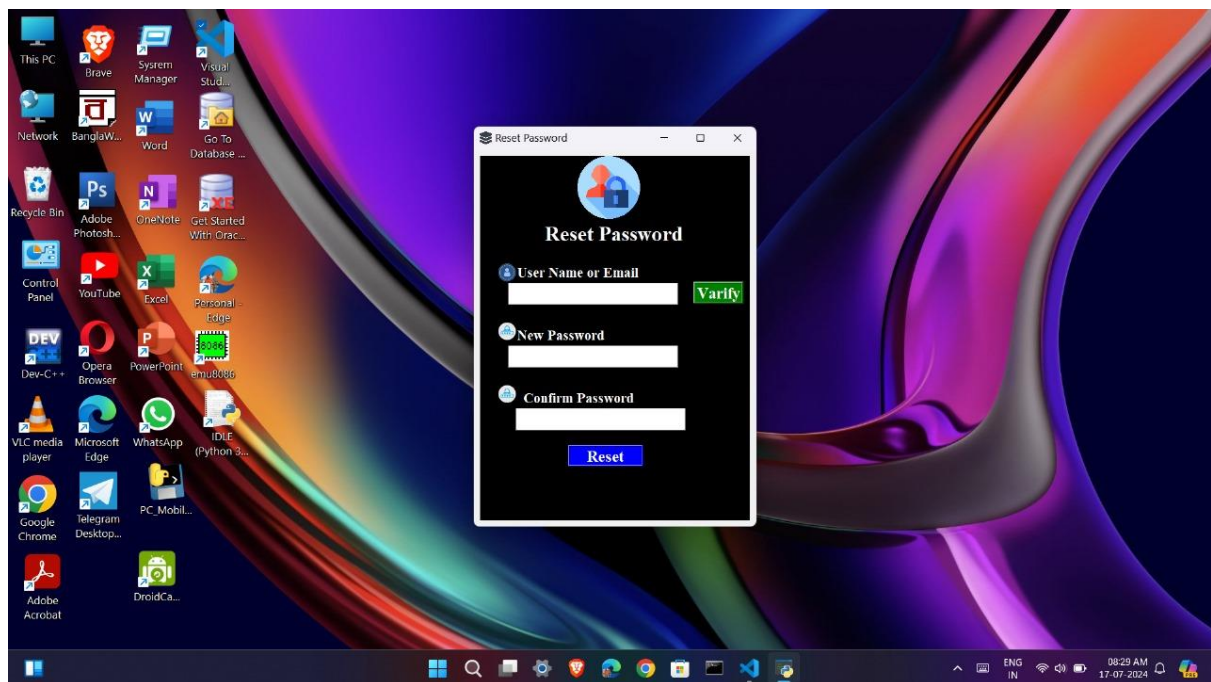
☐ I have read, understand and agree with the **TERMS & CONDITIONS.**

