Abstract:

As you read above my project is college management system which manages data of college this application is managed by college admin. So basically in this project there will be an login and registration page which is connected through database and it has authentication and validation of field also after registering you are able to use it then after get login dashboard page will open where 3 options are there for attendance, for storing data of student and inserting also and also for faculty data storing and inserting. According to user it selects its option. Now if user selected the student data button there will be student data page will open from where you can insert the new data records and update the data of student and also you can see the records that you entered and if you want to search any student by its name, roll number, or contact you can search from there and if you want to see all the data there will be a button to show all the data after selecting it will display all the data. Similarly for the faculty also where you can perform all operations as you perform in student data.

Now there comes an attendance system so here the attendance system is somewhat unique where the attendance can be done with biometric face-recognition where the system scan the student face and insert his name and the time when he scanned his face both data will be inserted in .csv file. If the image of that data is not present or not match with our data then it will show the failed message on face box. After when necessary you can see the attendance of student.

Modules:

Basically there are 4 modules:

1) Login and Registration:

In this module if you are the new admin to use this you can first register your data and then login to use the application.

2) Attendance:

This module is used to take attendance of an student in which the face of student are recognize and if it match with our stored image then it will insert the data in .csv file his and when he entered(timing).

3) Student data:

From this module you can add new data of students and you can update the data of student and when necessary you can search the data by its name, roll no, contact.

4) Faculty data:

From this module you can add new data of faculty member and you can update the data of faculty and when necessary you can search the data by its name, faculty id, contact.

Stockholders:

1) Principle:

Principle will manage all the parts such as attendance manages data of student and faculties.

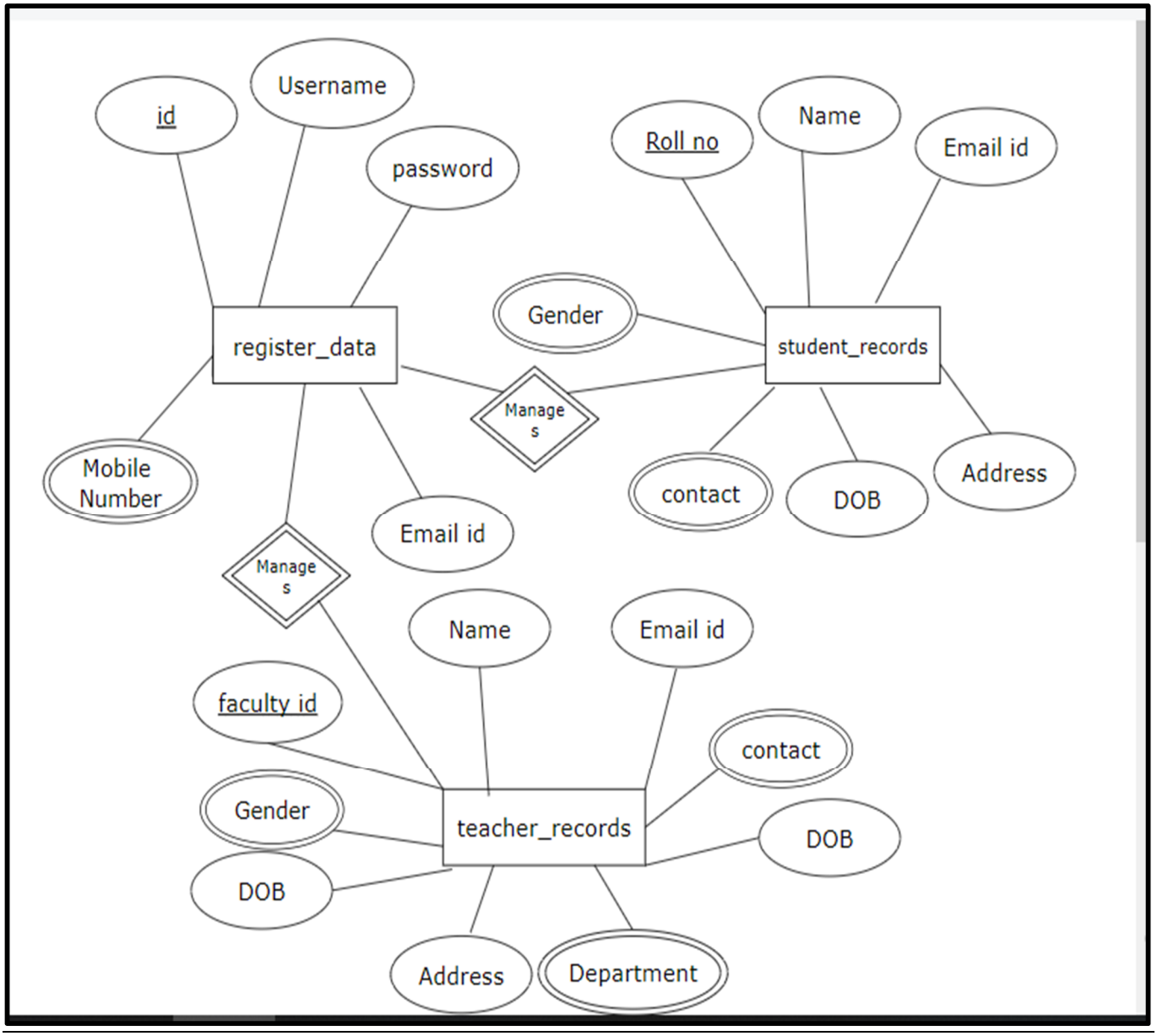
2) Teachers:

Teachers can see the data of student and his also and if possible they can add the data of student also and teachers can use the attendance system also.

3) Students:

Students use this for his or her attendance purpose.

ER diagram:


id
Username
password
register_data
Roll no
Name
Email id
student_records
Gender
Manage
s
Mobile
Number
Email id
contact
DOB
Address
Manage
s
Name
Email id
faculty id
contact
Gender
teacher_records
DOB
DOB
Address
Department

Code:

File: login

```
def account_not_have(self):
        self.main_window.destroy()
        import register

    def authenticate(self):
        if self.e1.get()=="" or self.e2.get()=="":
            messagebox.showerror("Error","All fields are
required",parent=self.main_window)
        else:
            try:
                con=connect(host='localhost',user='root',password='',db='register')
                cur=con.cursor()
                cur.execute("select Username,password from register_data where
Username=%s and password=%s",(self.e1.get(),self.e2.get()))
                row=cur.fetchone()
                #print(row)
                if row==None:
                    messagebox.showerror("Error","Invalid Username or
Password",parent=self.main_window)

                else:
                    messagebox.showinfo("Success","Login
Successful",parent=self.main_window)
                    self.main_window.destroy()
                    import main_page
                con.close()

            except Exception as ae:
                messagebox.showerror("Error",f"Error due to
{str(ae)}",parent=self.main_window)
```

File: register

```
def account(self):
        self.main_window.destroy()
        import login

    def clear(self):
        self.e1.delete(0,END)
        self.e2.delete(0,END)
        self.e3.delete(0,END)
        self.e4.delete(0,END)
        self.e5.delete(0,END)


    def connecting(self):
        if self.e1.get()=="" or self.e2.get()=="" or self.e3.get()=="" or
self.e4.get()=="" or self.e5.get()=="":
            messagebox.showerror("Error","All fields are
required",parent=self.main_window)
        elif self.e2.get()!=self.e4.get():
            messagebox.showerror("Error","Password and confirm password
should be same",parent=self.main_window)
        else:
            try:
                con=connect(host='localhost',user='root',password='',db='register')
                cur=con.cursor()
                cur.execute("select * from register_data where
email_id=%s",self.e5.get())
                row=cur.fetchone()
               # print(row)

                if row!=None:
                    messagebox.showerror("Error","User already Exist,Please try
with another Email",parent=self.main_window)

                else:
                    cur.execute("insert into
register_data(Username,password,mobile_number,email_id)
values(%s,%s,%s,%s)",
                                (
                                    self.e1.get(),
                                    self.e2.get(),
                                    self.e3.get(),
```

```
                                self.e5.get(),
                            ))
                con.commit()
                con.close()
                messagebox.showinfo("Success","Registered
Successful",parent=self.main_window)
                self.clear()
                self.main_window.destroy()
                import login
            except Exception as es:
                messagebox.showerror("Error",f"Error due to:
{str(es)}",parent=self.main_window)
```

