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: External:

1.Mr. Sanjoy sen 1.

|  |  |  |
| --- | --- | --- |
| 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 |
|  | 1. Software Requirement | 14 |
|  | 1. Hardware Requirement | 14 |
| 8. | System Design | 15 |
|  | 1. 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 |

index

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.

i

**Feasibility Study in Face Recognition Attendance Systems**

Feasibility study

**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

j

* 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

Update Photo

Update Details

END

Update User

Activate Registration

User ID/Password for LOGIN

START

SYSTEM FLOW FOR ADMIN

NO YES

VALID

NO YES

SYSTEM FLOW FOR ADMIN/CLIENT

END

Update Information

Registration

Registered

Active Registration

VALID

User ID/Password for LOGIN

Start

**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.A**dministrator**: 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**

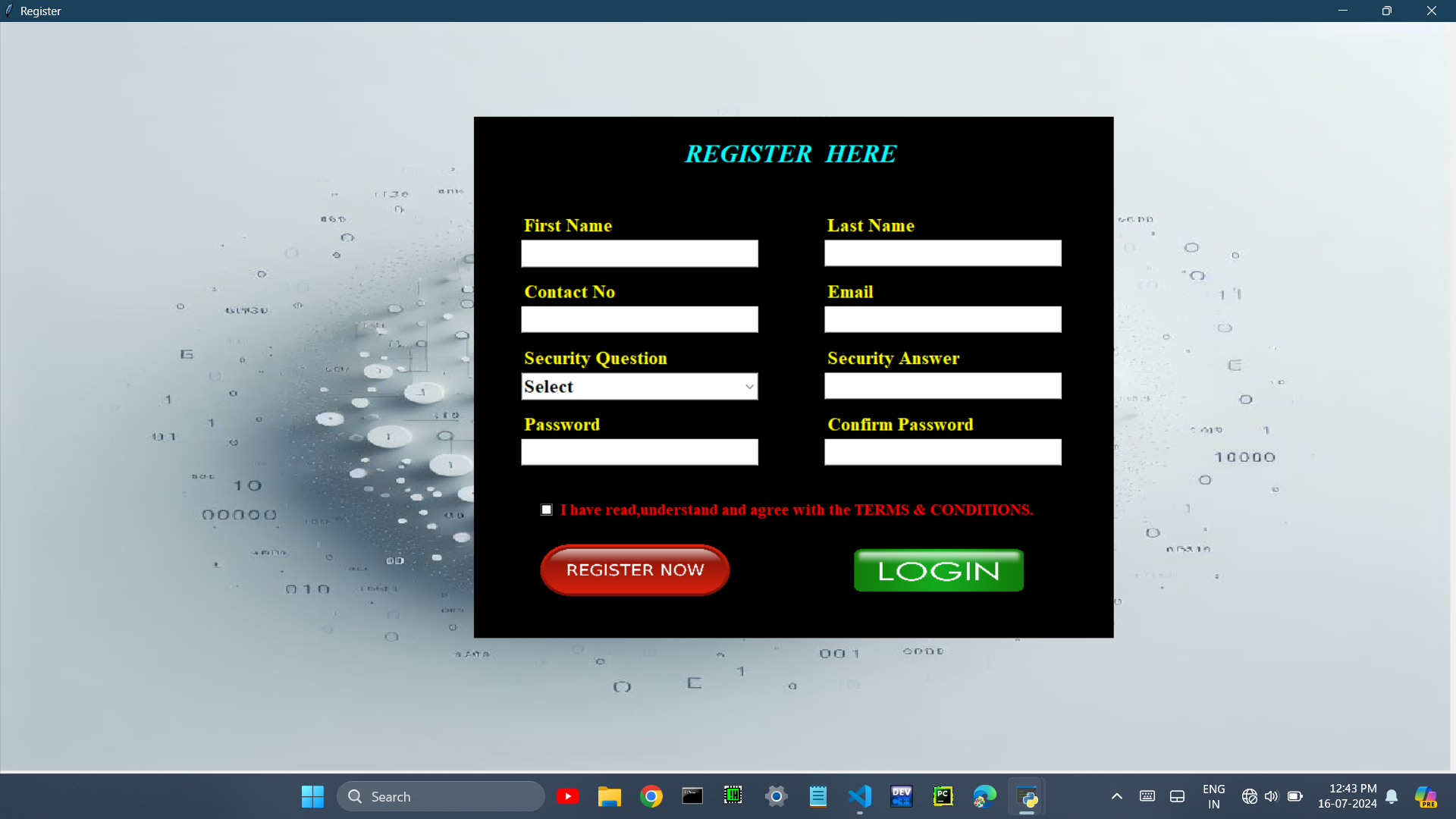
We are hereby confessing that, it is a mini project and it is not completed successfully. We will work more on it make it significant and useful. There are some future implementations we’ve thought