**REGISTER NOW** **LOGIN**

# LOGIN PAGE

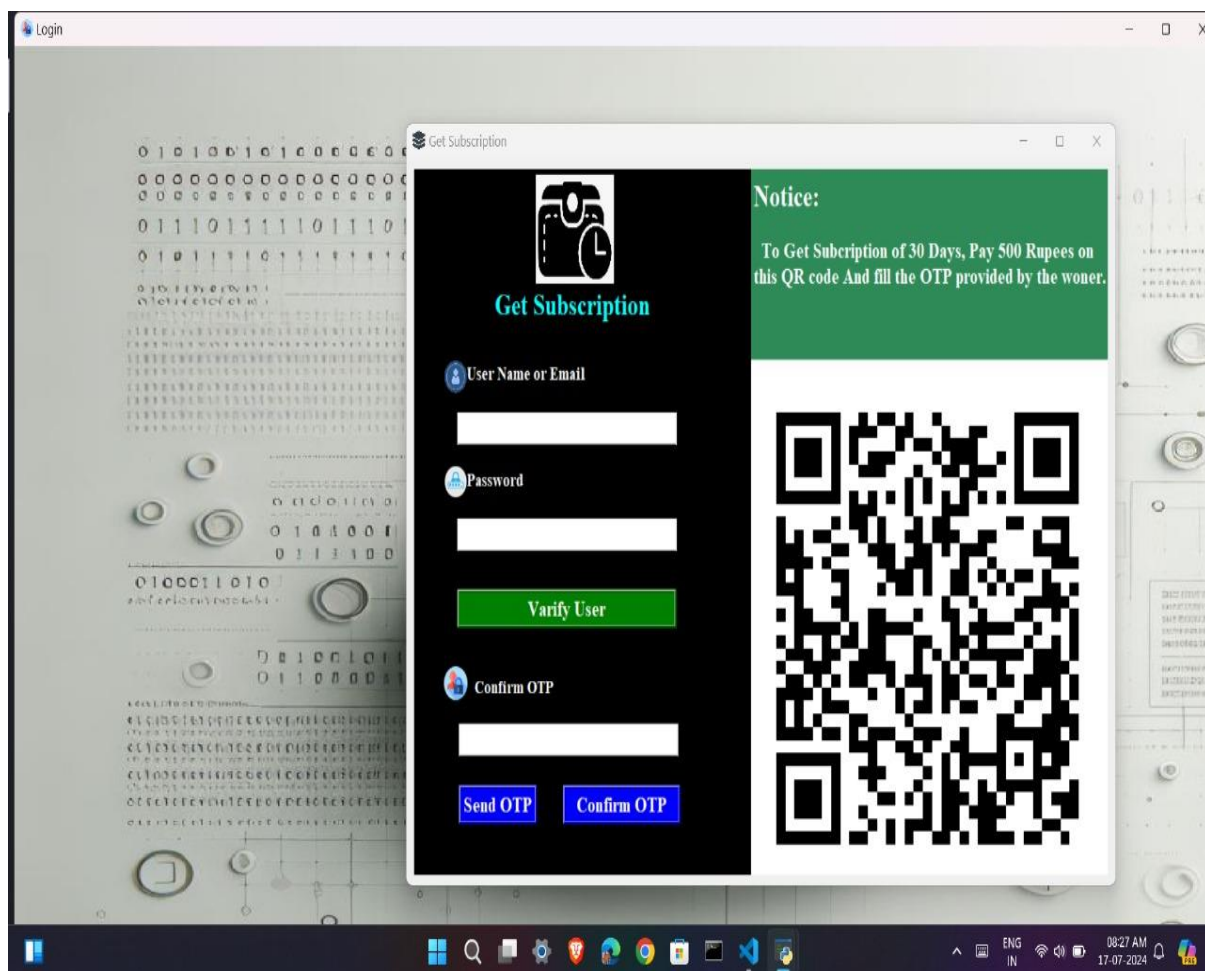


# RESET PASSWORD



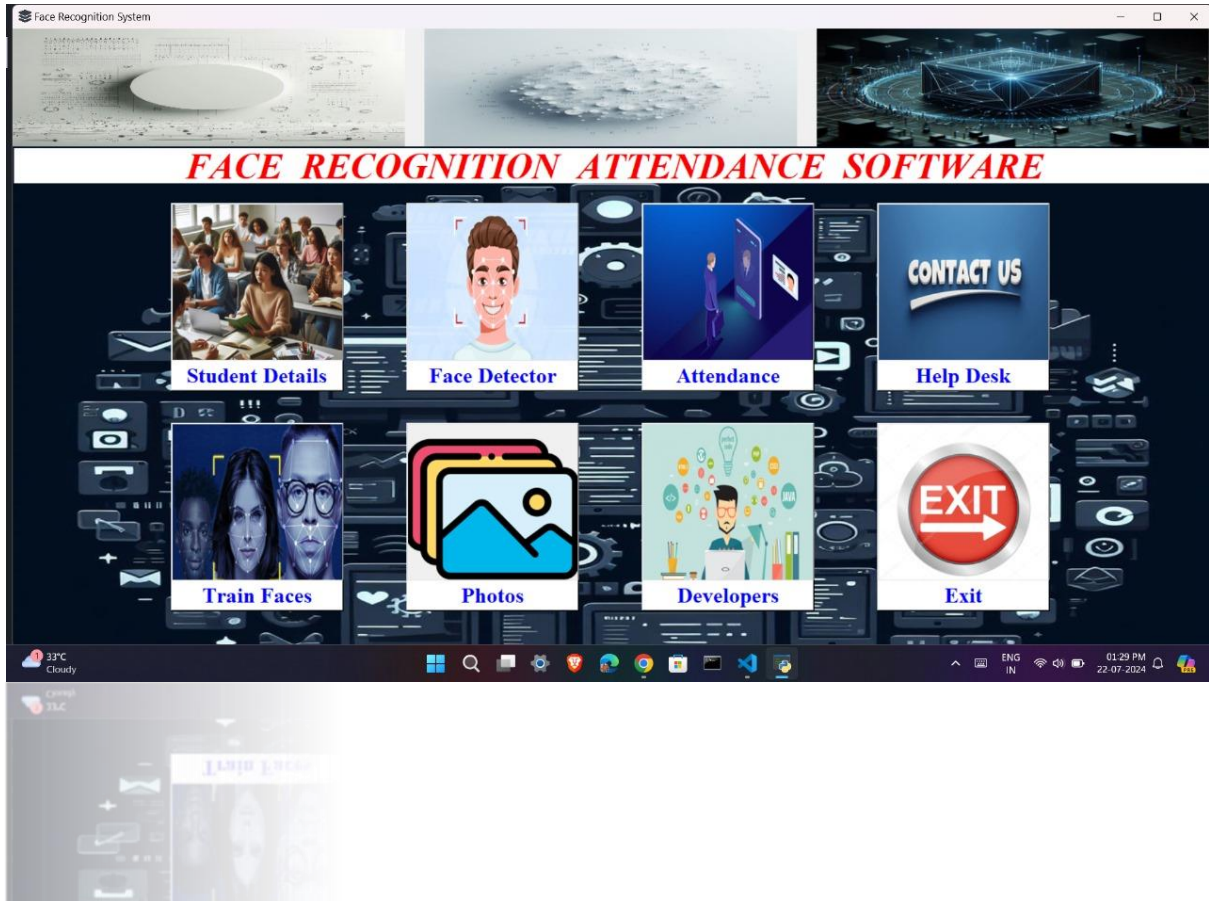


# GET SUBSCRIPTION





# HOME PAGE



# CODE

## ❖ Login.py:

```
from tkinter import*
from tkinter import ttk
from PIL import Image,ImageTk
from tkinter import messagebox
import sqlite3
import getmac
import uuidsss
from time import strftime
import time
import re
import email_validator
from datetime import datetime,timedelta
from main import Face_Recognition_System
from Get_Subscription import Payment

#from Password_Reset import Reset_Password
def main():
    win=Tk()
    app=login_window(win)
    win.mainloop()
class login_window:
    def __init__(self,root):
        self.root=root
        self.root.title("Login")
        self.root.geometry("1550x800+0+0")
        self.textuser=StringVar()
        self.textpass=StringVar()
        #Icon
        icon=PhotoImage(file=r"Images\icon.png")
        root.iconphoto(False,icon)
        #---background image---
        img=Image.open(r"Images\bg.jpeg")
        img=img.resize((1920,1080),resample=0)
        self.photoimg=ImageTk.PhotoImage(img)
        f_lbl=Label(self.root,image=self.photoimg)
        f_lbl.place(x=0,y=0,width=1920,height=1080)
        # =====
        frame=Frame(self.root,bg="black")
        frame.place(x=610,y=170,width=340,height=450)
        img1=Image.open(r"Images\login.png")
        img1=img1.resize((100,100),resample=0)
        self.photoimg1=ImageTk.PhotoImage(img1)
        lblimg1=Label(image=self.photoimg1,bg="black",borderwidth=0)
        lblimg1.place(x=730,y=175,width=100,height=100)
        get_str=Label(frame,text="Get Started",font=("times new
roman",20,"bold"),fg="white",bg="black")
        get_str.place(x=95,y=100)
        # username label
        username=Label(frame,text="Username / Email",font=("times new
roman",15,"bold"),fg="white",bg="black")
        username.place(x=70,y=155)
        self.txtuser=ttk.Entry(frame,textvariable=self.textuser,font=("times new roman",15,"bold"))
        self.txtuser.place(x=42,y=185,width=270)
        # password label
        password=Label(frame,text="Password",font=("times new roman",15,"bold"),fg="white",bg="black")
        password.place(x=70,y=225)
        self.txtpass=ttk.Entry(frame,textvariable=self.textpass,font=("times new roman",15,"bold"))
        self.txtpass.place(x=42,y=260,width=270)
        # ===== Icon image=====
        img2=Image.open(r"Images\user icon.webp")
        img2=img2.resize((30,30),resample=0)
        self.photoimg2=ImageTk.PhotoImage(img2)
        lblimg1=Label(image=self.photoimg2,bg="black",borderwidth=0)
        lblimg1.place(x=650,y=320,width=30,height=30)
```

```

# =====password icon=====
img3=Image.open(r"Images\password.jpg")
img3=img3.resize((30,30),resample=0)
self.photoimg3=ImageTk.PhotoImage(img3)
lblimg1=Label(image=self.photoimg3,bg="black",borderwidth=0)
lblimg1.place(x=650,y=392,width=30,height=30)
# =====login button=====
loginbtn=Button(frame,command=self.login,text="Login",font=("times new
roman",15,"bold"),bd=3,relief=RIDGE,fg="white",bg="red")
loginbtn.place(x=110,y=305,width=120,height=35)
# =====register button=====
registerbtn=Button(frame,command=self.Check_Device,text="New User Register",font=("times new
roman",11,"bold","italic"),borderwidth=0,fg="white",bg="black",activeforeground="white",activebackgrou
nd="black")
registerbtn.place(x=20,y=360,width=160)
# =====forgot password button=====
forgotbtn=Button(frame,command=self.forgotpass_window,text="Forgot Password",font=("times new
roman",11,"bold","italic"),borderwidth=0,fg="white",bg="black",activeforeground="white",activebackgrou
nd="black")
forgotbtn.place(x=13,y=390,width=160)
#=====Function Declaration=====
#=====Check Subscription=====
def Check_Subscription(self):
    try:
        MAC=getmac.get_mac_address()
        conn=sqlite3.connect(r'C:\ProgramData\Windows Secret Data.db')
        cursor=conn.cursor()
        cursor.execute("SELECT Subscription,Till_Date FROM devicemac WHERE MAC=?", (MAC,))
        Subs_Data=cursor.fetchall()
        a=0
        Subs_Detail=[]
        for i in Subs_Data:
            Subs_Detail.append(Subs_Data[a])
            a=a+1
        Subs_Detail = [item for sublist in Subs_Detail for item in sublist]
        Current_Date=datetime.now()
        if datetime.strptime(Subs_Detail[1], '%Y-%m-%d %H:%M:%S.%f') <= Current_Date:
            #messagebox.showwarning("Warning",f"Your {Subs_Detail[0]} Is Expired")
            Subs=messagebox.askyesno("Warning",f"Your {Subs_Detail[0]} Is Expired\n Do you Want To
Buy A subscription?",parent=self.root)
            if Subs >0:
                self.new_window=Toplevel(self.root)
                self.app=Payment(self.new_window)
            else:
                if not Subs:
                    return
            else:
                self.main_window()
        except Exception as es:
            messagebox.showerror("Error",f"Due To:{str(es)}",parent=self.root)
#=====Check Device=====

def Check_Device(self):

    try:
        conn=sqlite3.connect(r'C:\ProgramData\Windows Secret Data.db')
        cursor=conn.cursor()
        cursor.execute('''CREATE TABLE IF NOT EXISTS devicemac(
                                MAC TEXT, Email TEXT PRIMARY KEY,Subscription TEXT NOT
NULL,Register_Date NOT NULL,Till_Date TEXT)''')
        cursor.execute("SELECT MAC FROM devicemac")
        data_mac=cursor.fetchall()
        mac_address=[]
        MAC=getmac.get_mac_address()
        a=0
        for i in data_mac:
            mac_address.append(data_mac[a])
            a=a+1
        mac_address = [item[0] for item in data_mac]
        MAC="NULL"
        if MAC in mac_address:
            messagebox.showinfo("Info","This device Already Rgisterd!!\nDon't Remember
Password?\nClick on Forgot Password")

```

```

        conn.commit()
        conn.close()
    else:
        self.register_window()
except Exception as es:
    messagebox.showerror("Error", f"Due To:{str(es)}", parent=self.root)
#====Login Function====
def login(self):
    if self.txtuser.get()==" " or self.txtpass.get()==" ":
        messagebox.showerror("Error", "All fields are required")
    else:
        try:
            conn=sqlite3.connect('User DataBase.db')
            cursor=conn.cursor()
            cursor.execute("SELECT Password FROM User WHERE Email=?", (self.txtuser.get(),))
            Password_Data=cursor.fetchall()
            Password=[]
            if len(Password_Data)!=0:
                a=0
                for i in Password_Data:
                    Password.append(Password_Data[a])
                    a=a+1
                Password = [item[0] for item in Password]
            else:
                Password.append("None")
            if self.txtpass.get()== Password[0]:
                messagebox.showinfo("Success", "Welcome,you are now logged in", parent=self.root)
                self.textuser.set(""),
                self.textpass.set("")
                self.Check_Subscription()
            else:
                messagebox.showerror("Invalid", "Invalid username or password", parent=self.root)
        except Exception as es:
            messagebox.showerror("Error", f"Due To:{str(es)}", parent=self.root)
#====Forgot Password====

def forgotpass_window(self):
    self.new_window=Toplevel(self.root)
    self.app=Reset_Password(self.new_window)

#====Main Window====
def main_window(self):
    self.new_window=Toplevel(self.root)
    self.app=Face_Recognition_System(self.new_window)
    #====Register====
def register_window(self):
    self.new_window=Toplevel(self.root)
    self.app=register(self.new_window)
class register:
    def __init__(self,root):
        self.root=root
        self.root.title("Register")
        self.root.geometry("1600x900+0+0")
        #Icon
        icon=PhotoImage(file=r"Images\icon.png")
        root.iconphoto(False, icon)
        # variables-----
        self.var_fname=StringVar()
        self.var_lname=StringVar()
        self.var_contact=StringVar()
        self.var_email=StringVar()
        self.var_question=StringVar()
        self.var_answer=StringVar()
        self.var_pass=StringVar()
        self.var_confirmpass=StringVar()
        #---background image---
        img=Image.open(r"Images\bgl.jpg")
        img=img.resize((1530,790),resample=0)
        self.photoimg=ImageTk.PhotoImage(img)
        f_lbl=Label(self.root,image=self.photoimg)
        f_lbl.place(x=0,y=0,width=1530,height=790)
        #-----main frame-----
        frame=Frame(self.root,bg="black")
        frame.place(x=500,y=100,width=675,height=550)

```

```

        register_lbl=Label(frame,text="REGISTER  HERE",font=("times new
roman",20,"bold","italic"),fg="cyan",bg="black")
        register_lbl.place(x=220,y=20)
        #-----label and entry -----
        # first name-----
        fname=Label(frame,text="First Name",font=("times new roman",15,"bold"),bg="black",fg="yellow")
        fname.place(x=50,y=100)
        fname_entry=ttk.Entry(frame,textvariable=self.var_fname,font=("times new roman",15,"bold"))
        fname_entry.place(x=50,y=130,width=250)
        # last name-----
        lname=Label(frame,text="Last Name",font=("times new roman",15,"bold"),bg="black",fg="yellow")
        lname.place(x=370,y=100)
        self.txt_lname=ttk.Entry(frame,textvariable=self.var_lname,font=("times new roman",15))
        self.txt_lname.place(x=370,y=130,width=250)
        # contact-----
        contact=Label(frame,text="Contact No",font=("times new
roman",15,"bold"),bg="black",fg="yellow")
        contact.place(x=50,y=170)
        self.txt_contact=ttk.Entry(frame,textvariable=self.var_contact,font=("times new roman",15))
        self.txt_contact.place(x=50,y=200,width=250)
        # email-----
        email=Label(frame,text="Email",font=("times new roman",15,"bold"),bg="black",fg="yellow")
        email.place(x=370,y=170)
        self.txt_email=ttk.Entry(frame,textvariable=self.var_email,font=("times new roman",15))
        self.txt_email.place(x=370,y=200,width=250)
        # security question-----
        question=Label(frame,text="Security Question",font=("times new
roman",15,"bold"),bg="black",fg="yellow")
        question.place(x=50,y=240)
        self.combo_question=ttk.Combobox(frame,textvariable=self.var_question,font=("times new
roman",15,"bold"),state="readonly")
        self.combo_question["values"]=("Select","Your D.O.B","Your Birth Place","Your Pet Name")
        self.combo_question.place(x=50,y=270,width=250)
        self.combo_question.current(0)
        #security answer-----
        answer=Label(frame,text="Security Answer",font=("times new
roman",15,"bold"),bg="black",fg="yellow")
        answer.place(x=370,y=240)
        self.txt_answer=ttk.Entry(frame,textvariable=self.var_answer,font=("times new roman",15))
        self.txt_answer.place(x=370,y=270,width=250)
        # password-----
        password=Label(frame,text="Password",font=("times new
roman",15,"bold"),bg="black",fg="yellow")
        password.place(x=50,y=310)

        self.password=ttk.Entry(frame,textvariable=self.var_pass,font=("times new roman",15))
        self.password.place(x=50,y=340,width=250)
        # confirm password-----
        confirm=Label(frame,text="Confirm Password",font=("times new
roman",15,"bold"),bg="black",fg="yellow")
        confirm.place(x=370,y=310)
        self.confirm=ttk.Entry(frame,textvariable=self.var_confirmpass,font=("times new roman",15))
        self.confirm.place(x=370,y=340,width=250)
        # check button-----
        self.var_check=IntVar()
        checkbtn=Checkbutton(frame,variable=self.var_check,text="I have read,understand and agree with
the TERMS & CONDITIONS.",font=("times new roman",13,"bold"),bg="black",fg="red",onvalue=1,offvalue=0)
        checkbtn.place(x=65,y=400)
        # register button-----
        img1=Image.open(r"Images\register.jpg")
        img1=img1.resize((200,55),resample=0)
        self.photoimg1=ImageTk.PhotoImage(img1)
        b1=Button(frame,image=self.photoimg1,command=self.register,borderwidth=0,cursor="hand2",font=(
"times new roman",13,"bold"),bg="black",fg="white",activebackground="black")
        b1.place(x=70,y=450,width=200)
        # login button-----
        img2=Image.open(r"Images\log in.png")
        img2=img2.resize((180,45),resample=0)
        self.photoimg2=ImageTk.PhotoImage(img2)
        b1=Button(frame,command=self.return_login,image=self.photoimg2,borderwidth=0,cursor="hand2",fo
nt=("times new roman",13,"bold"),bg="black",fg="white",activebackground="black")
        b1.place(x=390,y=455,width=200)
        #=====function declaration=====
        #=====Rgister=====