File: Student_data

```
self.Student_table.bind("<ButtonRelease-1>",self.get_cursor)




    def back(self):
        self.window.destroy()
        import main_page


    def update_data(self):
        con=connect(host='localhost',user='root',password='',db='register')
        cur=con.cursor()
        cur.execute("update student_records set
Name=%s,Email_id=%s,Gender=%s,Contact=%s,DOB=%s,Address=%s
where Roll=%s",
                            (
                                self.name.get(),
                                self.email.get(),
                                self.gender.get(),
                                self.contact.get(),
                                self.dob.get(),
                                self.address.get("1.0",END),
                                self.roll.get()
                            ))
        con.commit()
        self.fetch_data()
        con.close()
```

```
        messagebox.showinfo("Success","Data updated
Successfully",parent=self.window)

    def clear(self):
        self.roll.delete(0,END)
        self.name.delete(0,END)
        self.email.delete(0,END)
        self.contact.delete(0,END)
        self.gender.delete("1",END)
        self.dob.delete(0,END)
        self.address.delete("1.0",END)


    def student_detail(self):
        if (self.roll.get()=="" or self.name.get()=="" or self.email.get()==""  or
self.contact.get()=="" or self.gender.get()=="" or self.dob.get()=="" or
self.address.get("1.0",END)==""):
            messagebox.showerror("Error","All fields are
required",parent=self.window)
        else:
            try:
                con=connect(host='localhost',user='root',password='',db='register')
                cur=con.cursor()
                cur.execute("select * from student_records where
Roll=%s",self.roll.get())
                row=cur.fetchone()
                # print(row)

                if row!=None:
                    messagebox.showerror("Error","Student already
Exist",parent=self.window)

                else:
                    cur.execute("insert into
student_records(Roll,Name,email_id,Gender,Contact,DOB,Address)
values(%s,%s,%s,%s,%s,%s,%s)",
                                (
                                    self.roll.get(),
                                    self.name.get(),
                                    self.email.get(),
                                    self.gender.get(),
                                    self.contact.get(),
                                    self.dob.get(),
```

```
                                self.address.get("1.0",END),
                            ))
                con.commit()
                self.fetch_data()
                con.close()
                messagebox.showinfo("Success","Student Record Inserted
Successful",parent=self.window)
                self.clear()

            except Exception as es:
                messagebox.showerror("Error",f"Error due to:
{str(es)}",parent=self.window)
    def fetch_data(self):
        con=connect(host='localhost',user='root',password='',db='register')
        cur=con.cursor()
        cur.execute("select * from student_records")
        rows=cur.fetchall()

        if len(rows)!=0:
            self.Student_table.delete(*self.Student_table.get_children())
            for row in rows:
                self.Student_table.insert("",END,values=row)
            con.commit()
        con.close()

    def get_cursor(self,ev):
        cursor_row=self.Student_table.focus()
        contents=self.Student_table.item(cursor_row)
        row=contents['values']

        self.roll_var.set(row[0])
        self.name_var.set(row[1])
        self.email_var.set(row[2])
        self.gender_var.set(row[3])
        self.contact_var.set(row[4])
        self.dob_var.set(row[5])
        self.address.delete("1.0",END)
        self.address.insert(END,row[6])

    def search_data(self):
        con=connect(host='localhost',user='root',password='',db='register')
        cur=con.cursor()
```

```
        cur.execute("select * from student_records where
"+str(self.search_by.get())+" LIKE '%"+str(self.search_txt.get())+"%'")
        rows=cur.fetchall()

        if len(rows)!=0:
            self.Student_table.delete(*self.Student_table.get_children())
            for row in rows:
                self.Student_table.insert("",END,values=row)
            con.commit()
        con.close()
```

File: faculty_data

```
def back(self):
        self.window.destroy()
        import main_page


    def update_data(self):
        con=connect(host='localhost',user='root',password='',db='register')
        cur=con.cursor()
        cur.execute("update teacher_record set
Name=%s,Email_id=%s,Gender=%s,Contact=%s,DOB=%s,Address=%s,Department=%s
where Faculty_id=%s",
                            (
                                self.name.get(),
                                self.email.get(),
                                self.gender.get(),
                                self.contact.get(),
                                self.dob.get(),
                                self.address.get("1.0",END),
                                self.department.get(),
                                self.roll.get()
                            ))
        con.commit()
        self.fetch_data()
        con.close()
        messagebox.showinfo("Success","Data updated Successfully",parent=self.window)

    def clear(self):
        self.roll.delete(0,END)
        self.name.delete(0,END)
        self.email.delete(0,END)
```

```
        self.contact.delete(0,END)
        self.gender.delete("1",END)
        self.dob.delete(0,END)
        self.address.delete("1.0",END)
        self.department.delete(0,END)


    def student_detail(self):
        if (self.roll.get()=="" or self.name.get()=="" or self.email.get()==""  or
self.contact.get()=="" or self.gender.get()=="" or self.dob.get()=="" or
self.address.get("1.0",END)=="" or self.department.get()==""):
            messagebox.showerror("Error","All fields are required",parent=self.window)
        else:
            try:
                con=connect(host='localhost',user='root',password='',db='register')
                cur=con.cursor()
                cur.execute("select * from teacher_record where Faculty_id=%s",self.roll.get())
                row=cur.fetchone()
               # print(row)

                if row!=None:
                    messagebox.showerror("Error","Student already Exist",parent=self.window)

                else:
                    cur.execute("insert into teacher_record
(Faculty_id,Name,email_id,Gender,Contact,DOB,Address,Department)
values(%s,%s,%s,%s,%s,%s,%s,%s)",
                                (
                                    self.roll.get(),
                                    self.name.get(),
                                    self.email.get(),
                                    self.gender.get(),
                                    self.contact.get(),
                                    self.dob.get(),
                                    self.address.get("1.0",END),
                                    self.department.get(),
                                ))
                    con.commit()
                    self.fetch_data()
                    con.close()
                    messagebox.showinfo("Success","Faculty Record Inserted
Successful",parent=self.window)
                    self.clear()
```

```
            except Exception as es:
                messagebox.showerror("Error",f"Error due to: {str(es)}",parent=self.window)
    def fetch_data(self):
        con=connect(host='localhost',user='root',password='',db='register')
        cur=con.cursor()
        cur.execute("select * from teacher_record")
        rows=cur.fetchall()

        if len(rows)!=0:
            self.Faculty_table.delete(*self.Faculty_table.get_children())
            for row in rows:
                self.Faculty_table.insert("",END,values=row)
            con.commit()
        con.close()

    def get_cursor(self,ev):
        cursor_row=self.Faculty_table.focus()
        contents=self.Faculty_table.item(cursor_row)
        row=contents['values']

        self.roll_var.set(row[0])
        self.name_var.set(row[1])
        self.email_var.set(row[2])
        self.gender_var.set(row[3])
        self.contact_var.set(row[4])
        self.dob_var.set(row[5])
        self.address.delete("1.0",END)
        self.address.insert(END,row[6])
        self.department_var.set(row[7])

    def search_data(self):
        con=connect(host='localhost',user='root',password='',db='register')
        cur=con.cursor()
        cur.execute("select * from teacher_record where "+str(self.search_by.get())+" LIKE
'%"+str(self.search_txt.get())+"%'")
        rows=cur.fetchall()

        if len(rows)!=0:
            self.Faculty_table.delete(*self.Faculty_table.get_children())
            for row in rows:
                self.Faculty_table.insert("",END,values=row)
            con.commit()
```

```
con.close()
```