* 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

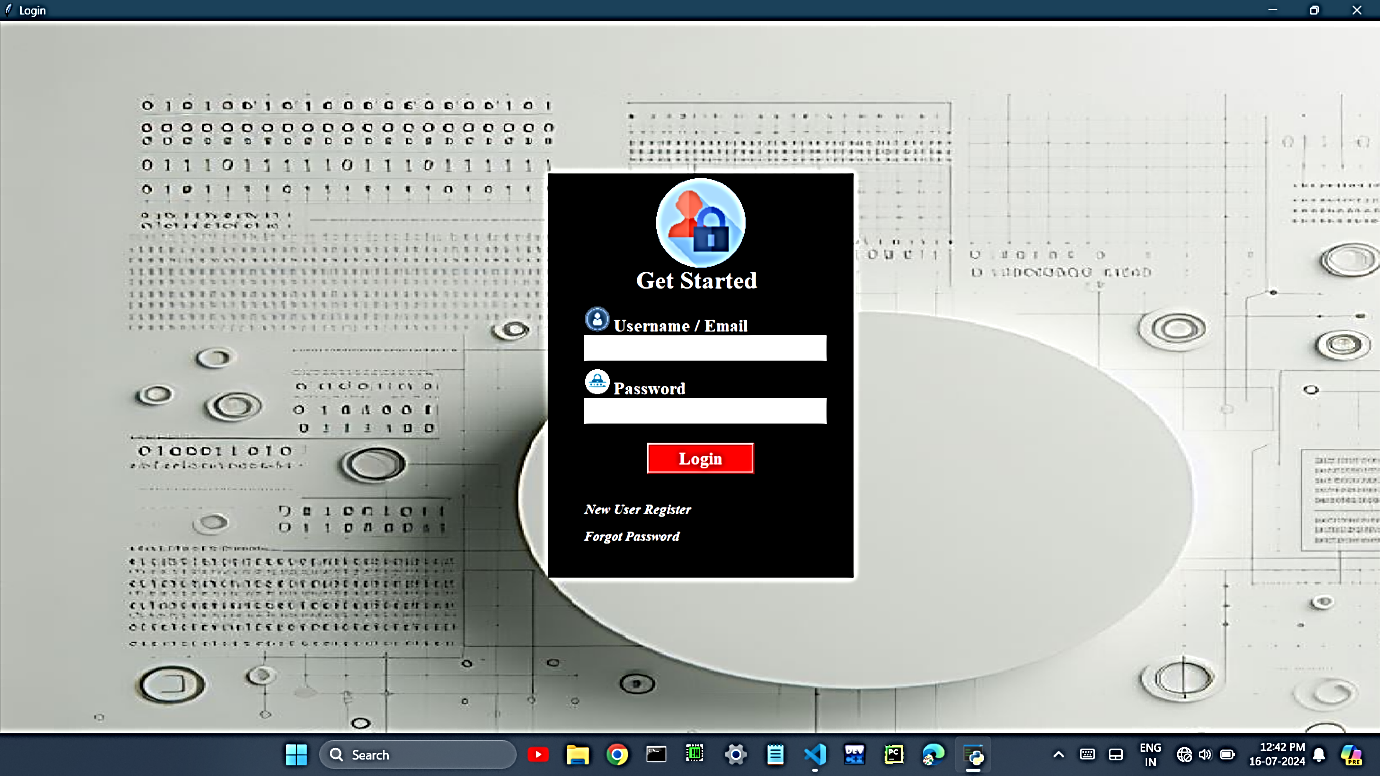
We will mainly focus on enhancing its accuracy, security and user experience.