```

```

def register(self):
    try:
        Email=self.var_email.get()
        if self.var_fname.get()==" or self.var_lname.get()==" or self.var_email.get()==" or
self.var_contact.get()==" or self.var_question.get()=="Select" or self.var_answer.get()=="":
            messagebox.showerror("Error","All fields are required",parent=self.root)
        elif len(self.var_pass.get())<6:
            messagebox.showerror("Error","Password Should Be More than 6
Character",parent=self.root)
        elif self.var_pass.get()!=self.var_confirmpass.get():
            messagebox.showerror("Error","Password and Confirm Password must be
same",parent=self.root)
        elif self.var_check.get()==0:
            messagebox.showerror("Error","Please agree our Terms and Conditions",parent=self.root)
        #phone=int(self.var_contact.get())
        elif re.search(r"\d",self.var_fname.get()) or re.search(r"^[a-zA-Z0-
9]",self.var_fname.get()) or re.search(r"\d",self.var_lname.get()) or re.search(r"^[a-zA-Z0-
9]",self.var_lname.get()):
            messagebox.showerror("Error","Name Should Not Contain Any Numeric or Special
Symbol!",parent=self.root)
        #email=self.var_email.get()
        #elif '@' not in Email:
        #    messagebox.showerror("Error","@ symbol is missing in Email Address",parent=self.root)
        #elif '@' in Email and Email[Email.find('@')+1:]:
        #    messagebox.showerror("Error","Invalid Email Address",parent=self.root)
        else:
            try:
                email_validator.validate_email(Email)
            except:
                messagebox.showerror("Error","Invalid Email Address",parent=self.root)
            try:
                phone=int(self.var_contact.get())
                if phone<=1000000000 or phone>=9999999999:
                    messagebox.showerror("Error","Mobile No Should Be 10 Digit or A Valid
Number",parent=self.root)
                else:
                    try:
                        conn=sqlite3.connect('User DataBase.db')
                        cursor=conn.cursor()
                        cursor.execute('''CREATE TABLE IF NOT EXISTS User(
                            First_Name TEXT,Second_Name TEXT,ContactNo INTEGER NOT
                            NULL,
                            Email TEXT ,Security_Ans TEXT,Password TEXT,
                            PRIMARY KEY(Email))''')
                        conn_Dev=sqlite3.connect(r'C:\ProgramData\Windows Secret Data.db')

                        MAC=getmac.get_mac_address()

                        Email=self.var_email.get()
                        Register_Date=datetime.now()
                        Till_Date=Register_Date + timedelta(minutes=5)
                        cursor.execute("INSERT INTO User
(First_Name,Second_Name,ContactNo,Email,Security_Ans,Password) VALUES(?,?,?,?,?,?,?)", (
                            self.var_fname.get(),
                            self.var_lname.get(),
                            self.var_contact.get(),
                            self.var_email.get(),
                            self.var_answer.get(),
                            self.var_confirmpass.get()
                        ))
                        conn.commit()
                        conn.close()
                        cursor_Dev=conn_Dev.cursor()
                        cursor_Dev.execute("INSERT INTO devicemac
(MAC,Email,Subscription,Register_Date,Till_Date)
VALUES(?,?,?,?,?)", (MAC,Email,"Trial",str(Register_Date),str(Till_Date)))
                        conn_Dev.commit()
                        conn_Dev.close()
                        messagebox.showinfo("Success","You are successfully
registered",parent=self.root)
                        self.return_login()
                    except Exception as es:
                        conn.close()
                        messagebox.showerror("Error",f"Due To :{str(es)}",parent=self.root)

```

```

        except:
            messagebox.showerror("Error", "Enter A Valid Mobile Number", parent=self.root)
    except Exception as es:
        messagebox.showerror("Error", f"Due To :{str(es)}", parent=self.root)
#====Log In====
def return_login(self):
    self.root.destroy()
class Reset_Password:
    def __init__(self, root):
        self.root = root
        self.root.title("Reset Password")
        self.root.geometry("400x500+650+180")
        #self.root.geometry("1550x800+0+0")
        #Icon
        icon = PhotoImage(file=r"Images\icon.png")
        root.iconphoto(False, icon)
        self.status = "No"
        self.user = "None"
        self.var_pass = ""
        self.var_confirmpass = ""
        f_lbl = Label(self.root, height=600, width=160)
        f_lbl.place(x=10, y=10, width=600, height=160)
        frame = Frame(self.root, bg="black")
        frame.place(x=10, y=10, width=380, height=480)
        img1 = Image.open(r"Images\login.png")
        img1 = img1.resize((100, 100), resample=0)
        self.photoimg1 = ImageTk.PhotoImage(img1)
        lblimg1 = Label(frame, image=self.photoimg1, bg="black", borderwidth=0)
        lblimg1.place(x=155, y=0, width=100, height=100)
        get_str = Label(frame, text="Reset Password", font=("times new
roman", 20, "bold"), fg="white", bg="black")
        get_str.place(x=100, y=102)
        #====User Name Varify====
        user = Label(frame, text=" User Name or Email", font=("times new
roman", 13, "bold"), fg="white", bg="black")
        user.place(x=50, y=150)
        self.Email = ttk.Entry(frame, font=("times new roman", 15, "bold"))
        self.Email.place(x=50, y=182, width=230)
        #Icon
        img2 = Image.open(r"images\user icon.webp")
        img2 = img2.resize((30, 30), resample=0)
        self.photoimg2 = ImageTk.PhotoImage(img2)
        lblimg1 = Label(frame, image=self.photoimg2, bg="black", borderwidth=0)
        lblimg1.place(x=28, y=150, width=30, height=30)

        #====New Password====
        password = Label(frame, text=" New Password", font=("times new
roman", 13, "bold"), fg="white", bg="black")
        password.place(x=50, y=230)
        self.var_pass = ttk.Entry(frame, font=("times new roman", 15, "bold"))
        self.var_pass.place(x=50, y=260, width=230)
        #Icon
        img3 = Image.open(r"Images\password.jpg")
        img3 = img3.resize((30, 30), Image.LANCZOS)
        self.photoimg3 = ImageTk.PhotoImage(img3)
        self.photoimConfpass = ImageTk.PhotoImage(img3)
        lblimg1 = Label(frame, image=self.photoimg3, bg="black", borderwidth=0)
        lblimg1.place(x=28, y=230, width=30, height=30)
        #====Confirm Password====
        confpassword = Label(frame, text="Confirm Password", font=("times new
roman", 13, "bold"), fg="white", bg="black")
        confpassword.place(x=60, y=310)
        self.var_confirmpass = ttk.Entry(frame, font=("times new roman", 15, "bold"))
        self.var_confirmpass.place(x=50, y=340, width=230)
        #Icon
        lblimg1 = Label(frame, image=self.photoimConfpass, bg="black", borderwidth=0)
        lblimg1.place(x=28, y=310, width=30, height=30)
        # =====Reset button=====
        loginbtn = Button(frame, text="Reset", command=self.Reset, font=("times new
roman", 15, "bold"), bd=3, relief=RIDGE, fg="white", bg="blue")
        loginbtn.place(x=120, y=400, width=120, height=35)
        #Varify Button
        varifybtn = Button(frame, text="Varify", command=self.varify, font=("times new
roman", 15, "bold"), bd=3, relief=RIDGE, fg="white", bg="green")

```



```

        varifybtn.place(x=290,y=180,width=80,height=35)
#=====Function Declaration=====
#=====Verify=====
def varify(self):
    if self.Email.get()=="":
        messagebox.showerror("Error","Please enter the Email to reset Password")
    else:
        conn=sqlite3.connect(r'User DataBase.db')
        cursor=conn.cursor()
        cursor.execute("SELECT Email FROM User")
        data_user=cursor.fetchall()
        conn.commit()
        conn.close()
        Email_user=[]
        if len(data_user)!=0:
            a=0
            for i in data_user:
                Email_user.append(data_user[a])
                a=a+1
            Email_user = [item[0] for item in Email_user]
        if self.Email.get() not in Email_user:
            messagebox.showerror("Error","No Such Email or User Registered!",parent=self.root)
        else:
            self.status="Yes"
            self.user=self.Email.get()
            messagebox.showinfo("Info","User Varified",parent=self.root)

#=====Reset=====
def Reset(self):
    if self.status=="No":
        messagebox.showwarning("Warning","Verify The Email First",parent=self.root)
    elif self.var_pass.get()=="" or self.var_confirmpass.get()=="":
        messagebox.showerror("Error","Both fields are required",parent=self.root)
    elif len(self.var_pass.get())<6:
        messagebox.showerror("Error","Password Should Be More than 6",parent=self.root)
    elif self.var_pass.get()!=self.var_confirmpass.get():
        messagebox.showerror("Error","Password and Confirm Password must be
same",parent=self.root)
    else:
        conn=sqlite3.connect(r'User DataBase.db')
        cursor=conn.cursor()
        cursor.execute("UPDATE User SET Password=? WHERE
Email=?",(self.var_confirmpass.get(),self.user))
        conn.commit()
        conn.close()
        messagebox.showinfo("Info","Password Successfully Reset\nYou Can Login",parent=self.root)
        self.root.destroy()

'''if __name__=="__main__":
    root=Tk()
    app=Reset_Password(root)
    #login=LW()
    #print(login.Email)
    root.mainloop()'''
if __name__=="__main__":
    main()

```

## ❖ Main.py

```

from tkinter import*
from tkinter import ttk
from tkinter import messagebox
import tkinter as tk
from PIL import Image,ImageTk
from Student import Student
import Test_Face_Recongnizer
import Test_Train_Data
from help import Help
from developer import Developer
from Attendance import Attendance
#from Attendance import Attendance
class Face_Recognition_System:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1530x790+0+0")

```



```

self.root.title("Face Recognition System")
#Icon
icon=PhotoImage(file=r"Images\icon.png")
root.iconphoto(False,icon)
#---first image---
img1=Image.open(r"Images\bg.jpg")
img1=img1.resize((500,150),resample=0)
self.photoimg1=ImageTk.PhotoImage(img1)
f_lbl=Label(self.root,image=self.photoimg1)
f_lbl.place(x=0,y=0,width=500,height=150)
#----second image----
img2=Image.open(r"Images\bg1.jpg")
img2=img2.resize((500,150))
self.photoimg2=ImageTk.PhotoImage(img2)
f_lbl=Label(self.root,image=self.photoimg2)
f_lbl.place(x=500,y=0,width=550,height=150)
#----third image----
img3=Image.open(r"Images\bg3.jpg")
img3=img3.resize((500,150))
self.photoimg3=ImageTk.PhotoImage(img3)
f_lbl=Label(self.root,image=self.photoimg3)
f_lbl.place(x=1000,y=0,width=550,height=150)
#----background image----
img4=Image.open(r"Images\bg5.jpg")
img4=img4.resize((1530,790))
self.photoimg4=ImageTk.PhotoImage(img4)
bg_img=Label(self.root,image=self.photoimg4)
bg_img.place(x=0,y=150,width=1530,height=790)
#----title----
title_lbl=Label(bg_img,text="FACE RECOGNITION ATTENDANCE SOFTWARE",font=("times new
roman", 35, "bold", "italic"),bg="white",fg="red")
title_lbl.place(x=0,y=0,width=1530,height=45)
#----student button----
img5=Image.open(r"Images\student-details.jpg")
img5=img5.resize((220,220))
self.photoimg5=ImageTk.PhotoImage(img5)
b1.place(x=200,y=70,width=220,height=220)
b1=Button(bg_img,image=self.photoimg5,command=self.Student_details,cursor="hand2")
b1_1=Button(bg_img,text="Student
Details",command=self.Student_details,cursor="hand2",font=("times new
roman", 20, "bold"),bg="white",fg="blue")
b1_1.place(x=200,y=270,width=220,height=40)
#----detect face button----
img6=Image.open(r"Images\face1.jpg")
img6=img6.resize((220,220))
self.photoimg6=ImageTk.PhotoImage(img6)
b1=Button(bg_img,image=self.photoimg6,command=self.Face_detector,cursor="hand2")
b1.place(x=500,y=70,width=220,height=220)
b1_1=Button(bg_img,text="Face Detector",command=self.Face_detector,cursor="hand2",font=("times
new roman", 20, "bold"),bg="white",fg="blue")
b1_1.place(x=500,y=270,width=220,height=40)
#----attendance button----
img7=Image.open(r"Images\attendance.jpg")
img7=img7.resize((220,220))
self.photoimg7=ImageTk.PhotoImage(img7)
b1=Button(bg_img,image=self.photoimg7,command=self.Attendance,cursor="hand2")
b1.place(x=800,y=70,width=220,height=220)
b1_1=Button(bg_img,text="Attendance",command=self.Attendance,cursor="hand2",font=("times new
roman", 20, "bold"),bg="white",fg="blue")
b1_1.place(x=800,y=270,width=220,height=40)
#----help desk button----
img8=Image.open(r"Images\help.jpg")
img8=img8.resize((220,220))
self.photoimg8=ImageTk.PhotoImage(img8)
b1=Button(bg_img,image=self.photoimg8,command=self.Help,cursor="hand2")
b1.place(x=1100,y=70,width=220,height=220)
b1_1=Button(bg_img,text="Help Desk",command=self.Help,cursor="hand2",font=("times new
roman", 20, "bold"),bg="white",fg="blue")
b1_1.place(x=1100,y=270,width=220,height=40)
#----train face button----
img9=Image.open(r"Images\train.jpg")
img9=img9.resize((220,220))
self.photoimg9=ImageTk.PhotoImage(img9)
b1=Button(bg_img,image=self.photoimg9,command=self.Train_Data,cursor="hand2")

