THE Choice SCHOOL
Bringing you the power of education
KNOWLEDGE ◆ CHARACTER ◆ HEALTH

COMPUTER SCIENCE PROJECT

School Management Software

Done by

VEER HEMAN KURUWA

AHILESH M

VEDANT SOMESH NAMBIAR

CLASS : XII A

TABLE OF CONTENTS

Acknowledgement

I would like to express my heartfelt gratitude and appreciation to everyone who supported me in completing this project.

First and foremost, I thank God Almighty for guiding me throughout the journey of this project. I am also deeply grateful to Ms. Rachel Ignatius, our Principal and Ms. Annie Preetha Gibu , our Vice Principal for providing all the necessary facilities and support that made this project possible. A special thanks goes to my Computer Science teacher, Ms. Maya Moneykumar, for all the support and guidance throughout the project.

Lastly, I would like to extend my sincere thanks to my parents and my group members, whose unwavering support and valuable input played an essential role in helping me complete this project.

Project Profile

A **Student Management System** is a sophisticated software solution designed to **streamline** administrative and academic processes within educational institutions. Utilizing **Python** for backend development, **Tkinter** for an intuitive graphical user interface, and **MySQL** for data management, the system integrates a range of functionalities to enhance efficiency and effectiveness.

Key Features:

Data Management: Enables the **addition**, **deletion**, **editing**, and **searching** of student and staff records, ensuring data accuracy.

Attendance Management: Tracks and **records** student attendance with reporting and analytics for monitoring.

Subject and Teacher Assignment: Facilitates the **assignment of subjects** to students and teachers to classes, optimizing curriculum management.

Database Management: Uses MySQL for efficient **data storage** and retrieval, supporting complex queries and real-time access.

Email Communication: Implements **SMTP** for teachers to email students directly from the system.

Python & MySQL Overview

Python is a high-level, versatile programming language known for its readability and **ease of use**. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python has a vast ecosystem of **libraries** and frameworks, making it **ideal** for various applications, from GUI's to data analysis.

MySQL is a popular open-source relational database management system (**RDBMS**). It uses Structured Query Language (**SQL**) for accessing and managing data. MySQL is widely used for applications and is known for its **reliability**, **performance**, and **scalability**. It allows users to create, read, update, and delete data efficiently, making it a core component of many backend systems.

Tkinter is the standard **GUI** (Graphical User Interface) toolkit for Python. It provides an easy way to create desktop applications with a user-friendly interface. Tkinter comes bundled with Python, making it readily **accessible** for developers. It allows the creation of windows, buttons, text fields, and other interactive elements, enabling users to interact with the application seamlessly.

Project Integration

In our project, we have used the **mysql-connector** module to connect Python to a MySQL database. This setup enables us to **perform database operations directly** from our Python application. Using Tkinter, we designed an intuitive user interface that allows users to interact with the data stored in MySQL.
This combination of Python, MySQL, and Tkinter facilitates the development of powerful applications with a robust backend and a user-friendly frontend, making it suitable for various projects.

Hardware & Software Requirements

Software:

Front End: Python 3.11
Python with libraries and Connectivity to MySQL
Back end: MySQL
Operating System : Windows 11

Hardware:

Device Name : Surface Pro 7
RAM : 16 GB
Processor: Intel Core i7

Methodology

1. **Project Planning**

- **Objective:** To develop a **School Management System** where users can perform operations like adding, updating , deleting and viewing student and staff details, managing user accounts etc.
- **Scope:** The system will allow teachers, admin staff, and parents to interact with it via a Python Tkinter-based interface, and all data will be stored in a **MySQL** database.
- **Technologies:**
  - **Front End:** Tkinter
  - **Backend:** MySQL
  - **Libraries:** tkinter , smtplib , base64 , PIL , email , os , datetime , mysql.connector , tkcalendar

2. **System Design**

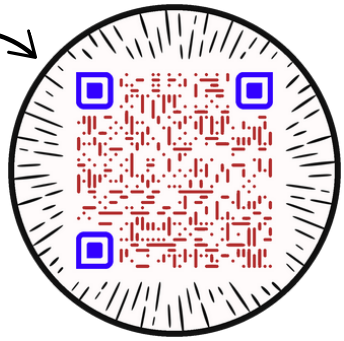
- **Database Design:**
  - **credentials:** stores the username and login passwords for all the users.
  - **students:** stores personal details of all the students
  - **staff:** stores personal details of all the staff
  - **std_attendance:** stores the attendance records of all the students
  - **profile:** stores the images of all the users.
- **Frontend Design:**
  - We have used open source tkinter design code to design the elements of the UI.
  - **Login Page:** A simple user authentication page that accepts the username and password
  - **Dashboard Windows:** Custom Dashboard that links all of the different windows to the user using buttons.
  - **Functional Requirements:**
    - Add, Update, Delete and Search users
    - Student / Teacher Data Entry
    - Searching Records
    - Email Students
    - Mark and View Attendance
    - Update Contact Information and Photo