FORM LAYOUT

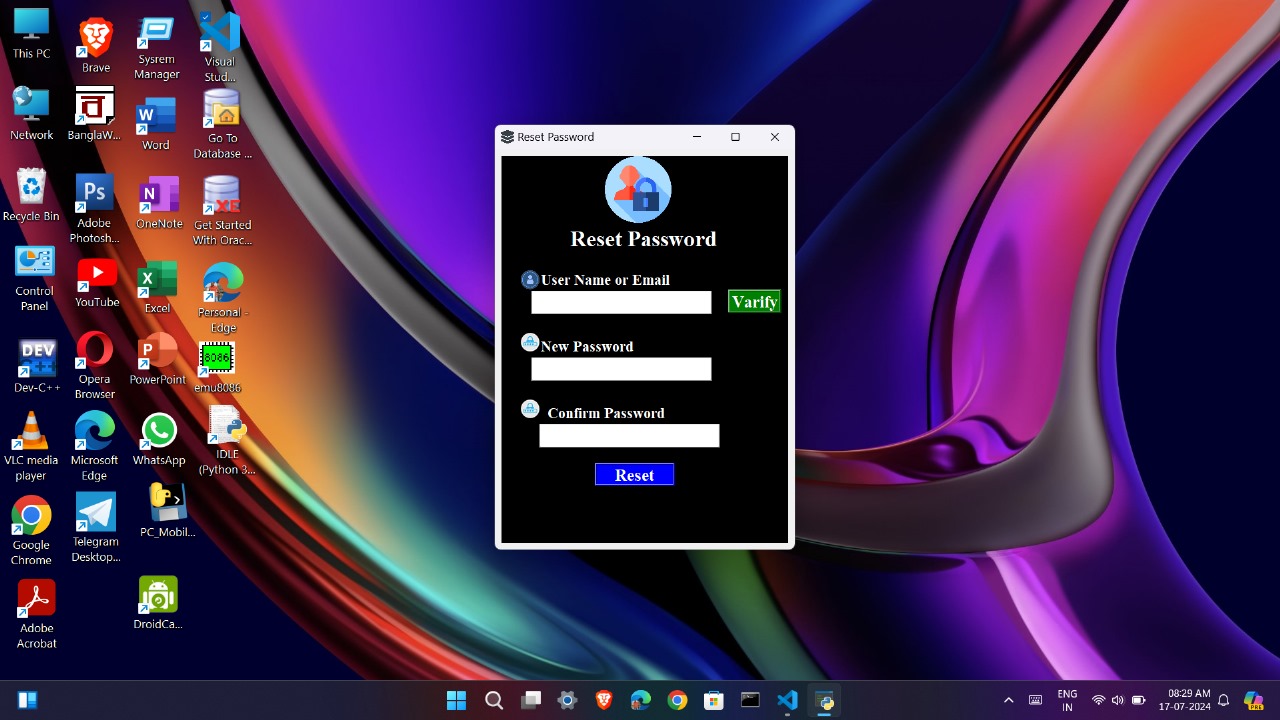
**REGISTRATION PAGE**

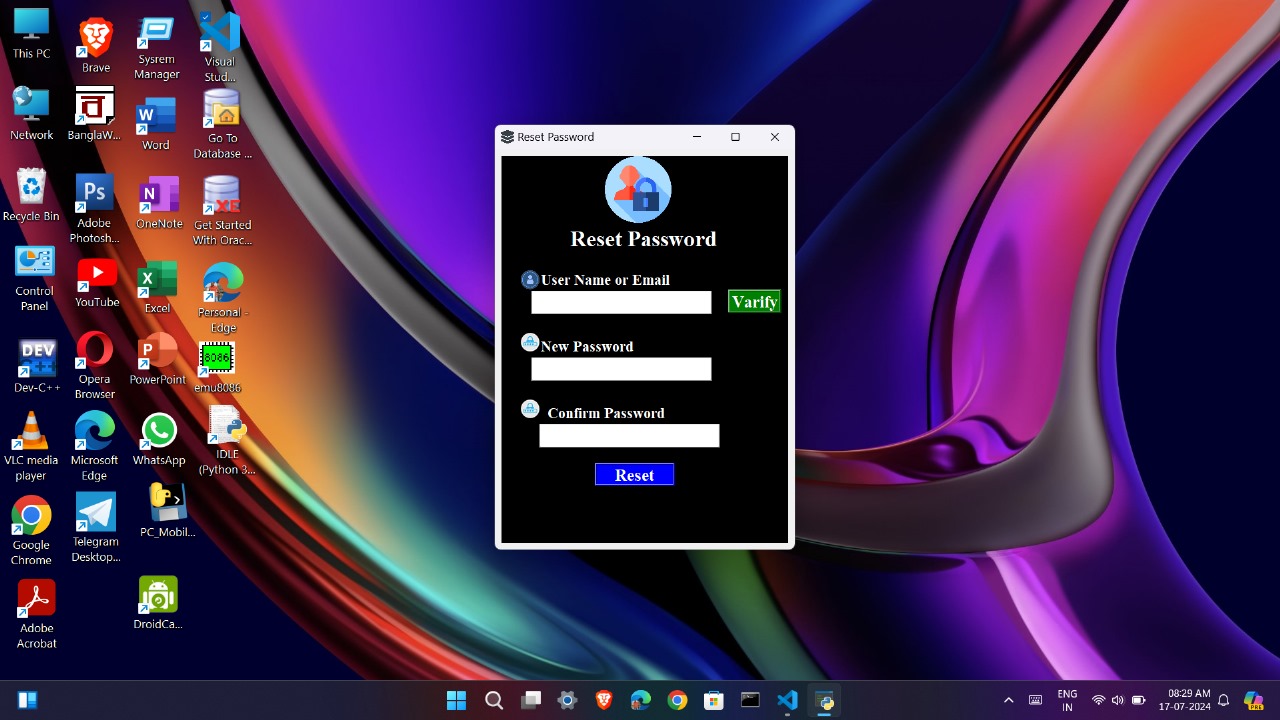


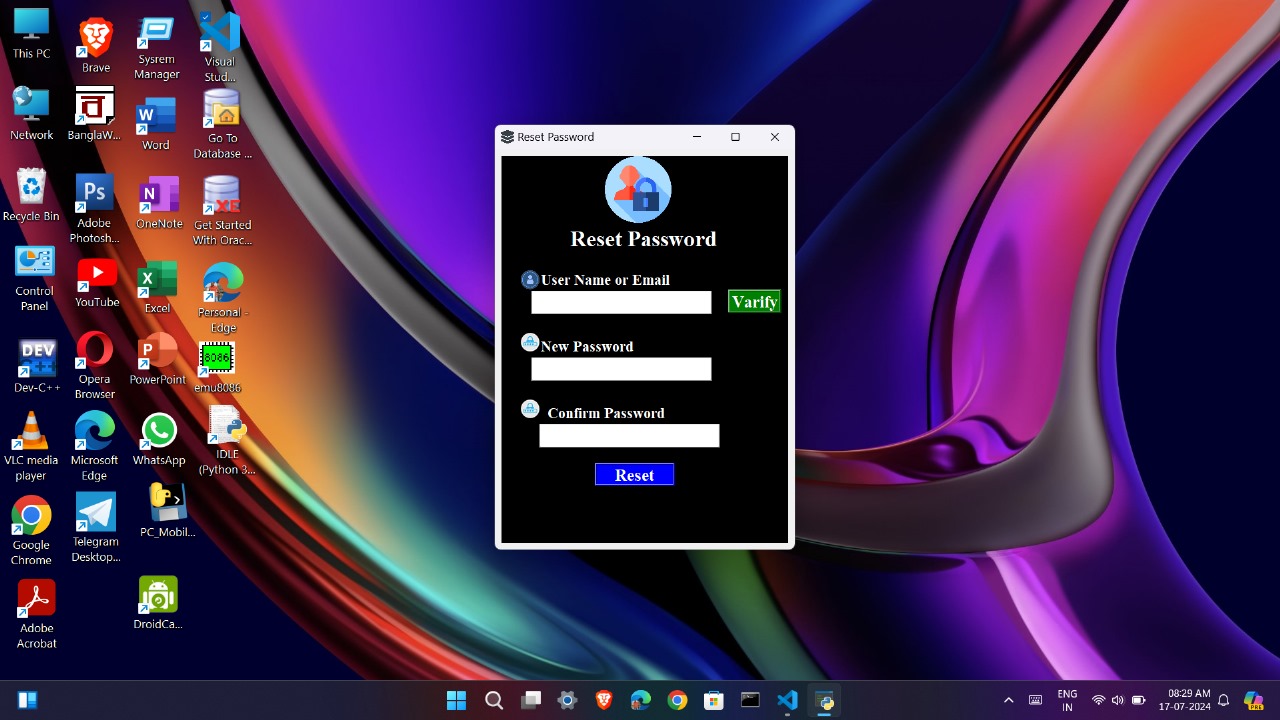
**LOGIN PAGE**



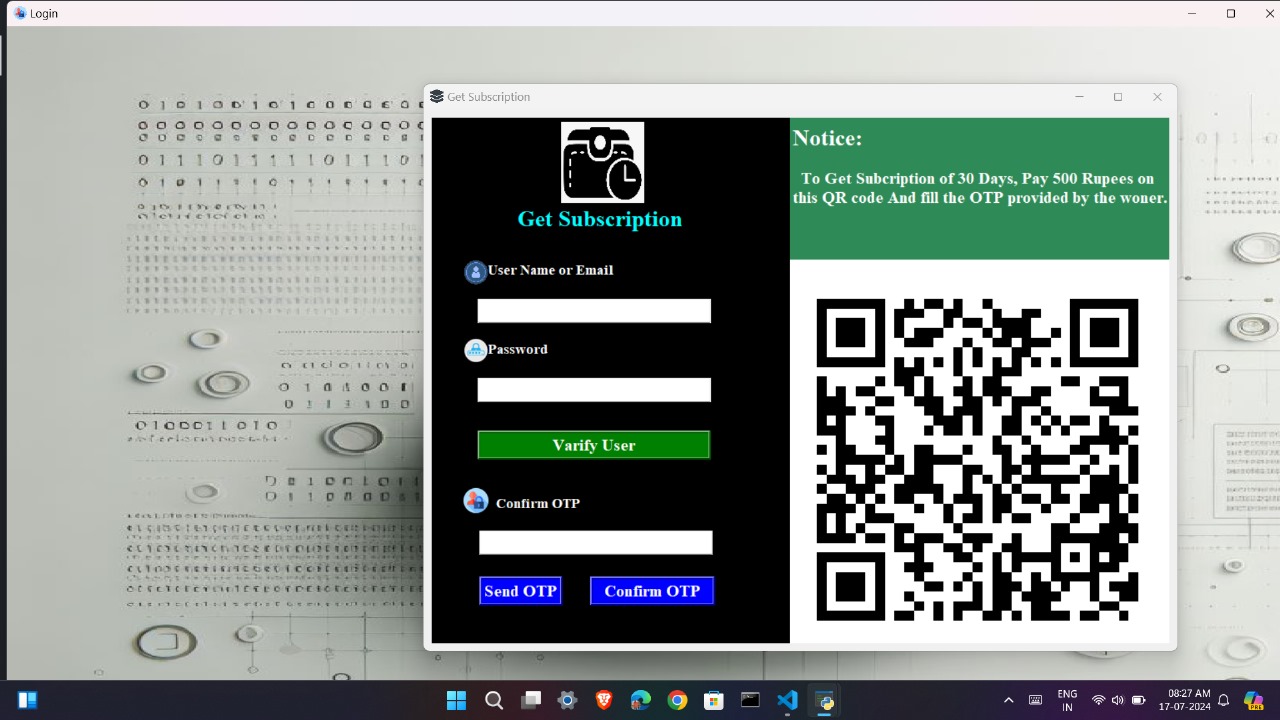
**RESET PASSWORD**

****

****



**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",activebackground="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",activebackground="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\bg1.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",font=("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 Charecter",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

        varifynbtn=Button(frame,text="Varify",command=self.varify,font=("times new roman",15,"bold"),bd=3,relief=RIDGE,fg="white",bg="green")

        varifynbtn.place(x=290,y=180,width=80,height=35)

#==========Function Declaration======

    #=====Varify=====

    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","Varify 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",font=("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.varifyOTP,font=("times new roman",15,"bold"),bd=3,relief=RIDGE,fg="white",bg="blue")

        ConfirmOTP.place(x=190,y=550,width=150,height=35)

        #Varify User

        varifynbtn=Button(frameBlack,text="Varify User",command=self.varify,font=("times new roman",15,"bold"),bd=3,relief=RIDGE,fg="white",bg="green")