File: Attendance

```
import cv2
import numpy as np
import face_recognition
import os
from datetime import datetime


#taking images from a directory and storing in array
path='D:/AMAN/college practicals sem3/advanced
python/code/amanproject/login/images'
images=[]
classnames=[]
mylist=os.listdir(path)
print(mylist)

#Storing name from the filename
for cl in mylist:
    curImg=cv2.imread(f'{path}/{cl}')
    images.append(curImg)
    classnames.append(os.path.splitext(cl)[0])
print(classnames)

#encoding the images from the file
def findEncodings(images):
    encodeList=[]
    for img in images:
        img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
        encode=face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList


def markAttendance(name):
    with open('attendance.csv','r+') as f:
        mydatalist=f.readlines()
        nameList=[]
        for line in mydatalist:
            entry=line.split(',')
            nameList.append(entry[0])
```

```
        if name not in nameList:
            now=datetime.now()
            dtstring=now.strftime('%H:%M:%S')
            f.writelines(f'{name},{dtstring}')

encodeListKnown=findEncodings(images)
print("Encoding complete")

#capturing video
cap=cv2.VideoCapture(0)

while True:
    #reading the webcam image
    success,img=cap.read()
    imgs=cv2.resize(img,(0,0),None,0.25,0.25) #resizing the file image
    imgs=cv2.cvtColor(imgs,cv2.COLOR_BGR2RGB)

    facesCurFrame=face_recognition.face_locations(imgs) #finding face
location
    encodesCurFrame=face_recognition.face_encodings(imgs,facesCurFrame)
#encoding the webcam images

    for encodeFace,faceLoc in zip(encodesCurFrame,facesCurFrame):

matches=face_recognition.compare_faces(encodeListKnown,encodeFace)
#comparing file image and webcam imagesss
        facedis=face_recognition.face_distance(encodeListKnown,encodeFace)
#finding the distance between both the images

        #print(facedis)
        #finfing minimum distance of images
        matchIndex=np.argmin(facedis)
        #print(matchIndex)
        #we get multiple faces sometimes so we need to find the minimum
distance between the images if it matches it will show the name
        if matches[matchIndex]:
            #print(matches[matchIndex])
            name=classnames[matchIndex].upper()
            #print(name)
            y1,x2,y2,x1=faceLoc
            y1,x2,y2,x1=y1*4,x2*4,y2*4,x1*4
            cv2.rectangle(img,(x1,y1),(x2,y2),(0,255,0),2)
            cv2.rectangle(img,(x1,y2-35),(x2,y2),(0,255,0),cv2.FILLED)
```

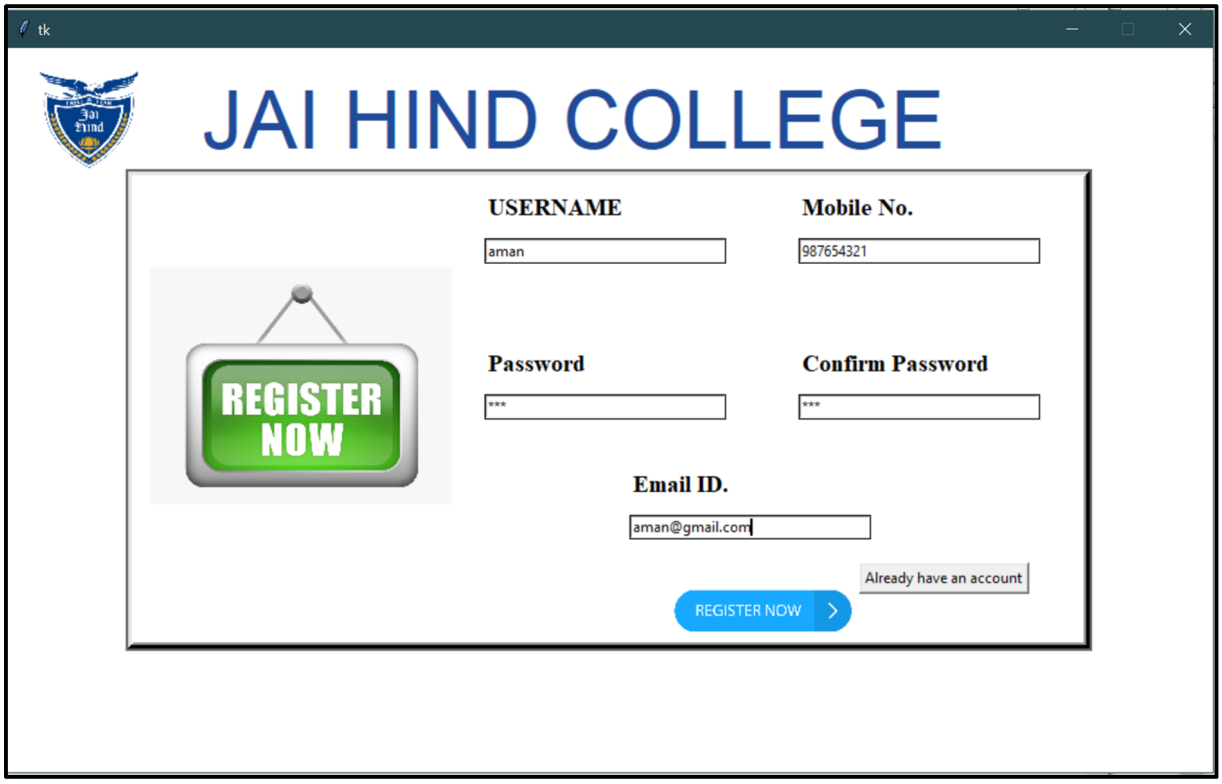
```
            cv2.putText(img,name,(x1+6,y2-
6),cv2.FONT_HERSHEY_COMPLEX,1,(255,255,255),2)
            markAttendance(name)
        else:
            y1,x2,y2,x1=faceLoc
            y1,x2,y2,x1=y1*4,x2*4,y2*4,x1*4
            cv2.rectangle(img,(x1,y1),(x2,y2),(0,0,255),2)
            cv2.rectangle(img,(x1,y2-35),(x2,y2),(0,0,255),cv2.FILLED)
            cv2.putText(img,"Failed",(x1+6,y2-
6),cv2.FONT_HERSHEY_COMPLEX,1,(255,255,255),2)

    cv2.imshow('Webcam',img)
    cv2.waitKey(1)
```

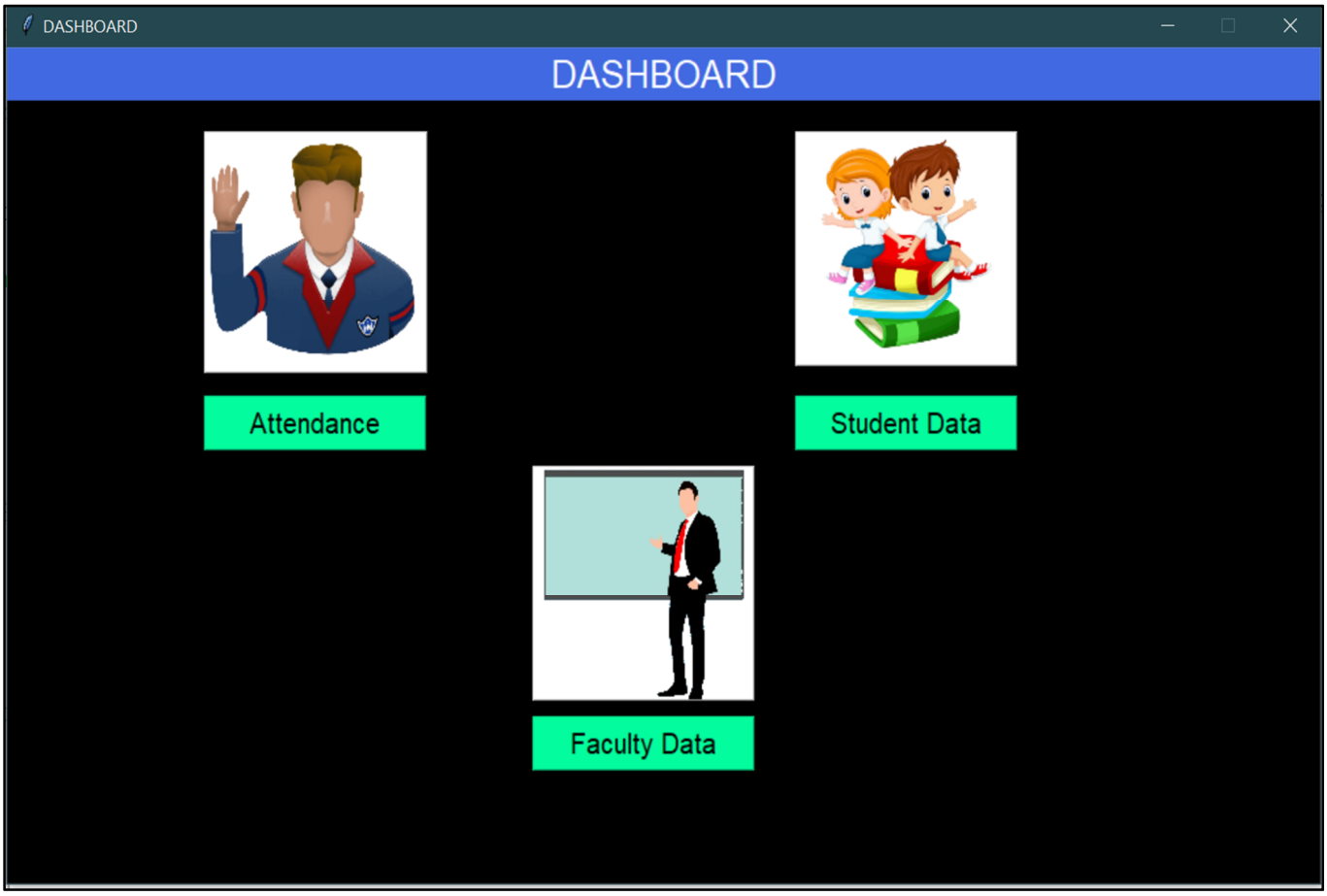
Screenshots:

Login page

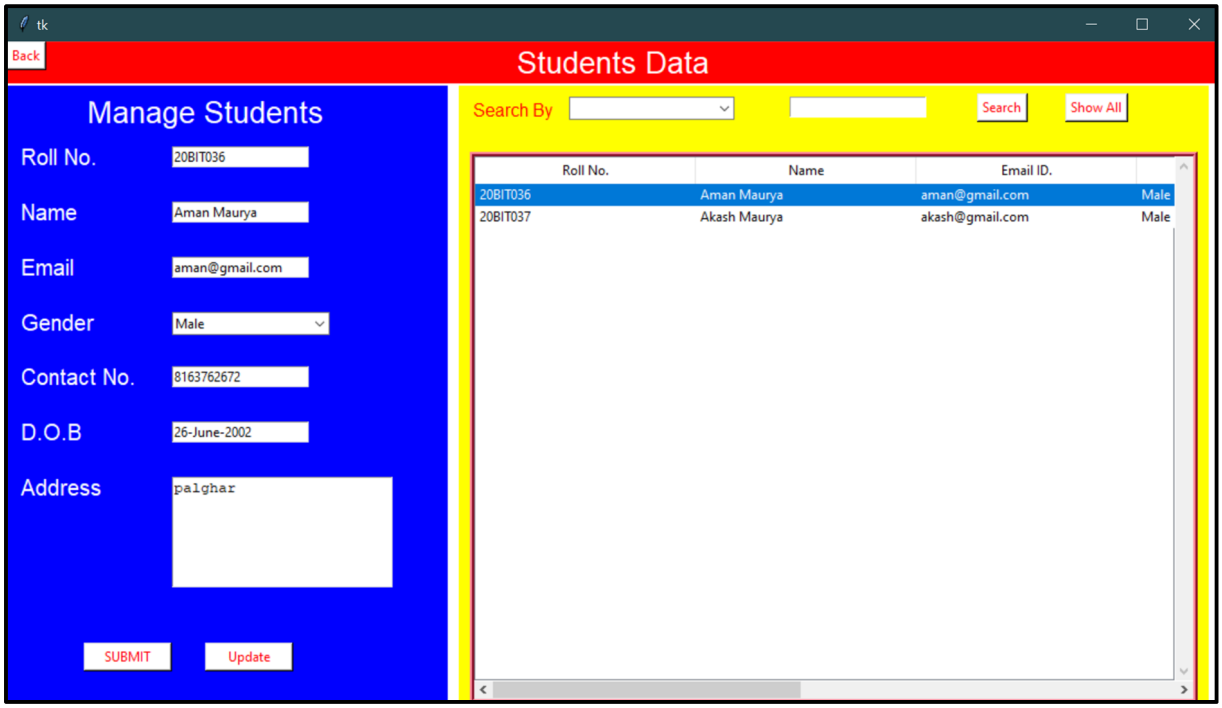
Registration page



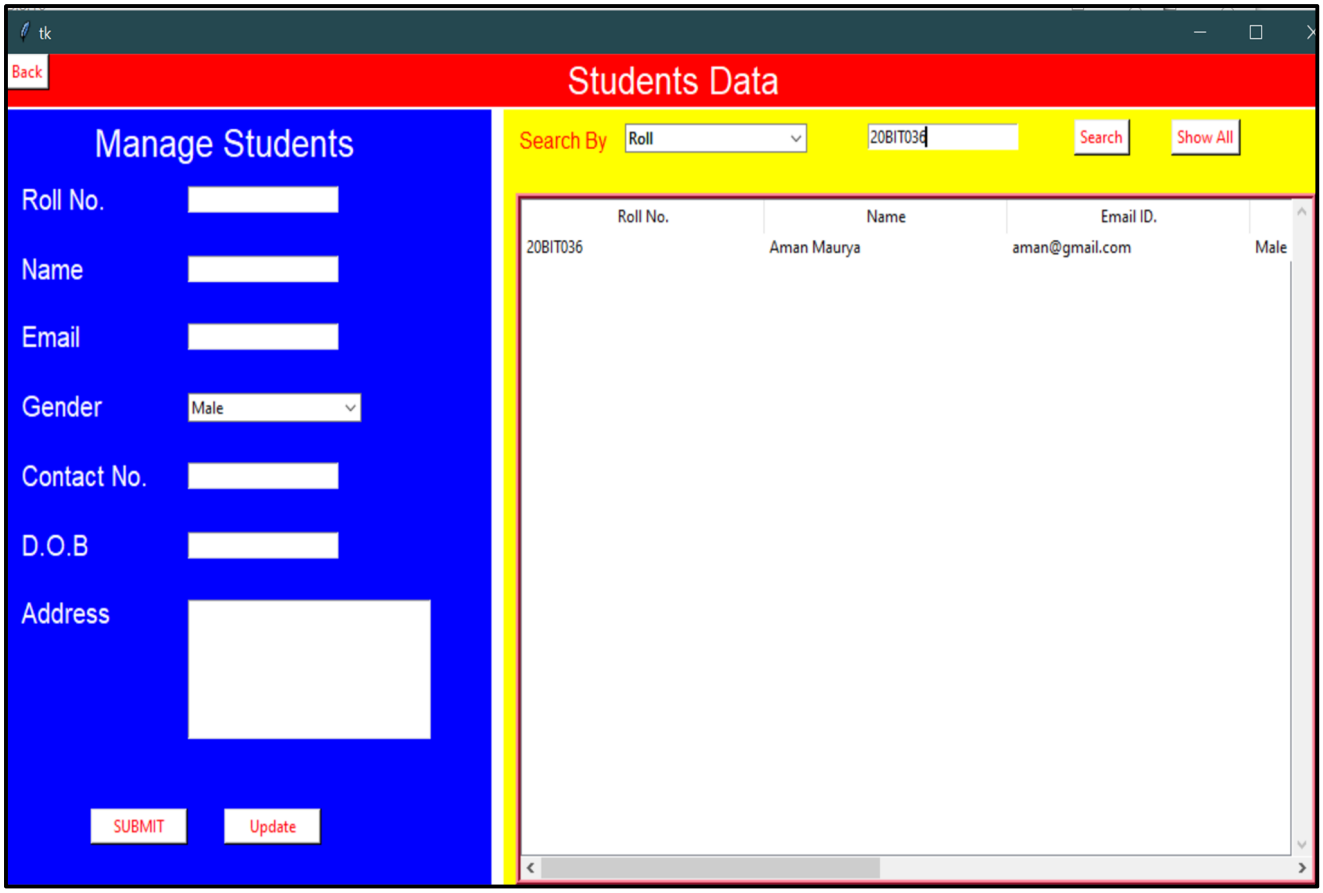
Dashboard page



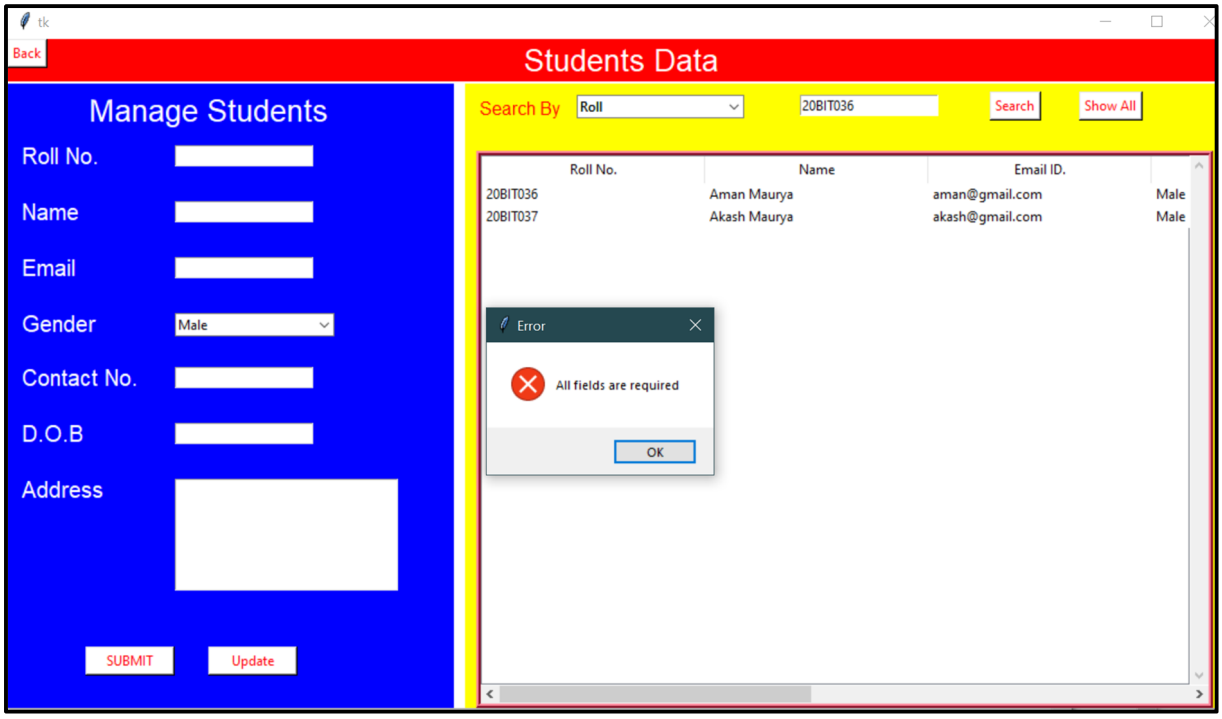