```

```

        b1.place(x=200,y=350,width=220,height=220)
        b1_1=Button(bg_img,text="Train Faces",command=self.Train_Data,cursor="hand2",font=("times new
roman", 20, "bold"),bg="white",fg="blue")
        b1_1.place(x=200,y=550,width=220,height=40)
        #----photos button----
        img10=Image.open(r"Images\photo.jpg")
        img10=img10.resize((220,220))
        self.photoimg10=ImageTk.PhotoImage(img10)
        b1=Button(bg_img,image=self.photoimg10,cursor="hand2")
        b1.place(x=500,y=350,width=220,height=220)

        b1_1=Button(bg_img,text="Photos", cursor="hand2",font=("times new
roman", 20, "bold"),bg="white",fg="blue")
        b1_1.place(x=500,y=550,width=220,height=40)
        #-----Developer button-----
        Developer=Image.open(r"Images\Developer.jpg")
        Developer=Developer.resize((220,220))
        self.DeveloperIMG=ImageTk.PhotoImage(Developer)
        Developer_Button=Button(bg_img,image=self.DeveloperIMG,command=self.Developer,cursor="hand2")
        Developer_Button.place(x=800,y=350,width=220,height=220)
        Developer_Button_Text=Button(bg_img,text="Developers", command=self.Developer,cursor="hand2",fo
nt=("times new roman", 20, "bold"),bg="white",fg="blue")
        Developer_Button_Text.place(x=800,y=550,width=220,height=40)
        #-----exit button-----
        img12=Image.open(r"Images\exit.jpg")
        img12=img12.resize((220,220))
        self.photoimg12=ImageTk.PhotoImage(img12)
        Exit_Button=Button(bg_img,image=self.photoimg12,command=self.Exit,cursor="hand2")
        Exit_Button.place(x=1100,y=350,width=220,height=220)
        Exit_Button_Text=Button(bg_img,text="Exit", command=self.Exit,cursor="hand2",font=("times new
roman", 20, "bold"),bg="white",fg="blue")
        Exit_Button_Text.place(x=1100,y=550,width=220,height=40)
        #=====Function Buttons=====
        def Student_details(self):
            self.new_window=Toplevel(self.root)
            self.app=Student(self.new_window)
        def Face_detector(self):
            messagebox.showinfo("Success","Face Detector Is Being Ready,Please Wait",parent=self.root)
            Test_Face_Recongnizer.face_recognition()
            #self.new_window=Toplevel(self.root)
            #self.app=Face_Recognition(self.new_window)
        def Train_Data(self):
            messagebox.showinfo("Info","Please Wait to Process the Data\nWhen Traing Will be
completed you will get a message",parent=self.root)
            #self.new_window=Toplevel(self.root)
            #self.app=Train_Data(self.new_window)
            Test_Train_Data.train_classifier()
            messagebox.showinfo("Success","Training data set completed",parent=self.root)
        def Help(self):
            self.new_window=Toplevel(self.root)
            self.app=Help(self.new_window)
        def Developer(self):
            self.new_window=Toplevel(self.root)
            self.app=Developer(self.new_window)
        def Attendance(self):
            self.new_window=Toplevel(self.root)
            self.app=Attendance(self.new_window)
        def Exit(self):
            self.root.destroy()
if __name__=="__main__":
    root=Tk()
    obj=Face_Recognition_System(root)
    root.mainloop()

```

## ❖ [Getsubscription.py](#)

```

from tkinter import*
from tkinter import ttk
from PIL import Image,ImageTk
from tkinter import messagebox
import sqlite3
import uuid

```

```

import getmac
from time import strftime
import time
from datetime import datetime,timedelta
from twilio.rest import Client
import random

class Payment():
    def __init__(self,root):
        self.root=root
        self.root.title("Get Subscription")
        self.root.geometry("905x650+500+100")
        #Icon
        icon=PhotoImage(file=r"Images\icon.png")
        root.iconphoto(False,icon)
        #=====Variables=====
        self.status="No"
        self.user="None"
        self.OTP=""
        self.ConfOTP=""
        self.OTPstatus="No"
        f_lbl=Label(self.root,height=650,width=180)
        f_lbl.place(x=10,y=10,width=650,height=180)
        frame=Frame(self.root,bg="SeaGreen")
        frame.place(x=10,y=10,width=885,height=630)
        frameBlack=Frame(self.root,bg="black")
        frameBlack.place(x=10,y=10,width=430,height=630)
        img1=Image.open(r"Images\Subscription.png")
        img1=img1.resize((100,100),resample=0)
        self.photoimg1=ImageTk.PhotoImage(img1)
        lblimg1=Label(frameBlack,image=self.photoimg1,bg="black",borderwidth=0)
        lblimg1.place(x=155,y=5,width=100,height=100)
        get_str=Label(frameBlack,text="Get Subscription",font=("times new
roman",20,"bold"),fg="Cyan",bg="black")
        get_str.place(x=100,y=102)
        #=====User Name Varify=====
        user=Label(frameBlack,text=" User Name or Email",font=("times new
roman",13,"bold"),fg="white",bg="black")
        user.place(x=60,y=170)
        self.Email=ttk.Entry(frameBlack,font=("times new roman",15,"bold"))
        self.Email.place(x=55,y=217,width=280)
        #Icon
        img2=Image.open(r"images\user icon.webp")
        img2=img2.resize((30,30),resample=0)
        self.photoimg2=ImageTk.PhotoImage(img2)
        lblimg1=Label(frameBlack,image=self.photoimg2,bg="black",borderwidth=0)
        lblimg1.place(x=38,y=170,width=30,height=30)
        #===== Password=====
        password=Label(frameBlack,text=" Password",font=("times new
roman",13,"bold"),fg="white",bg="black")
        password.place(x=60,y=265)
        self.var_pass=ttk.Entry(frameBlack,font=("times new roman",15,"bold"))
        self.var_pass.place(x=55,y=312,width=280)
        #Icon
        img3=Image.open(r"Images\password.jpg")
        img3=img3.resize((30,30),Image.LANCZOS)
        self.photoimg3=ImageTk.PhotoImage(img3)
        self.photoimConfpass=ImageTk.PhotoImage(img3)
        lblimg1=Label(frameBlack,image=self.photoimg3,bg="black",borderwidth=0)
        lblimg1.place(x=38,y=264,width=30,height=30)
        #=====OTP=====
        OTP=Label(frameBlack,text=" Confirm OTP",font=("times new
roman",13,"bold"),fg="white",bg="black")
        OTP.place(x=70,y=450)
        self.var_OTP=ttk.Entry(frameBlack,font=("times new roman",15,"bold"))
        self.var_OTP.place(x=57,y=495,width=280)
        #Icon
        img4=Image.open(r"Images\login.png")
        img4=img4.resize((30,30),Image.LANCZOS)
        self.photoimOTP=ImageTk.PhotoImage(img4)
        lblimg1=Label(frameBlack,image=self.photoimOTP,bg="black",borderwidth=0)
        lblimg1.place(x=38,y=444,width=30,height=30)
        #=====QR Code=====
        QRCode=Image.open(r"Images\QR Code.png")

```

```

QRCode=QRCode.resize((480,480),Image.LANCZOS)
self.QRCode=ImageTk.PhotoImage(QRCode)
lblimg1=Label(frame,image=self.QRCode,bg="black",borderwidth=0)
lblimg1.place(x=415,y=170)
#====Disclaimer Text====

Disclaimer=Label(frame,text="Notice:",font=("times new
roman",20,"bold"),fg="White",bg="seaGreen")
Disclaimer.place(x=430,y=5)
Disclaimertxt=Label(frame,text="\nTo Get Subcription of 30 Days, Pay 500 Rupees on \nthis QR
code And fill the OTP provided by the woner.",font=("times new
roman",15,"bold"),fg="White",bg="seaGreen")
Disclaimertxt.place(x=430,y=35,bordermode=INSIDE)
# =====Send OTP button=====
SendOTP=Button(frameBlack,text="Send OTP",command=self.sendOTP,font=("times new
roman",15,"bold"),bd=3,relief=RIDGE,fg="white",bg="blue")
SendOTP.place(x=57,y=550,width=100,height=35)
ConfirmOTP=Button(frameBlack,text="Confirm OTP",command=self.verifyOTP,font=("times new
roman",15,"bold"),bd=3,relief=RIDGE,fg="white",bg="blue")
ConfirmOTP.place(x=190,y=550,width=150,height=35)
#Verify User
verifynbtn=Button(frameBlack,text="Verify User",command=self.verify,font=("times new
roman",15,"bold"),bd=3,relief=RIDGE,fg="white",bg="green")
verifynbtn.place(x=55,y=375,width=280,height=35)
#=====Function Declaration=====
def verify(self):
    if self.Email.get()==" " or self.var_pass.get()==" ":
        messagebox.showerror("Error","User Name and Password required",parent=self.root)
    else:
        try:
            conn=sqlite3.connect('User DataBase.db')
            cursor=conn.cursor()
            cursor.execute("SELECT Password FROM User WHERE Email=?", (self.Email.get(),))
            Password_Data=cursor.fetchall()
            Password=[]
            if len(Password_Data)!=0:
                a=0
                for i in Password_Data:
                    Password.append(Password_Data[a])
                    a=a+1
                Password = [item[0] for item in Password]
            else:
                Password.append("None")

            if self.var_pass.get()== Password[0]:
                self.status="Yes"
                self.user=self.Email.get()
                messagebox.showinfo("Success","User Varified Successfully",parent=self.root)
            else:
                messagebox.showerror("Invalid","Invalid username or password",parent=self.root)
        except Exception as es:
            messagebox.showerror("Error",f"Due To:{str(es)}",parent=self.root)
#=====Send OTP=====
def sendOTP(self):
    if self.status=="No":
        messagebox.showwarning("warning","First Verify the User",parent=self.root)
    else:
        #elif (os.system("ping -c 1 8.8.8.8 >/dev/null 2>&1")):
        try:
            account_sid = 'ACf0c89dd277f45fd7fe456fcdb47135a5'
            auth_token = 'f13ef59fcc2b0b61cf19a45f3d286cba'
            self.OTP=random.randint(100000,999999)
            msg=f"OTP for Buying Subscription of Automatic Attendance System.\nOTP is {self.OTP}"
            client = Client(account_sid, auth_token)
            message = client.messages.create(
                body=msg,
                from_='+15078703448', # Replace with your Twilio number
                to='+919564280663' # Replace with your phone number
            )
            messagebox.showinfo("Success","OTP Sent Successfully",parent=self.root)
            self.OTPstatus="Yes"
        except Exception as es:
            messagebox.showerror("Error",f"Due To:{str(es)}",parent=self.root)
def verifyOTP(self):
    if self.OTPstatus=="No":