SQL Table Structures

```
mysql> desc credentials;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| name     | varchar(30) | YES  |     | NULL    |       |
| username | varchar(10) | YES  |     | NULL    |       |
| password | varchar(10) | YES  |     | NULL    |       |
| role     | varchar(10) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> desc profile;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| id      | varchar(25) | NO   | PRI | NULL    |       |
| picture | longblob    | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> desc staff;
+------------------+--------------+------+-----+---------+-------+
| Field            | Type         | Null | Key | Default | Extra |
+------------------+--------------+------+-----+---------+-------+
| STAFFID          | varchar(5)   | NO   | PRI | NULL    |       |
| NAME             | varchar(100) | YES  |     | NULL    |       |
| SUBJECT_TAUGHT   | varchar(50)  | YES  |     | NULL    |       |
| CLASS1           | varchar(20)  | YES  |     | NULL    |       |
| CLASS2           | varchar(20)  | YES  |     | NULL    |       |
| CLASS3           | varchar(20)  | YES  |     | NULL    |       |
| EMAIL            | varchar(100) | YES  |     | NULL    |       |
| SPOUSE_NAME      | varchar(100) | YES  |     | NULL    |       |
| GENDER           | enum('M','F') | YES  |     | NULL    |       |
| EMERGENCY_CONTACT | varchar(50)  | YES  |     | NULL    |       |
| ADDRESS          | varchar(255) | YES  |     | NULL    |       |
| DOB              | date         | YES  |     | NULL    |       |
| DATE_OF_JOINING  | date         | YES  |     | NULL    |       |
+------------------+--------------+------+-----+---------+-------+
13 rows in set (0.00 sec)

mysql> desc students;
+------------------+--------------+------+-----+---------+-------+
| Field            | Type         | Null | Key | Default | Extra |
+------------------+--------------+------+-----+---------+-------+
| admno            | varchar(15)  | NO   | PRI | NULL    |       |
| first_name       | varchar(50)  | YES  |     | NULL    |       |
| last_name        | varchar(50)  | YES  |     | NULL    |       |
| dob              | date         | YES  |     | NULL    |       |
| gender           | enum('M','F') | YES  |     | NULL    |       |
| address          | varchar(255) | YES  |     | NULL    |       |
| emergency_contact | varchar(50)  | YES  |     | NULL    |       |
| subject1         | varchar(50)  | YES  |     | NULL    |       |
| subject2         | varchar(50)  | YES  |     | NULL    |       |
| subject3         | varchar(50)  | YES  |     | NULL    |       |
| subject4         | varchar(50)  | YES  |     | NULL    |       |
| subject5         | varchar(50)  | YES  |     | NULL    |       |
| class            | varchar(20)  | YES  |     | NULL    |       |
| email            | varchar(100) | YES  |     | NULL    |       |
+------------------+--------------+------+-----+---------+-------+
14 rows in set (0.00 sec)

mysql> desc std_attendance;
+---------+-------------------+------+-----+---------+-------+
| Field   | Type              | Null | Key | Default | Extra |
+---------+-------------------+------+-----+---------+-------+
| admno   | varchar(15)       | NO   |     | NULL    |       |
| adate   | date              | NO   |     | NULL    |       |
| status  | enum('P','A','L') | NO   |     | NULL    |       |
| subject | varchar(25)       | YES  |     | NULL    |       |
+---------+-------------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Source Code

View on Github



```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
from tkinter import PhotoImage
from tkinter import filedialog
import smtplib
import base64
from PIL import Image, ImageTk
from PIL import ImageDraw
import io
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders
import os
from datetime import datetime
from tkinter import messagebox
import mysql.connector
from tkcalendar import Calendar
con=mysql.connector.connect(host='localhost',user='root',password='********',database='cscfyp')
cur=con.cursor()

def log_out():
    global root, dashboard_window
    if dashboard_window:
        dashboard_window.destroy()
    root = tk.Tk()
    root.geometry('925x500+300+200')
    root.tk.call('source', 'forest-dark.tcl')
    ttk.Style().theme_use('forest-dark')

    img = PhotoImage(file='login.PNG')
    img_label = tk.Label(root, image=img)
    img_label.place(x=50, y=50)

    frame = ttk.Frame(root, width=350, height=350)
    frame.place(x=625, y=60)

    header2 = ttk.Label(frame, text="Sign In", anchor=tk.CENTER, font=("Open Sans", 19, "bold"))
    header2.pack(pady=10)
```

Source Code