Student data: If you press on data it will automatically fill in textbox



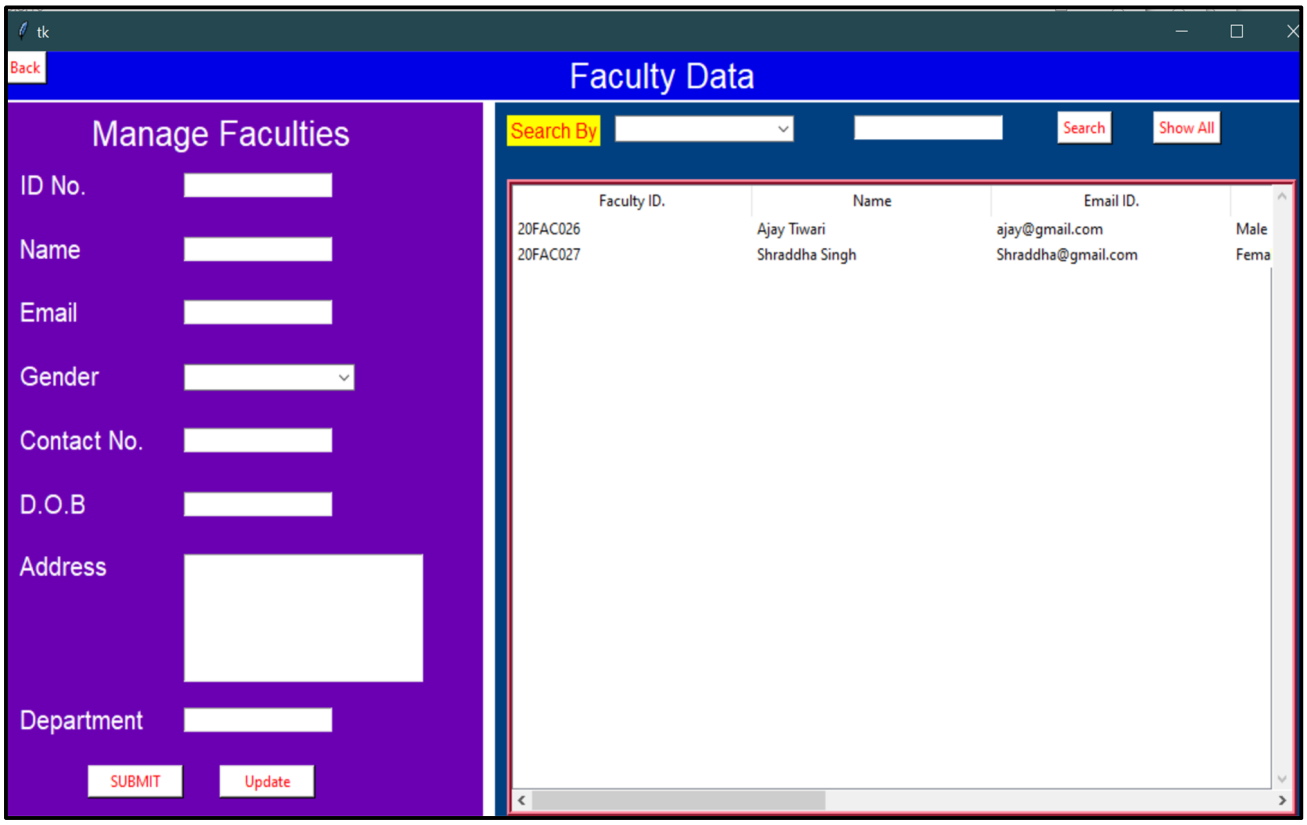
You can search by roll number name and contact of student