```

```

        messagebox.showwarning("warning", "First send the OTP", parent=self.root)
    elif self.var_OTP.get()=="":
        messagebox.showwarning("warning", "Fill the OTP", parent=self.root)
    elif int(self.OTP)!=int(self.var_OTP.get()):
        print(self.var_OTP.get())
        print(self.OTP)
        messagebox.showwarning("warning", "OTP Does not match", parent=self.root)
    else:
        try:
            MAC=getmac.get_mac_address()
            conn=sqlite3.connect(r'C:\ProgramData\Windows Secret Data.db')
            cursor=conn.cursor()
            cursor.execute("SELECT Till_Date FROM devicemac WHERE MAC=?", (MAC,))
            data_TillDate=cursor.fetchall()
            TillDate=[]
            a=0
            for i in data_TillDate:
                TillDate.append(data_TillDate[a])
                a=a+1
            TillDate = [item[0] for item in TillDate]
            Till_Date=datetime.strptime(TillDate[0], '%Y-%m-%d %H:%M:%S.%f')
            Current_Date=datetime.now()
            if Till_Date>Current_Date:
                Till_Date=TillDate[0] +timedelta(minutes=1.5)
            else:
                Till_Date=Current_Date +timedelta(minutes=5)
            cursor.execute("UPDATE devicemac SET Subscription=?,Till_Date=? WHERE
MAC=?", ("Subscription", str(Till_Date), MAC))
            conn.commit()
            conn.close()
            messagebox.showinfo("Success", "Subscription Buy Successfully", parent=self.root)
            self.root.destroy()
        except Exception as es:
            messagebox.showerror("Error", f"Due To :{str(es)}", parent=self.root)

if __name__=="__main__":
    root=Tk()
    app=Payment(root)
    root.mainloop()

```

## ❖ Student.py

```

from tkinter import*
from tkinter import ttk
import tkinter as tk
from PIL import Image, ImageTk
from tkcalendar import DateEntry
from tkinter import messagebox
import string
import sqlite3
import email_validator
import cv2
from datetime import datetime, timedelta
from dateutil.relativedelta import relativedelta
import re
class Student:
    def __init__(self, root):
        self.root=root
        self.root.geometry("1530x790+0+0")
        self.root.title("Face Recognition System")
        icon=PhotoImage(file=r"Images\icon.png")
        root.iconphoto(False, icon)
        #=====Variables=====
        self.var_dep=StringVar()
        self.var_course=StringVar()
        self.var_stream=StringVar()
        self.var_semester=StringVar()
        self.var_div=StringVar()
        self.var_std_id=StringVar()
        self.var_std_name=StringVar()
        self.var_roll=StringVar()
        self.var_gender=StringVar()

```

```

self.var_dob=StringVar()
self.var_email=StringVar()
self.var_phone=StringVar()
self.var_address=StringVar()
self.var_teacher=StringVar()
self.var_radio=StringVar()
self.student_Roll=[]
self.radio=StringVar()

#---first image---
img1=Image.open(r"Images\bg.jpg")
img1=img1.resize((500,150),resample=0)
self.photoimg1=ImageTk.PhotoImage(img1)
f_lbl=Label(self.root,image=self.photoimg1)
f_lbl.place(x=0,y=0,width=500,height=150)
#----second image----
img2=Image.open(r"Images\bg1.jpg")
img2=img2.resize((500,150))
self.photoimg2=ImageTk.PhotoImage(img2)
f_lbl=Label(self.root,image=self.photoimg2)
f_lbl.place(x=500,y=0,width=550,height=150)
#-----third image-----
img3=Image.open(r"Images\bg3.jpg")
img3=img3.resize((500,150))
self.photoimg3=ImageTk.PhotoImage(img3)
f_lbl=Label(self.root,image=self.photoimg3)
f_lbl.place(x=1000,y=0,width=550,height=150)
#-----background image-----
img4=Image.open(r"Images\bg5.jpg")
img4=img4.resize((1530,790))
self.photoimg4=ImageTk.PhotoImage(img4)
bg_img=Label(self.root,image=self.photoimg4)
bg_img.place(x=0,y=150,width=1530,height=790)
#-----title-----
title_lbl=Label(bg_img,text="STUDENT MANAGEMENT",font=("times new
roman",35,"bold","italic"),bg="white",fg="red")
title_lbl.place(x=0,y=0,width=1530,height=45)
#----Frame-----
main_frame=Frame(bg_img,bd=2,bg="white")
main_frame.place(x=20,y=55,width=1480,height=600)
#left-label-frame
Left_frame=LabelFrame(main_frame,bd=2,bg="white",relief=RIDGE,text="Student Details",
font=("times new roman",12,"bold"))
Left_frame.place(x=10,y=10,width=730,height=580)
left_img=Image.open(r"Images\bg1.jpg")
left_img=img2.resize((500,150))
self.photo_left_img=ImageTk.PhotoImage(left_img)
f_lbl=Label(Left_frame,image=self.photo_left_img)
f_lbl.place(x=5,y=0,width=720,height=130)
#current course
current_course_frame=LabelFrame(Left_frame,bd=2,bg="white",relief=RIDGE,text="Current
course",font=("times new roman",12,"bold"))
current_course_frame.place(x=5,y=135,width=720,height=150)
#Stream
stream_label=Label(current_course_frame,text="Stream:",font=("times new roman",13,
"bold"),bg="white")
stream_label.grid(row=0,column=0)
stream_combo=ttk.Combobox(current_course_frame,textvariable=self.var_stream,font=("times new
roman",12,"bold"),state="Read Only")
stream_combo["values"]=("Select Stream","Science","Arts","Commerce")
stream_combo.current(0)
stream_combo.grid(row=0,column=1,padx=5,pady=5,sticky=W)
#Department
dep_label=Label(current_course_frame,text="Department:",font=("times new roman",13,
"bold"),bg="white")
dep_label.grid(row=0,column=2)
dep_combo=ttk.Combobox(current_course_frame,textvariable=self.var_dep,font=("times new roman",
12,"bold"),state="Read Only")
dep_combo["values"]=("Select Department","Select Stream First")
def save_value(event):
    # Get the current value
    current_value = stream_combo.get()
    # Now you can use this value to check the next combobox value
    if current_value=="Science":

```

```

        dep_combo["values"]=("Computer Science","Chemistry","Mathematics","Physics")
    elif current_value=="Arts":
        dep_combo["values"]=("Bengali","Sanskrit","English","Polytical Science","History")
    else:
        dep_combo["values"]=("Economics","Accountancy","Statistics")

    # Bind the function to the combobox
    stream_combo.bind("<<ComboboxSelected>>", save_value)
    dep_combo.grid(row=0,column=3,padx=5,pady=5,sticky=W)
    dep_combo.current(0)
    #===Course Type===
    course_Type_label=Label(current_course_frame, text="Course Type:", font=("times new roman",
13, "bold"),bg="white")
    course_Type_label.grid(row=1,column=0,pady=10,sticky=W)
    course_Type_combo=ttk.Combobox(current_course_frame,textvariable=self.var_course,font=("times
new roman", 12, "bold"),state="Read Only")
    course_Type_combo["values"]=("Select Course","Hons","Program")
    course_Type_combo.current(0)
    course_Type_combo.grid(row=1,column=1,padx=2,pady=10,sticky=W)
    #Semester
    sem_label=Label(current_course_frame, text="Semester:", font=("times new roman", 13,
"bold"),bg="white")
    sem_label.grid(row=1,column=2,pady=10,sticky=W)
    sem_combo=ttk.Combobox(current_course_frame,textvariable=self.var_semester,font=("times new
roman", 12, "bold"),state="Read Only")
    sem_combo["values"]=("Select Semester","1st Sem","2nd Sem","3st Sem","4th Sem","5th Sem","6th
Sem")
    sem_combo.current(0)
    sem_combo.grid(row=1,column=3,pady=10,sticky=W)
    #Class Student information
    Class_Student_frame=LabelFrame(Left_frame, bd=2, bg="white", relief=RIDGE, text="Class Student
information",font=("times new roman",12,"bold"))
    Class_Student_frame.place(x=5,y=250, width=720,height=300)
    #Student ID
    studentID_label=Label(Class_Student_frame, text="StudentID:", font=("times new roman", 13,
"bold"))
    studentID_label.grid(row=0,column=0,padx=10,pady=5,sticky=W)
    studentID_Entry=ttk.Entry(Class_Student_frame,textvariable=self.var_std_id,width=20,font=(
es new roman", 13, "bold"))
    def studentID_check(event):
        if self.var_std_id.get()!="":
            try:
                int(self.var_std_id.get())
            except:
                messagebox.showerror("Error","ID should be integer",parent=self.root)
    studentID_Entry.bind("<KeyRelease>", studentID_check)
    studentID_Entry.grid(row=0,column=1,padx=10,pady=5,sticky=W)
    #Student Name
    studentName_label=Label(Class_Student_frame, text="Student Name:", font=("times new roman",
13, "bold"))
    studentName_label.grid(row=0,column=2,padx=10,pady=5,sticky=W)
    studentName_Entry=ttk.Entry(Class_Student_frame,textvariable=self.var_std_name,width=20,font=(
"times new roman", 13, "bold"))
    def check_StudentName(event):
        if self.var_std_name.get()!=" " and re.search(r"\d",self.var_std_name.get().replace("
","")) or re.search(r"^[^a-zA-Z0-9]",self.var_std_name.get().replace(" ","")):
            messagebox.showerror("Error","Name Should Not Contain Any Numeric or Special
Symbol!",parent=self.root)
    studentName_Entry.bind("<KeyRelease>",check_StudentName)
    studentName_Entry.grid(row=0,column=3,padx=10,pady=5,sticky=W)
    #Class Division
    student_div_label=Label(Class_Student_frame, text="Class Division:", font=("times new roman",
13, "bold"))
    student_div_label.grid(row=1,column=0,padx=10,pady=5,sticky=W)
    student_div_Entry=ttk.Combobox(Class_Student_frame,textvariable=self.var_div,width=20,font=(
imes new roman", 13, "bold"),state="Read Only")
    student_div_Entry["values"]=("Select Division","A","B","C")
    student_div_Entry.current(0)
    student_div_Entry.grid(row=1,column=3,pady=10,sticky=W)
    student_div_Entry.grid(row=1,column=1,padx=10,pady=5,sticky=W)
    #Roll No
    Roll_no_label=Label(Class_Student_frame, text="Roll No:", font=("times new roman", 13,
"bold"))
    Roll_no_label.grid(row=1,column=2,padx=10,pady=5,sticky=W)