```
lab1 = ttk.Label(frame, text="Username:", anchor=tk.CENTER, font=("Segoe UI", 13, "bold"))
global username
username = ttk.Entry(frame)
lab1.pack(pady=10)
username.pack(pady=2)

lab2 = ttk.Label(frame, text="Password:", anchor=tk.CENTER, font=("Segoe UI", 13, "bold"))
global pwd
pwd = ttk.Entry(frame, show="*")
lab2.pack(pady=10)
pwd.pack(pady=2)

button = ttk.Button(frame, text='Login', style='Accent.TButton', command=handle_login)
button.pack(pady=10)
root.mainloop()

file_name=[]
def openFile():
 global file_name
 file_name=[]
 '''file_name =filedialog.askopenfilename().replace("\\" , "\\\\")'''
 temp=filedialog.askopenfilenames()
 for i in temp:
 file_name.append(i.replace("\\" , "\\\\"))

def handle_login():
    global username_text
    username_text = username.get()
    password_text = pwd.get()
    cur.execute(f"select * from credentials where username='{username_text}';")
    global creds
    creds=cur.fetchone()
    if creds[2]==password_text:
        global role
        role =creds[3]
        open_dashboard(role)
    else:
        messagebox.showerror("Login Error", "Invalid username or password")
```

Source Code

```
def email_student():
    smt_port = 587
    smtp_server = 'smtp.gmail.com'
    email_from = 'veerhk2007@gmail.com'
    pswd = '**** **** **** ****'

    email_window = tk.Tk()
    email_window.geometry('400x400')
    email_window.title("Send Email")
    email_window.tk.call('source', 'forest-dark.tcl')
    ttk.Style().theme_use('forest-dark')

    cur.execute(f"SELECT NAME FROM STAFF WHERE STAFFID='{username_text}';")
    name=cur.fetchone()[0]
    subj = f'New Announcement from {name}'

    inpembody = ttk.Label(email_window, text="Body:", anchor=tk.CENTER, font=("Segoe UI", 13, "bold"))
    body = ttk.Entry(email_window)
    inpembody.pack(pady=20)
    body.pack(pady=20)

    ttk.Label(email_window, text="Select Class:").pack(pady=5)
    selected_class = ttk.Combobox(email_window)

    cur.execute(f"SELECT class1, class2, class3 FROM staff WHERE staffid='{username_text}'")
    classes = cur.fetchone()
    class_names = [c for c in classes if c]
    selected_class['values'] = class_names
    selected_class.pack(pady=10)

    attach = ttk.Button(email_window, text='Add Attachments', style='Accent.TButton', command=openFile)
    attach.pack()
def send_emails():
        email_list = []
        selected = selected_class.get()

        cur.execute(f"SELECT email FROM students WHERE class='{selected}'")
        emails = cur.fetchall()
        email_list.extend([email[0] for email in emails])  # Add emails to email_list

        body_text = body.get()
```

Source Code

```
styl = f"""
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <style>
 body {{
 display: grid;
 place-items: center;
 background: #1a191f;
 }}
 h1 {{
 margin: 0;
 font-family: "Mulish", sans-serif;
 font-weight: 1000;
 font-size: 32px;
 color: #deb460;
 }}
 h2 {{
 font-family: "Roboto Condensed", sans-serif;
 color: #deb460;
 }}
 </style>
</head>
<body>
 <h1>New Announcement from your Teacher</h1>
 <h2>{body_text}</h2>
</body>
</html>
"""

      for person in email_list:
         msg = MIMEMultipart()
         msg['From'] = email_from
         msg['To'] = person
         msg['Subject'] = subj
         msg.attach(MIMEText(styl, 'html'))
```

Source Code

```
if file_name!=[]:
 for i in file_name:
 try:
 file_base_name = os.path.basename(i)
 with open(i, 'rb') as attachment:
 p = MIMEBase('application', 'octet-stream')
 p.set_payload((attachment).read())
 encoders.encode_base64(p)
 p.add_header('Content-Disposition', f"attachment; filename={file_base_name}")
 msg.attach(p)
 except Exception as e:
 messagebox.showerror("Error", f"Failed to attach file: {file_base_name}\n{str(e)}")

 try:
 with smtplib.SMTP(smtp_server, smt_port) as email_server:
 email_server.starttls()
 email_server.login(email_from, pswd)
 email_server.sendmail(email_from, person, msg.as_string())
 print(msg)
 except Exception as e:
 messagebox.showerror("Error", f"Failed to send email to {person}\n{str(e)}")

 messagebox.showinfo("Success", "Emails Successfully Sent!")
 email_window.destroy()

 send_button = ttk.Button(email_window, text='Send Email', style='Accent.TButton', command=send_emails)
 send_button.pack(pady=20)
```

Source Code