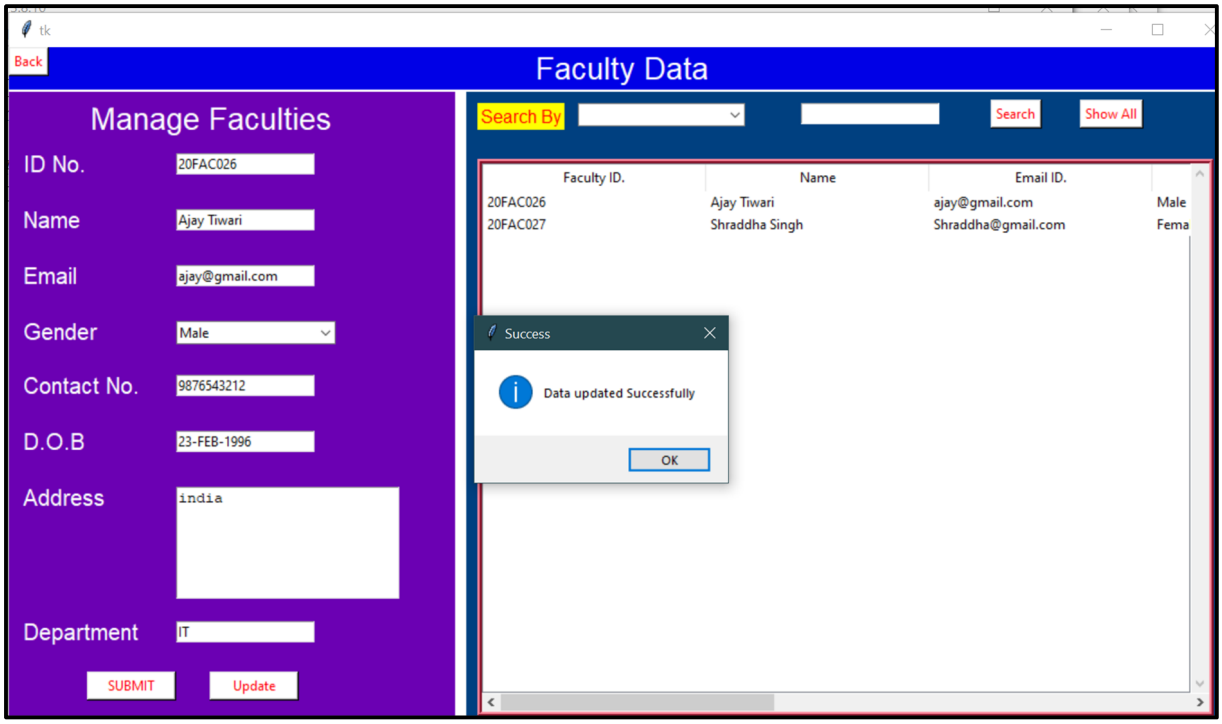
While entering the data if any field is empty it will show to error.



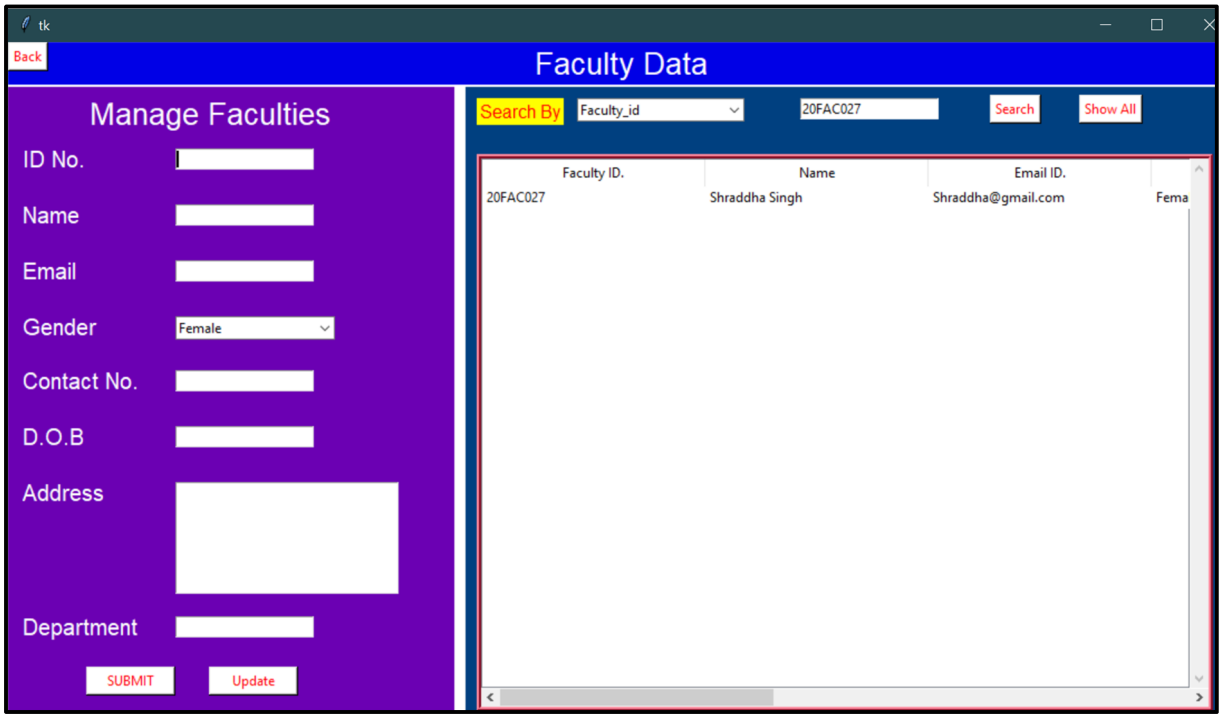
Faculty data



You can update the data of faculty



Here also you can search the data of faculty by there name, faculty_id and contact.

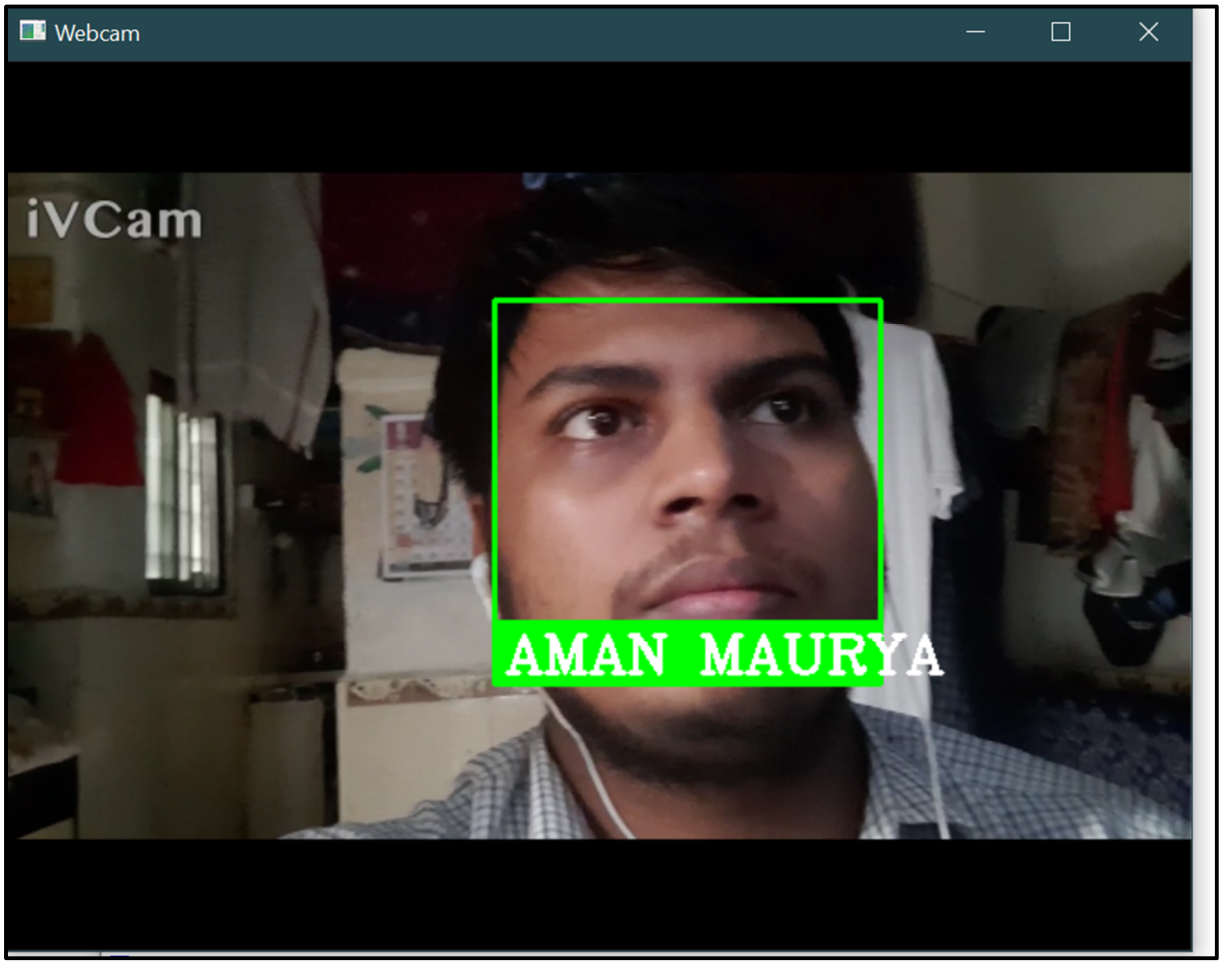


Attendance:

Before attendance taken.