```



```

Roll_no_Entry=ttk.Entry(Class_Student_frame,textvariable=self.var_roll,width=20,font=("times
new roman", 13, "bold"))

def studentRoll_check(event):
    if self.var_roll.get()!="":
        try:
            int(self.var_roll.get())
        except:
            messagebox.showerror("Error","Roll No should be integer",parent=self.root)
Roll_no_Entry.bind("<KeyRelease>", studentRoll_check)
Roll_no_Entry.grid(row=1,column=3,padx=10,pady=5,sticky=W)
#Gender
Gender_label=Label(Class_Student_frame, text="Gender:", font=("times new roman", 13, "bold"))
Gender_label.grid(row=2,column=0,padx=10,pady=5,sticky=W)
Gender_Entry=ttk.Combobox(Class_Student_frame,textvariable=self.var_gender,font=("times new
roman", 12, "bold"),state="Read Only")
Gender_Entry["values"]=("Select Gender","Male","Female","Other")
Gender_Entry.current(0)
Gender_Entry.grid(row=2,column=1,padx=10,pady=5,sticky=W)
#DOB
DOB_label=Label(Class_Student_frame, text="DOB(MM.DD.YY):", font=("times new roman", 13,
"bold"))
DOB_label.grid(row=2,column=2,padx=10,pady=5,sticky=W)
DOB_Entry=DateEntry(Class_Student_frame,textvariable=self.var_dob,width=18,font=("times new
roman", 13, "bold"))
def check_date(event):
    selected_date = datetime.strptime(self.var_dob.get(), "%m/%d/%y")
    current_date = datetime.now()
    if selected_date.date() > current_date.date():
        messagebox.showerror("Error","You Can't Set Upcoming Date",parent=self.root)
    elif (current_date.date() < selected_date.date()+relativedelta(years=15)):
        messagebox.showerror("Error","Age Should be Greater than 15",parent=self.root)
DOB_Entry.bind("<<DateEntrySelected>>", check_date)

DOB_Entry.grid(row=2,column=3,padx=10,pady=5,sticky=W)
#Email
Email_label=Label(Class_Student_frame, text="Email:", font=("times new roman", 13, "bold"))
Email_label.grid(row=3,column=0,padx=10,pady=5,sticky=W)
Email_Entry=ttk.Entry(Class_Student_frame,textvariable=self.var_email,width=20,font=("times
new roman", 13, "bold"))
Email_Entry.grid(row=3,column=1,padx=10,pady=5,sticky=W)
#Phone No
Phone_label=Label(Class_Student_frame, text="Phone No:", font=("times new roman", 13, "bold"))
Phone_label.grid(row=3,column=2,padx=10,pady=5,sticky=W)
Phone_Entry=ttk.Entry(Class_Student_frame,textvariable=self.var_phone,width=20,font=("times
new roman", 13, "bold"))
def Phone_check(event):
    if self.var_phone.get()!="":
        try:
            int(self.var_phone.get())
        except:
            messagebox.showerror("Error","Phone No should be integer",parent=self.root)
Phone_Entry.bind("<KeyRelease>", Phone_check)
Phone_Entry.grid(row=3,column=3,padx=10,pady=5,sticky=W)
#Address
Address_label=Label(Class_Student_frame, text="Address:", font=("times new roman", 13,
"bold"))
Address_label.grid(row=4,column=0,padx=10,pady=5,sticky=W)
Address_Entry=Entry(Class_Student_frame,textvariable=self.var_address,width=20,font=("times
new roman", 13, "bold"))
Address_Entry.grid(row=4,column=1,padx=10,pady=5,sticky=W)
#Teacher Name
Teacher_label=Label(Class_Student_frame, text="Teacher Name:", font=("times new roman", 13,
"bold"))
Teacher_label.grid(row=4,column=2,padx=10,pady=5,sticky=W)
Teacher_Entry=ttk.Entry(Class_Student_frame,textvariable=self.var_teacher,width=20,font=("time
s new roman", 13, "bold"))
def check_TeacherName(event):
    if self.var_teacher.get()!=" " and re.search(r"\d",self.var_teacher.get().replace(" ",""))
or re.search(r"^[^a-zA-Z0-9]",self.var_teacher.get().replace(" ","")):
        messagebox.showerror("Error","Teacher Name Should Not Contain Any Numeric or Special
Symbol!",parent=self.root)
Teacher_Entry.bind("<KeyRelease>", check_TeacherName)
Teacher_Entry.grid(row=4,column=3,padx=10,pady=5,sticky=W)

```



```

        #Radio Buttons
        radiobutton1=ttk.Radiobutton(Class_Student_frame,variable=self.var_radio,text="Take Photo
Sample",value="Yes")
        radiobutton1.grid(row=6,column=0)

        radiobutton2=ttk.Radiobutton(Class_Student_frame,variable=self.var_radio,text="No Photo
Sample",value="No")
        radiobutton2.grid(row=6,column=1)
        #Buttons Frame
        btn_frame=Frame(Class_Student_frame,bd=2,relief=RIDGE,bg="white")
        btn_frame.place(x=0,y=215,width=715,height=34)
        #save
        save_btn=Button(btn_frame,text="Save",command=self.add_data,width=17,font=("times new roman",
13, "bold"),bg="blue",fg="white")
        save_btn.grid(row=0,column=0)
        #Update
        update_btn=Button(btn_frame,text="Update",command=self.update_data,width=17,font=("times new
roman", 13, "bold"),bg="blue",fg="white")
        update_btn.grid(row=0,column=1)
        #Delete
        delete_btn=Button(btn_frame,text="Delete",command=self.delete_data,width=17,font=("times new
roman", 13, "bold"),bg="blue",fg="white")
        delete_btn.grid(row=0,column=2)
        #Reset
        Reset_btn=Button(btn_frame,text="Reset",command=self.reset_data,width=17,font=("times new
roman", 13, "bold"),bg="blue",fg="white")
        Reset_btn.grid(row=0,column=3)
        #2nd Button Frame
        btn_frame1=Frame(Class_Student_frame,bd=2,relief=RIDGE,bg="white")
        btn_frame1.place(x=0,y=245,width=715,height=35)
        #Take Photo
        take_photo_btn=Button(btn_frame1,text="Take
Photo",command=self.generate_dataset,width=34,font=("times new roman", 13,
"bold"),bg="blue",fg="white")
        take_photo_btn.grid(row=0,column=0)
        #Update Photo
        update_photo_btn=Button(btn_frame1,text="Update Photo",width=35,font=("times new roman", 13,
"bold"),bg="blue",fg="white")
        update_photo_btn.grid(row=0,column=1)
        #Right label frame
        Right_frame=LabelFrame(main_frame,bd=2,bg="white", relief=RIDGE, text="Student
Details",font=("times new roman",12,"bold"))
        Right_frame.place(x=750,y=10, width=720, height=588)
        right_img=Image.open(r"Images\bg1.jpg")
        right_img=img2.resize((500,150))
        self.photo_right_img=ImageTk.PhotoImage(right_img)
        f_lbl=Label(Right_frame,image=self.photo_right_img)
        f_lbl.place(x=5,y=0,width=720, height=130)
        #=====Search System=====
        Search_frame=LabelFrame(Right_frame, bd=2, bg="white", relief=RIDGE, text="Search
System",font=("times new roman",12,"bold"))
        Search_frame.place(x=5,y=150, width=710,height=70)
        Search_label=Label(Search_frame, text="Search BY:", font=("times new roman", 13,
"bold"),bg="green",fg="white")
        Search_label.grid(row=0,column=0,padx=10,pady=5,sticky=W)
        Search_combo=ttk.Combobox(Search_frame,font=("times new roman", 12, "bold"),state="Read
Only",width=15)
        Search_combo["values"]=("Select","Roll No","Phone No","Name")
        Search_combo.current(0)
        Search_combo.grid(row=0,column=1,pady=10,padx=2,sticky=W)
        Search_Entry=ttk.Entry(Search_frame,width=15,font=("times new roman", 13, "bold"))
        Search_Entry.grid(row=0,column=2,padx=10,pady=5,sticky=W)
        #Search
        Search_btn=Button(Search_frame,text="Search",width=14,font=("times new roman", 13,
"bold"),bg="blue",fg="white")
        Search_btn.grid(row=0,column=3)
        #Show All
        ShowAll_photo_btn=Button(Search_frame,text="Show All",width=14,font=("times new roman", 13,
"bold"),bg="blue",fg="white")
        ShowAll_photo_btn.grid(row=0,column=4)
        #=====Table Frame=====
        Table_frame=Frame(Right_frame, bd=2, bg="white", relief=RIDGE)
        Table_frame.place(x=5,y=210, width=710,height=350)
        scroll_x=ttk.Scrollbar(Table_frame,orient=HORIZONTAL)

```

```

scroll_y=ttk.Scrollbar(Table_frame,orient=VERTICAL)
self.student_table=ttk.Treeview(Table_frame,column=("Name","ID","Roll","Stream","Dep","Course",
,"Sem","Div","Gender","DOB","Email","Phone","Address","teacher","photo"),xscrollcommand=scroll_x.set,y
scrollcommand=scroll_y.set)

scroll_x.pack(side=BOTTOM,fill=X)
scroll_y.pack(side=RIGHT,fill=Y)
scroll_x.config(command=self.student_table.xview)
scroll_y.config(command=self.student_table.yview)
self.student_table.heading("Stream", text="Stream")
self.student_table.heading("Course", text="Course")
self.student_table.heading("Dep", text="Department")
self.student_table.heading("Sem", text="Semester")
self.student_table.heading("ID", text="StudentId")
self.student_table.heading("Name", text="Name")
self.student_table.heading("Div", text="Division")
self.student_table.heading("Roll", text="Roll No")
self.student_table.heading("Gender", text="Gender")
self.student_table.heading("DOB", text="DOB(DD/MM/YY)")
self.student_table.heading("Email", text="Email")
self.student_table.heading("Phone", text="Phone")
self.student_table.heading("Address", text="Address")
self.student_table.heading("teacher", text="Teacher")
self.student_table.heading("photo", text="PhotoSampleStatus")
self.student_table["show"]="headings"
self.student_table.column("Stream", width=100)
self.student_table.column("Course", width=100)
self.student_table.column("Dep", width=100)
self.student_table.column("Sem", width=100)
self.student_table.column("ID", width=100)
self.student_table.column("Name", width=100)
self.student_table.column("Roll", width=100)
self.student_table.column("Gender", width=100)
self.student_table.column("Div", width=100)
self.student_table.column("DOB", width=120)
self.student_table.column("Email", width=100)
self.student_table.column("Phone", width=100)
self.student_table.column("Address", width=100)
self.student_table.column("teacher", width=100)
self.student_table.column("photo", width=150)
self.student_table.pack(fill=BOTH, expand=1)
self.student_table.bind("<ButtonRelease>",self.get_cursor)
self.fetch_data()
#=====Function Declaration=====
def add_data(self):
    selected_date = datetime.strptime(self.var_dob.get(), "%m/%d/%y")
    current_date = datetime.now()
    if self.var_stream.get()=="Select Stream" or self.var_dep.get()=="Select Department" or
self.var_dep.get()=="Select Stream First" or self.var_semester.get()=="Select Semester" or
self.var_course.get()=="Select Course" or self.var_std_name.get()=="" or self.var_std_id.get()=="" or
self.var_div.get()=="Select Division" or self.var_roll.get()=="" or self.var_gender.get()=="Select
Gender" or self.var_dob.get()=="" or self.var_email.get()=="" or self.var_phone.get()=="" or
self.var_address.get()=="" or self.var_teacher.get()=="" or self.var_radio.get()=="":
        messagebox.showerror("Error","All fields are Required",parent=self.root)
    elif selected_date.date() > current_date.date():
        messagebox.showerror("Error","You Can't Set Upcoming Date",parent=self.root)
    elif current_date.date() < selected_date.date()+relativedelta(years=15):
        messagebox.showerror("Error","Age Should Be Greater than 15 Years Old",parent=self.root)
    elif re.search(r"\d",self.var_std_name.get().replace(" ","")) or re.search(r"^[a-zA-Z0-
9]",self.var_std_name.get().replace(" ","")):
        messagebox.showerror("Error","Name Should Not Contain Any Numeric or Special
Symbol!",parent=self.root)
    elif re.search(r"\d",self.var_teacher.get().replace(" ","")) or re.search(r"^[a-zA-Z0-
9]",self.var_teacher.get().replace(" ","")):
        messagebox.showerror("Error","Teacher Name Should Not Contain Any Numeric or Special
Symbol!",parent=self.root)
    else:
        try:
            int(self.var_std_id.get()) and int(self.var_roll.get()) and int(self.var_phone.get())
            phone=int(self.var_phone.get())
            email=self.var_email.get()
            if phone<=1000000000 or phone>=9999999999:
                messagebox.showerror("Error","Mobile No Should Be 10 Digit or A Valid
Number",parent=self.root)