        varifynbtn.place(x=55,y=375,width=280,height=35)

#========Function Declaration=====

    def varify(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 Varify 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 varifyOTP(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)

        #======Varables=====

        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 cource",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===

        cource\_Type\_label=Label(current\_course\_frame, text="Course Type:", font=("times new roman", 13, "bold"),bg="white")

        cource\_Type\_label.grid(row=1,column=0,pady=10,sticky=W)

        cource\_Type\_combo=ttk.Combobox(current\_course\_frame,textvariable=self.var\_course,font=("times new roman", 12, "bold"),state="Read Only")

        cource\_Type\_combo["values"]=("Select Course","Hons","Program")

        cource\_Type\_combo.current(0)

        cource\_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=("times 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=("times 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=("times 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,yscrollcommand=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 reqired",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.reset\_data()

                self.fetch\_data()

            except Exception as es:

                        conn.close()

                        messagebox.showerror("Error",f"Due To :{str(es)}",parent=self.root)

    #====Upadte Function====

    def update\_data(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)

        else:

            try:

                Update=messagebox.askyesno("Update","Do you want to Update this student details",parent=self.root)

                if Update!=True:

                    if not Update:

                         return

                else:

                    email=self.var\_email.get()

                    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 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:

                            phone=int(self.var\_phone.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)

                                    conn=sqlite3.connect('Face Recognizer.db')

                                    cursor=conn.cursor()

                                    if self.radio=="Yes" and self.var\_radio.get()=="No":

                                         messagebox.showwarning("Warning","Already Photo Available You Can't Deny it",parent=self.root)

                                         self.var\_radio.set("Yes")

                                    cursor.execute("UPDATE student SET Stream=?,Department=?,Course\_Type=?,Sem=?,Name=?,Div=?,RollNo=?,Gender=?,DOB=?,Email=?,PhoneNo=?,Address=?,Teacher\_Name=?,Photosample=? WHERE StudentID=?",(

                                                                                                            self.var\_stream.get(),

                                                                                                            self.var\_dep.get(),

                                                                                                            self.var\_course.get(),

                                                                                                            self.var\_semester.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(),

                                                                                                            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=?",(

                                                                                                    self.var\_radio.get(),

                                                                                                    self.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()

* **Attendence.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)+"succesfully")

        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

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 Snowflake with solid fill Bankura