	Name	Time
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		

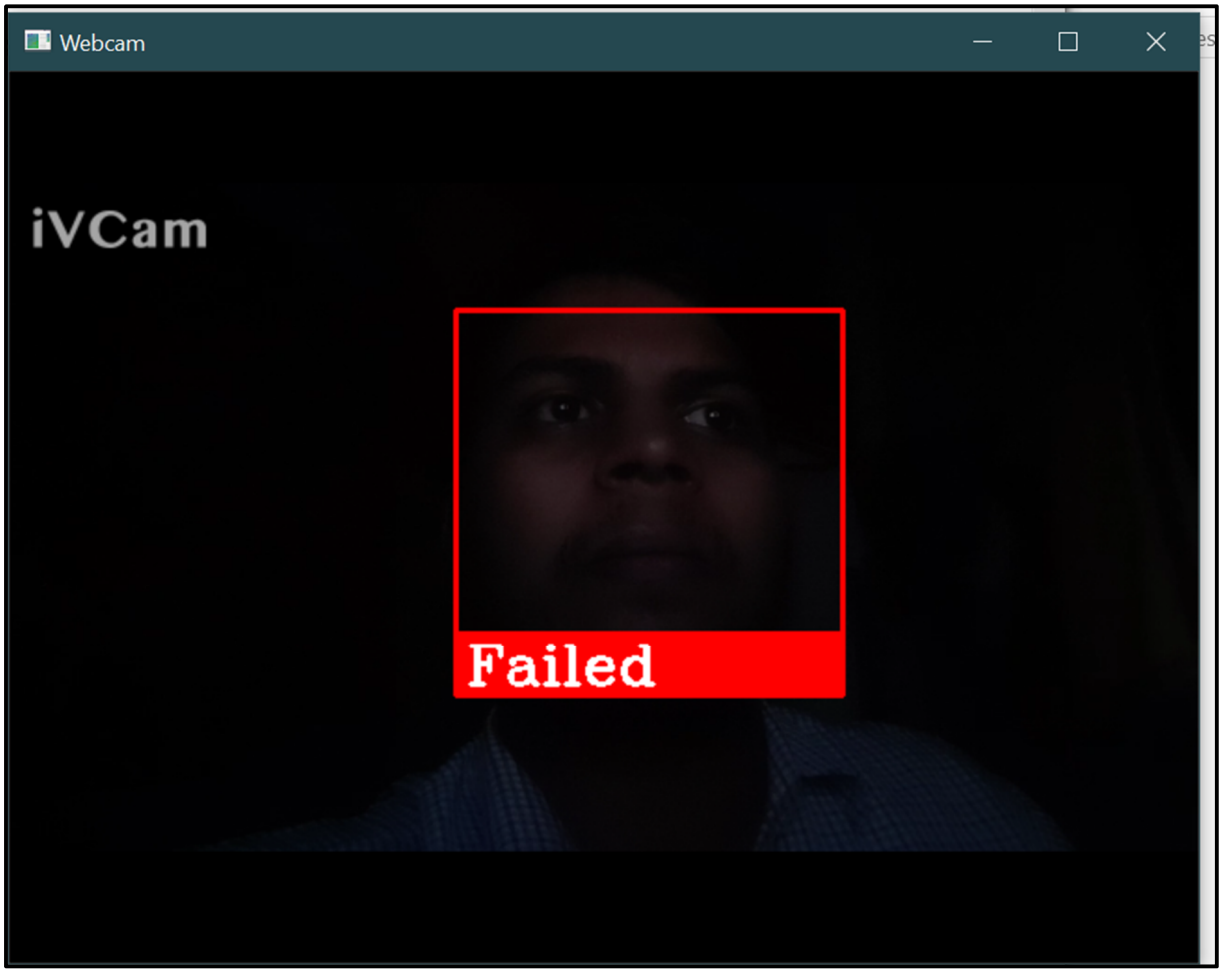
Recognizing the face of student.



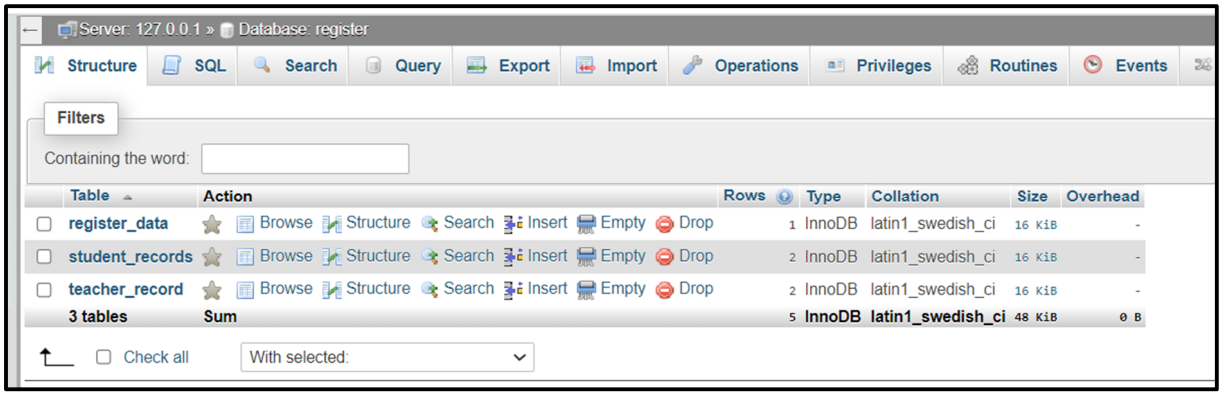
After recognizing the face the name and timing of student is inserted.

	A	B	C	D
1	Name	Time		
2				
3	AMAN MA	01:55:39		
4				
5				
6				
7				
8				

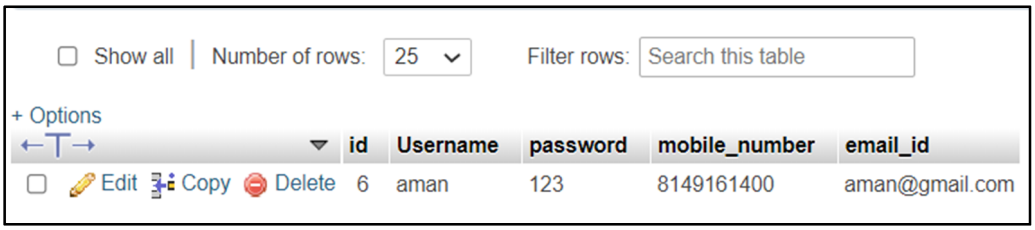
If the data of student is not present in our file then it will show you fail and attendance is not placed.



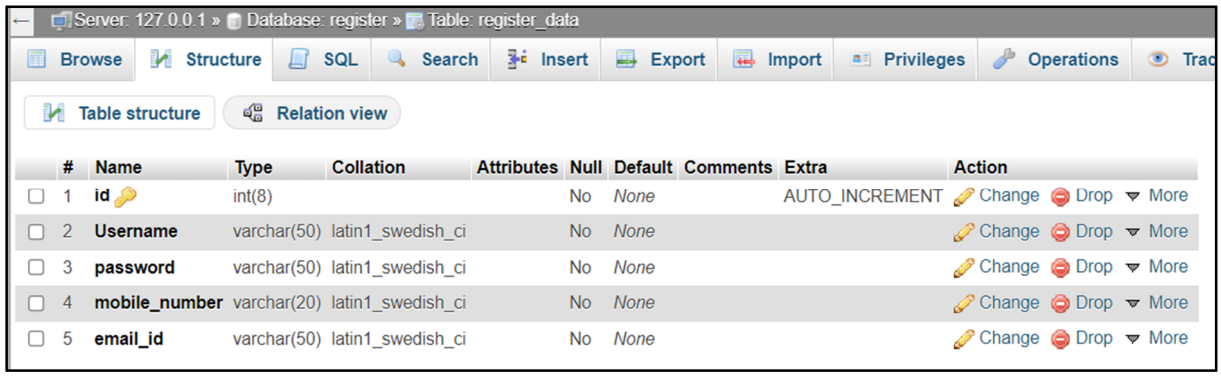
Structure of database.