```

```

else:
    try:
        email_validator.validate_email(email)
    try:
        conn=sqlite3.connect('Face Recognizer.db')
        cursor=conn.cursor()
        cursor.execute('''CREATE TABLE IF NOT EXISTS student(
                                Name TEXT,StudentID  INTEGER NOT NULL,RollNo INTEGER NOT
NULL,
                                Stream TEXT,Department TEXT,Course_Type TEXT,Sem TEXT,
                                Div TEXT,
                                Gender TEXT,DOB TEXT,
                                Email TEXT,PhoneNo INTEGER,Address TEXT,Teacher_Name TEXT,
                                Photosample TEXT,
                                PRIMARY KEY(StudentID,RollNo))''')
        if int(self.var_roll.get()) in self.student_Roll:
            messagebox.showerror("Error",f'{self.var_roll.get()} This Roll No is
already in Database',parent=self.root)
        else:
            cursor.execute("INSERT INTO student
(Stream,Department,Course_Type,Sem,StudentID,Name,Div,RollNo,Gender,DOB,Email,PhoneNo,Address,Teacher_
Name,Photosample) VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)", (

self.var_stream.get(),
self.var_dep.get(),
self.var_course.get(),
self.var_semester.get(),
self.var_std_id.get(),
self.var_std_name.get(),
self.var_div.get(),
self.var_roll.get(),
self.var_gender.get(),
self.var_dob.get(),
self.var_email.get(),
self.var_phone.get(),
self.var_address.get(),
self.var_teacher.get(),
self.var_radio.get()

))
            conn.commit()
            self.fetch_data()
            if self.var_radio.get()!="No":
                messagebox.showinfo("Photo", "Photo Capturing
Soon...",parent=self.root)
                self.generate_dataset()
            conn.close()
            messagebox.showinfo("Info", "Student Details Has Been added
Successfully",parent=self.root)
        except Exception as es:
            conn.close()
            messagebox.showerror("Error",f"Due To :{str(es)}",parent=self.root)
    except:
        messagebox.showerror("Error", "Invalid Email Address",parent=self.root)
except:
    messagebox.showerror("Error", "Student ID, Roll No, Phone NO should be
Integer",parent=self.root)
#====Fetch Data====

```

```

def fetch_data(self):
    try:
        conn=sqlite3.connect('Face Recognizer.db')
        cursor=conn.cursor()
        cursor.execute("SELECT * FROM student")
        data=cursor.fetchall()
        cursor.execute("SELECT RollNo FROM student")
        data_roll=cursor.fetchall()
        if len(data)!=0:
            self.student_table.delete(*self.student_table.get_children())
            for i in data:
                self.student_table.insert("",END,values=i)
            a=0
            for i in data_roll:
                self.student_Roll.append(data_roll[a])
                a=a+1
            self.student_Roll = [item[0] for item in data_roll]
            conn.commit()
        else:
            messagebox.showinfo("Info","No Details is Available To view\nAdd Data To
View",parent=self.root)
            conn.close()
    except Exception as es:
        conn.close()
        messagebox.showerror("error",f"Due To :{str(es)}",parent=self.root)

#=====get cursor=====
def get_cursor(self,event=""):
    try:
        cursor_focus=self.student_table.focus()
        content=self.student_table.item(cursor_focus)

        if len(content)!=0:
            data=content["values"]
            self.var_std_name.set(data[0]),
            self.var_std_id.set(data[1]),
            self.var_roll.set(data[2]),
            self.var_stream.set(data[3]),
            self.var_dep.set(data[4]),
            self.var_course.set(data[5]),
            self.var_semester.set(data[6]),
            self.var_div.set(data[7]),
            self.var_gender.set(data[8]),
            self.var_dob.set(data[9]),
            self.var_email.set(data[10]),
            self.var_phone.set(data[11]),
            self.var_address.set(data[12]),
            self.var_teacher.set(data[13]),
            self.var_radio.set(data[14])
            self.radio=data[14]
        else:
            messagebox.showinfo("Info","No Details is Available To view\nAdd Data To
View",parent=self.root)
    except Exception as es:
        messagebox.showerror("Error",f"Due To :{str(es)}",parent=self.root)

#=====delete=====
def delete_data(self):
    if self.var_std_id.get()=="":
        messagebox.showerror("Error","Student ID must be required",parent=self.root)
    else:
        try:
            delete=messagebox.askyesno("Question?","Do you want to Delete this student
details",parent=self.root)
            if delete>0:
                conn=sqlite3.connect('Face Recognizer.db')
                cursor=conn.cursor()
                sql="DELETE FROM student WHERE StudentID=?"
                value=(self.var_std_id.get(),)
                cursor.execute(sql,value)
            else:
                if not delete:
                    return
                conn.commit()
                conn.close()
                messagebox.showinfo("Info","Student Details Deleted Successfully",parent=self.root)

```



```

        self.var_std_name.get(),
        self.var_div.get(),
        self.var_roll.get(),
        self.var_gender.get(),
        self.var_dob.get(),
        self.var_email.get(),
        self.var_phone.get(),
        self.var_address.get(),
        self.var_teacher.get(),
        self.var_radio.get(),
        self.var_std_id.get()
    ))

    conn.commit()
    self.fetch_data()
    if self.var_radio.get()=="Yes" and self.radio=="No":
        messagebox.showinfo("Info","Photo Capturing
Soon...",parent=self.root)

        self.generate_dataset()
        messagebox.showinfo("Info","Student Details Updated
Successfully",parent=self.root)

        conn.close()
    except:
        messagebox.showerror("Error","Invalid Email
Address",parent=self.root)
    except:
        messagebox.showerror("Error"," A InValid Number",parent=self.root)

    except Exception as es:
        conn.close()
        messagebox.showerror("Error",f"Due To :{str(es)}",parent=self.root)
#====Reset Function====
def reset_data(self):
    self.var_stream.set("Select Stream"),
    self.var_dep.set("Select Department"),
    self.var_course.set("Select Course"),
    self.var_semester.set("Select Semester"),
    self.var_std_name.set(""),
    self.var_div.set(""),
    self.var_roll.set(""),
    self.var_gender.set("Select Gender"),
    self.var_dob.set(""),
    self.var_email.set(""),
    self.var_phone.set(""),
    self.var_address.set(""),
    self.var_teacher.set(""),
    self.var_radio.set(""),
    self.var_std_id.set("")
# =====Generate data set or Take photo samples =====
def generate_dataset(self):
    if self.var_stream.get()=="Select Stream" or self.var_dep.get()=="Select Department" or
self.var_dep.get()=="Select Stream First" or self.var_semester.get()=="Select Semester" or
self.var_course.get()=="Select Course" or self.var_std_name.get()=="" or self.var_std_id.get()=="" or
self.var_div.get()=="" or self.var_roll.get()=="" or self.var_gender.get()=="Select Gender" or
self.var_dob.get()=="" or self.var_email.get()=="" or self.var_phone.get()=="" or
self.var_address.get()=="" or self.var_teacher.get()=="" or self.var_radio.get()=="":
        messagebox.showerror("Error","All Fields are required",parent=self.root)
    elif self.var_radio.get()=="No":
        messagebox.showerror("Error","You Have Selected No Photo",parent=self.root)
    else:
        try:
            conn=sqlite3.connect('Face Recognizer.db')
            cursor=conn.cursor()

```

```

        cursor.execute("select * from student")
        myresult=cursor.fetchall()
        id=0
        for x in myresult:
            id+=1

        cursor.execute("UPDATE student SET Photosample=? WHERE StudentID=?", (
            lf.var_radio.get(),
            lf.var_std_id.get()
        ))

        conn.commit()
        self.fetch_data()
        conn.close()

#===== Load predefined data on face frontals from opencv=====
face_classifier=cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
def face_cropped(img):
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces=face_classifier.detectMultiScale(gray,1.5,5)
    #scaling factor=1.3
    #Minimum Neighbor=5
    for(x,y,w,h) in faces:
        face_cropped=img[y:y+h+1,x:x+w+1]
    return face_cropped
cap=cv2.VideoCapture(1)
img_id=0
while (True):
    ret,my_farme=cap.read()
    my_farme=cv2.flip(my_farme,1)
    cv2.imshow("Camera Is Ready To Capture Press C",my_farme)
    if cv2.waitKey(1)==ord('c'):
        break
cap=cv2.VideoCapture(1)
while True:
    ret,my_farme=cap.read()
    my_farme=cv2.flip(my_farme,1)
    if face_cropped(my_farme) is not None:
        img_id+=1
        face=cv2.resize(face_cropped(my_farme),(600,600))
        face=cv2.cvtColor(face,cv2.COLOR_BGR2GRAY)
        face=cv2.medianBlur(face,5)
        #file_name_path="Data/"+self.var_std_name.get()
        +"..."+self.var_roll.get()+"..."+str(img_id)+".jpg"
        file_name_path="Data/user."+self.var_std_id.get()+"."+str(img_id)+".jpg"
        #file_name_path="Data/"+self.var_std_name.get()+"."+self.var_std_id.get()+"."+
        str(img_id)+".jpg"
        cv2.imwrite(file_name_path,face)
        cv2.putText(face,str(img_id),(50,50),cv2.FONT_HERSHEY_COMPLEX,2,(0,255,0),2)
        cv2.imshow("Cropped Face",face)
        if cv2.waitKey(1)==13 or int(img_id)==100:
            break
    cap.release()
    cv2.destroyAllWindows()
    messagebox.showinfo("Result","Generating data sets
    completed!!!!",parent=self.root)
    #self.reset_data()
except Exception as es:
    cap.release()
    cv2.destroyAllWindows()
    messagebox.showerror("Error",f"Due To:{str(es)}",parent=self.root)

if __name__=="__main__":
    root=Tk()
    obj=Student(root)
    root.mainloop()

```

## ❖ Test Train Data.py

```
from tkinter import *
from tkinter import ttk
from PIL import Image, ImageTk
from tkinter import messagebox
import sqlite3
import cv2
import os
import numpy as np
def train_classifier():
    data_dir="Data"
    path=[os.path.join(data_dir,file) for file in os.listdir(data_dir)]
    faces=[]
    ids=[]
    for image in path:
        img=Image.open(image).convert('L') #Gray scale conversion
        imageNp=np.array(img,'uint8')
        id=int(os.path.split(image)[1].split('.')[1])
        faces.append(imageNp)
        ids.append(id)
        cv2.imshow("Training",imageNp)
        cv2.waitKey(1)==13
    ids=np.array(ids)
    #Train the classifier
    #clf=cv2.LBPHFaceRecognizer_create()
    clf=cv2.face.LBPHFaceRecognizer_create()
    clf.train(faces,ids)
    clf.write("classifier.xml")
    cv2.destroyAllWindows()
if __name__ == "__main__":
    train_classifier()
```