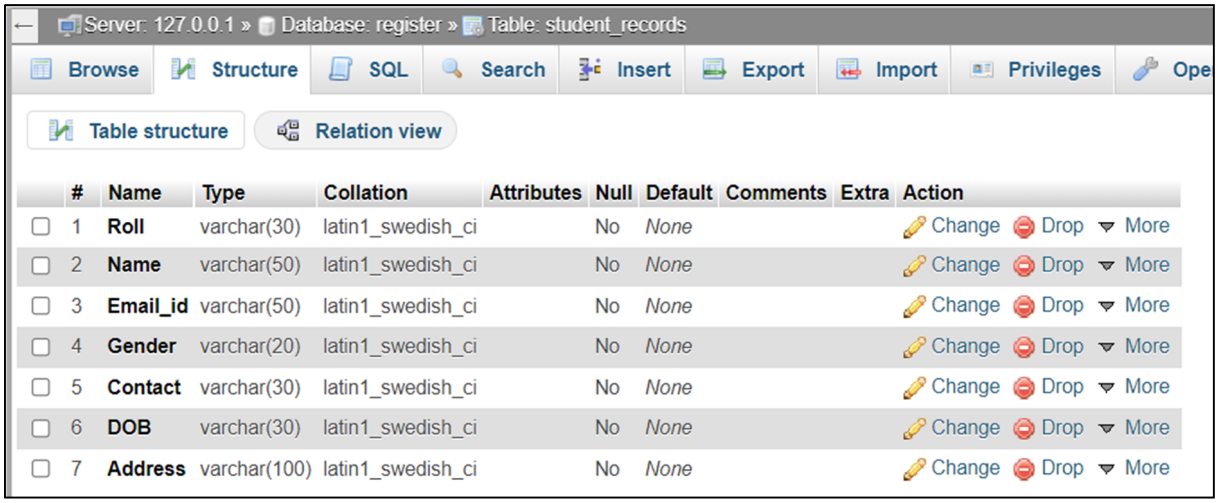
Data in register data table.



Structure of register table.

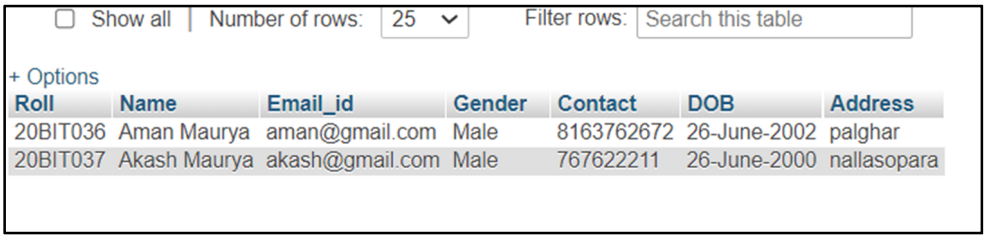


Structure of student records table.



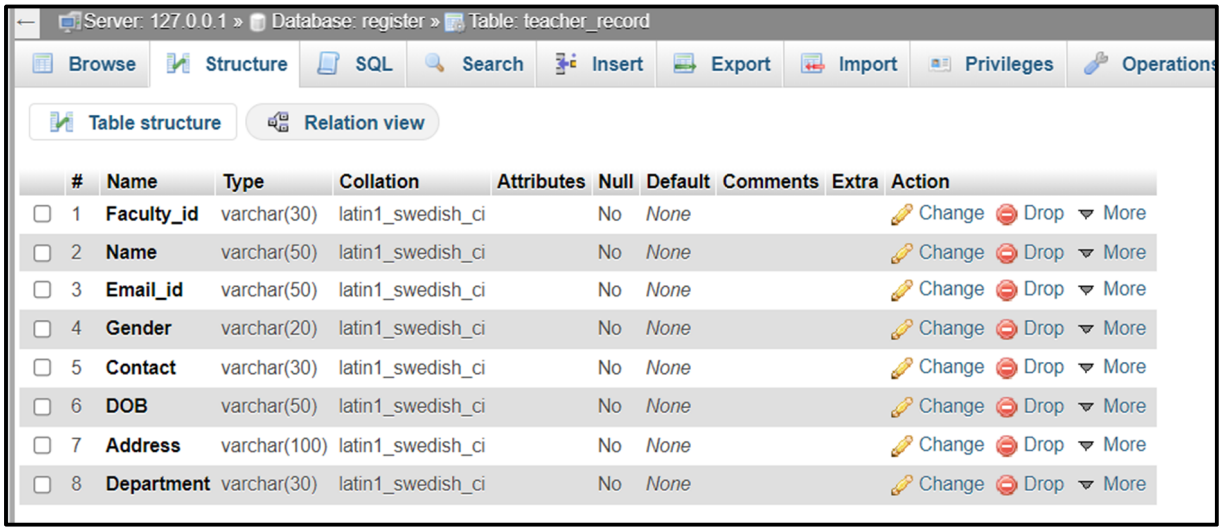
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Roll	varchar(30)	latin1_swedish_ci		No	None			Change Drop More
2	Name	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
3	Email_id	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
4	Gender	varchar(20)	latin1_swedish_ci		No	None			Change Drop More
5	Contact	varchar(30)	latin1_swedish_ci		No	None			Change Drop More
6	DOB	varchar(30)	latin1_swedish_ci		No	None			Change Drop More
7	Address	varchar(100)	latin1_swedish_ci		No	None			Change Drop More

Data in student record table.



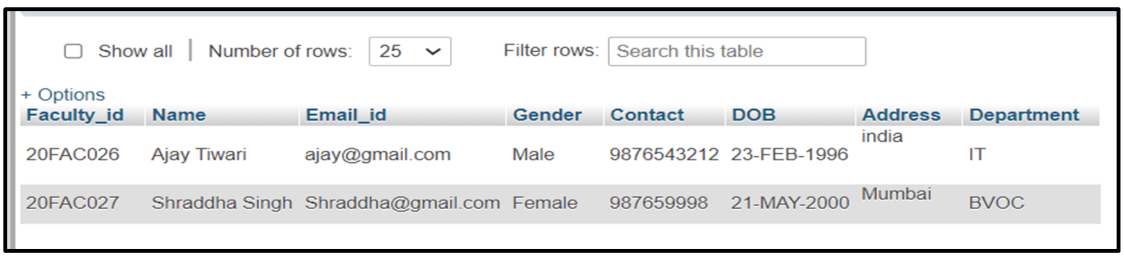
Roll	Name	Email_id	Gender	Contact	DOB	Address
20BIT036	Aman Maurya	aman@gmail.com	Male	8163762672	26-June-2002	palghar
20BIT037	Akash Maurya	akash@gmail.com	Male	767622211	26-June-2000	nallasopara

Structure of teacher record table



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Faculty_id	varchar(30)	latin1_swedish_ci		No	None			Change Drop More
2	Name	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
3	Email_id	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
4	Gender	varchar(20)	latin1_swedish_ci		No	None			Change Drop More
5	Contact	varchar(30)	latin1_swedish_ci		No	None			Change Drop More
6	DOB	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
7	Address	varchar(100)	latin1_swedish_ci		No	None			Change Drop More
8	Department	varchar(30)	latin1_swedish_ci		No	None			Change Drop More

Data in faculty records table.



Faculty_id	Name	Email_id	Gender	Contact	DOB	Address	Department
20FAC026	Ajay Tiwari	ajay@gmail.com	Male	9876543212	23-FEB-1996	india	IT
20FAC027	Shraddha Singh	Shraddha@gmail.com	Female	987659998	21-MAY-2000	Mumbai	BVOC