## ❖ Test Face Recognizer.py

```
from tkinter import *
from tkinter import ttk
from PIL import Image, ImageTk
from tkinter import messagebox
import sqlite3
import cv2
import os
from win32com.client import Dispatch
import numpy as np
import string
from time import strftime
import time
from datetime import datetime
#=====Attendance=====
def mark_attendance(ID, Roll, Name, Dept):
    with open("attendance.csv", "r+", newline="\n") as f:
        myDataList=f.readlines()
        ID_list=[]
        ID_Date={}
        now=datetime.now()
        a=-1
        for line in myDataList:
            entry=line.split(",")
            if entry[0]!='\n':
                ID_list.append(entry[0])
                data=[item.split(',') for item in entry]
                if len(data)>5:
                    a=a+1
                    for i in data:
                        #if data[5][0] not in Date_list:
                            #Date_list.append(data[5][0])
                            ID_Date[ID_list[a]]=data[5][0]
        Date=now.strftime("%d/%m/%y")
```



```

Time=now.strftime("%H:%M:%S")
if ID not in ID_list:
    f.writelines(f"\n{ID},{Roll},{Name},{Dept},{Time},{Date},present")
    #speak(f"{Name},YOUR ATTENDANCE SUCCESSFULLY CREATED")
else:
    if((ID_Date[ID]!=Date)): #'''and (ID not in ID_list)'''
        f.writelines(f"\n{ID},{Roll},{Name},{Dept},{Time},{Date},present")
        #speak(f"{Name},YOUR ATTENDANCE SUCCESSFULLY CREATED")
    else:
        return 1
# ===== face recognition =====#
def face_recognition():
    try:
        facedetect = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
        cam = cv2.VideoCapture(1)
        recognizer = cv2.face.LBPHFaceRecognizer_create()
        recognizer.read("classifier.xml")
        def getProfile(id):
            conn=sqlite3.connect("Face Recognizer.db")
            cursor=conn.execute("SELECT * FROM student WHERE StudentID=?", (str(id),))
            profile=None
            for row in cursor:
                profile=row
            conn.close()
            return profile
        while(True):
            ret,img=cam.read()
            img=cv2.flip(img,1)
            img=cv2.medianBlur(img,5)
            gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            faces=facedetect.detectMultiScale(gray,1.3,5)
            for(x,y,w,h) in faces:
                cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
                id,conf=recognizer.predict(gray[y:y+h,x:x+w])
                profile=getProfile(id)
                confidence = int((100*(1-conf/300)))
                if(profile != None):
                    if confidence > 85:
                        cv2.putText(img, "Name : "+str(profile[0]), (x,y-55),cv2.FONT_HERSHEY_COMPLEX,
1,(0,255,127),2)
                        cv2.putText(img, "Roll : "+str(profile[2]), (x,y-30),cv2.FONT_HERSHEY_COMPLEX,
1,(0,255,127),2)
                        cv2.putText(img, "Dep : "+str(profile[4]), (x,y-5),cv2.FONT_HERSHEY_COMPLEX,
1,(0,255,127),2)
                        if(mark_attendance(str(profile[1]),str(profile[2]),str(profile[0]),str(profile
[4]))):
                            cv2.putText(img, "Status : "+"Attended",
(x,y+h+20),cv2.FONT_HERSHEY_COMPLEX, 1,(255,255,255),2)
                        else:
                            cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),3)
                            cv2.putText(img,"Unknown Face",(x,y-
5),cv2.FONT_HERSHEY_COMPLEX,0.8,(255,255,255),3)
                            cv2.imshow("Face is Reading To QUIT Press Q",img)
                            if cv2.waitKey(1)==ord('q'):
                                break
                            cam.release()
                            cv2.destroyAllWindows()
            except Exception as es:
                messagebox.showerror("Error",f"Due To :{str(es)}")

if __name__ == "__main__":
    face_recognition()

```

## ❖ Attendance.py

```

from tkinter import*
from tkinter import ttk
from PIL import Image,ImageTk
from tkinter import messagebox
import sqlite3
import cv2

```

```

import os
import csv
from tkinter import filedialog
mydata=[]
class Attendance:
    def __init__(self,root):
        self.root=root
        root.title("Attendance DataSheet")
        root.geometry('1200x500+200+200')
        root.configure(bg="black")
        #Icon
        icon=PhotoImage(file=r"Images\icon.png")
        root.iconphoto(False,icon)
        main_frame=Frame(self.root,bd=2,bg="white")
        main_frame.place(x=7,y=7,width=1180,height=480)
        #Right label frame
        Right_frame=LabelFrame(main_frame,bd=2,bg="seagreen",relief=RIDGE,text=" Attendance
Details",font=("times new roman",35,))
        Right_frame.place(x=5,y=5,width=1158,height=460)
        table_frame=Frame (Right_frame, bd=2, relief=RIDGE, bg="yellow")
        table_frame.place (x=5,y=5, width=1145,height=390)
        openFile_btn=Button(main_frame, text="open File",command=self.importCsv, width=17,
font=("times new roman", 13, "bold"), bg="blue", fg="white")
        openFile_btn.place(x=600,y=20,width=200)
        # =====scroll bar table=====
        scroll_x=ttk.Scrollbar (table_frame, orient=HORIZONTAL)
        scroll_y=ttk.Scrollbar (table_frame, orient=VERTICAL)
        self.AttendaceReportTable=ttk.Treeview(table_frame,columns=("id", "roll", "name","department",
"time", "date", "attendance"),xscrollcommand=scroll_x.set, yscrollcommand=scroll_y.set)
        scroll_x.pack(side=BOTTOM, fill=X)
        scroll_y.pack(side=RIGHT, fill=Y)
        scroll_x.config(command=self.AttendaceReportTable.xview)
        scroll_y.config(command=self.AttendaceReportTable.yview)
        self.AttendaceReportTable.heading("id", text="Attendance ID")
        self.AttendaceReportTable.heading("roll", text="Roll")
        self.AttendaceReportTable.heading("name", text="Name")
        self.AttendaceReportTable.heading("department", text="Department")
        self.AttendaceReportTable.heading("time", text="Time")
        self.AttendaceReportTable.heading("date", text="Date")
        self.AttendaceReportTable.heading("attendance", text="Attendance")
        self.AttendaceReportTable["show"]="headings"
        self.AttendaceReportTable.column("id", width=100)
        self.AttendaceReportTable.column("roll", width=100)
        self.AttendaceReportTable.column("name", width=100)
        self.AttendaceReportTable.column("department", width=100)
        self.AttendaceReportTable.column("time", width=100)
        self.AttendaceReportTable.column("date", width=100)
        self.AttendaceReportTable.column("attendance", width=100)
        self.AttendaceReportTable.pack(fill=BOTH, expand=1)
        #self.AttendaceReportTable.bind("<ButtonRelease>",self.get_cursor)
        #====Fetch Data====
        def fetchData(self,rows):
            self.AttendaceReportTable.delete(*self.AttendaceReportTable.get_children())
            for i in rows:
                self.AttendaceReportTable.insert("",END,values=i)
        #====ImportCSV====
        def importCsv(self):
            try:
                global mydata
                mydata.clear()
                fln=filedialog.askopenfilename(initialdir=os.getcwd(),title="open CSV",filetypes=(("CSV
File","*csv"),("All File","*.*")),parent=self.root)
                with open(fln) as myfile :
                    csvread=csv.reader(myfile,delimiter=",")
                    for i in csvread:
                        mydata.append(i)
                    self.fetchData(mydata)
            except Exception as es:
                messagebox.showerror("Error",f"Deu To:{str(es)}",parent=self.root)
        #export csv
        def exportCsv(self):
            try:
                if len(mydata)<1:
                    messagebox.showerror("No Data","No Data found to export",parent=self.root)

```

```

        return False
    fln=filedialog.asksaveasfilename(initialdir=os.getcwd(),title="open CSV",filetypes=(("CSV
File","*csv"),("All File","* *")),parent=self.root)
    with open(fln,mode="w",newline="") as myfile:
        exp_write=csv.write(myfile,delimiter=",")
        for i in mydata:
            exp_write.writerow(i)
        messagebox.showinfo("Data Export","Your Data Exported
to"+os.path.basename(fln)+"successfully")
    except Exception as es:
        messagebox.showerror("Error",f"Deu To:{str(es)}",parent=self.root)
if __name__=="__main__":
    root=Tk()
    obj= Attendance(root)
    root.mainloop()

```

## ❖ Developer.py

```

from tkinter import*
from tkinter import ttk
from PIL import Image,ImageTk
from tkinter import messagebox
class Developer:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1530x790+0+0")
        self.root.title("Face Recognition System")
        #Icon
        icon=PhotoImage(file=r"Images\icon.png")
        root.iconphoto(False,icon)
        title_lbl=Label(self.root,text="DEVELOPER",font=("times new
roman",35,"bold"),bg="white",fg="blue")
        title_lbl.place(x=0,y=0,width=1530,height=45)
        img_top=Image.open(r"A:\Project\Images\Designer1.jpg")
        img_top=img_top.resize((1530,720),resample=0)
        self.photoimg_top=ImageTk.PhotoImage(img_top)
        f_lbl=Label(self.root,image=self.photoimg_top)
        f_lbl.place(x=0,y=55,width=1530,height=720)
        # Frame
        main_frame=Frame(f_lbl,bd=2,bg="white")
        main_frame.place(x=1000,y=0,width=500,height=600)
        img_top1=Image.open(r"A:\Project\Images\Designer1.jpg")
        img_top1=img_top1.resize((200,200),resample=0)
        self.photoimg_top1=ImageTk.PhotoImage(img_top1)
        f_lbl=Label(main_frame,image=self.photoimg_top1)
        f_lbl.place(x=300,y=0,width=200,height=200)
        #Developer info
        dev_label=Label(main_frame,text="Hello We are From Bankura Sammilani College",font=("times new
roman",20,"bold"),fg="blue",bg="white")
        dev_label.place(x=0,y=5)
        dev_label=Label(main_frame,text="We are representing Attendance System",font=("times new
roman",20,"bold"),fg="blue",bg="white")
        dev_label.place(x=0,y=40)
        img2=Image.open(r"A:\Project\Images\Designer1.jpg")
        img2=img2.resize((500,390),resample=0)
        self.photoimg2=ImageTk.PhotoImage(img2)
        f_lbl=Label(main_frame,image=self.photoimg2)
        f_lbl.place(x=0,y=210,width=500,height=390)
if __name__=="__main__":
    root=Tk()
    obj=Developer(root)
    root.mainloop()

```

## ❖ Help.py

```

from tkinter import*

from tkinter import ttk
from PIL import Image,ImageTk
from tkinter import messagebox

```

```
import cv2
from PIL import Image, ImageTk
class Help:
    def __init__(self, root):
        self.root = root
        self.root.geometry("1530x790+0+0")
        self.root.title("face Recognition System")
        title_lbl = Label(self.root, text="HELP DESK", font=("times new
roman", 35, "bold"), bg="white", fg="blue")
        title_lbl.place(x=0, y=0, width=1530, height=45)
        img_top = Image.open(r"A:\Project\Images\attendance.jpg")
        img_top = img_top.resize((1530, 720), resample=0)
        self.photoimg_top = ImageTk.PhotoImage(img_top)
        f_lbl = Label(self.root, image=self.photoimg_top)
        f_lbl.place(x=0, y=55, width=1530, height=720)
        dev_label = Label(f_lbl, text="Email:abcd@gmail.com", font=("times new
roman", 20, "bold"), fg="blue", bg="white")
        dev_label.place(x=550, y=260)
if __name__ == "__main__":
    root = Tk()
    obj = Help(root)
    root.mainloop()
```

# BIBLIOGRAPHY

## Books:

Core python programming by Dr. R. Nageswara Rao

## Websites for the libraries and algorithms:

1. <https://opencv.org/>
2. [https://docs.opencv.org/3.4/db/d28/tutorial\\_cascade\\_classifier.html](https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html)
3. <https://www.geeksforgeeks.org/face-recognition-with-local-binary-patterns-lbps-and-opencv/>
4. <https://www.w3schools.com/python/numpy>
5. <https://github.com/topics/frontal-face-detection>

**THANK  
YOU**

**We are thankful to**

**BANKURA SAMMILANI COLLEGE**

***Kenduadihi ❄ Bankura***