```
def update_info_parent():
    try:

        update_window = tk.Toplevel()
        update_window.title("Update Information")
        update_window.geometry("400x400")

        admno = username_text
        cur.execute("SELECT first_name, last_name, dob, gender, address, emergency_contact, email,class
FROM students WHERE admno=%s", (admno,))
        current_info = cur.fetchone()

        ttk.Label(update_window, text="First Name:").pack(pady=5)
        first_name_entry = ttk.Entry(update_window)
        first_name_entry.insert(0, current_info[0] if current_info else "")
        first_name_entry.pack()

        ttk.Label(update_window, text="Last Name:").pack(pady=5)
        last_name_entry = ttk.Entry(update_window)
        last_name_entry.insert(0, current_info[1] if current_info else "")
        last_name_entry.pack()

        ttk.Label(update_window, text="Date of Birth:").pack(pady=5)
        dob_entry = ttk.Entry(update_window)
        dob_entry.insert(0, current_info[2] if current_info else "")
        dob_entry.pack()

        ttk.Label(update_window, text="Gender:").pack(pady=5)
        gender_entry = ttk.Combobox(update_window, values=["M", "F"])
        gender_entry.set(current_info[3] if current_info else "")
        gender_entry.pack()

        ttk.Label(update_window, text="Address:").pack(pady=5)
        address_entry = ttk.Entry(update_window)
        address_entry.insert(0, current_info[4] if current_info else "")
        address_entry.pack()

        ttk.Label(update_window, text="Emergency Contact:").pack(pady=5)
        emergency_contact_entry = ttk.Entry(update_window)
        emergency_contact_entry.insert(0, current_info[5] if current_info else "")
        emergency_contact_entry.pack()
```

Source Code

```
ttk.Label(update_window, text="Email:").pack(pady=5)
 email_entry = ttk.Entry(update_window)
 email_entry.insert(0, current_info[6] if current_info else "")
 email_entry.pack()

 def submit_update():
 new_info = (
 first_name_entry.get(),
 last_name_entry.get(),
 dob_entry.get(),
 gender_entry.get(),
 address_entry.get(),
 emergency_contact_entry.get(),
 email_entry.get(),

 )
 if current_info is None:
 messagebox.showinfo("Failed", "Please Ask Admin to Register Student!")

 else:
 cur.execute("""
 UPDATE students
 SET first_name=%s, last_name=%s, dob=%s, gender=%s, address=%s, emergency_contact=%s, email=%s
 WHERE admno=%s
 """, new_info + (admno,))
 con.commit()
 messagebox.showinfo("Success", "Information updated successfully!")
 update_window.destroy()

 ttk.Button(update_window, text="Update", command=submit_update).pack(pady=20)

 except Exception as e:
 messagebox.showerror("Error", f"An error occurred: {e}")

def update_info_teacher():
   try:
      update_window = tk.Toplevel()
      update_window.title("Update Information")
      update_window.geometry("400x400")
```

Source Code

```
staff_id = username_text
cur.execute("SELECT name, dob, gender, address, emergency_contact, email FROM staff WHERE
STAFFID=%s", (staff_id,))
current_info = cur.fetchone()

ttk.Label(update_window, text="Name:").pack(pady=5)
name_entry = ttk.Entry(update_window)
name_entry.insert(0, current_info[0] if current_info else "")
name_entry.pack()
    ttk.Label(update_window, text="Date of Birth:").pack(pady=5)
    dob_entry = ttk.Entry(update_window)
    dob_entry.insert(0, current_info[1] if current_info else "")
    dob_entry.pack()

    ttk.Label(update_window, text="Gender:").pack(pady=5)
    gender_entry = ttk.Combobox(update_window, values=["M", "F"])
    gender_entry.set(current_info[2] if current_info else "")
    gender_entry.pack()

    ttk.Label(update_window, text="Address:").pack(pady=5)
    address_entry = ttk.Entry(update_window)
    address_entry.insert(0, current_info[3] if current_info else "")
    address_entry.pack()

    ttk.Label(update_window, text="Emergency Contact:").pack(pady=5)
    emergency_contact_entry = ttk.Entry(update_window)
    emergency_contact_entry.insert(0, current_info[4] if current_info else "")
    emergency_contact_entry.pack()

    ttk.Label(update_window, text="Email:").pack(pady=5)
    email_entry = ttk.Entry(update_window)
    email_entry.insert(0, current_info[5] if current_info else "")
    email_entry.pack()

    def submit_update():
      new_info = (
        name_entry.get(),
        dob_entry.get(),
        gender_entry.get(),
        address_entry.get(),
        emergency_contact_entry.get(),
        email_entry.get(),
        staff_id
      )
```

Source Code

```
 if current_info is None:
messagebox.show("Error","Kindly Ask Admin to Register Teacher")
else:
cur.execute("""
UPDATE staff
SET NAME=%s, DOB=%s, GENDER=%s, ADDRESS=%s, EMERGENCY_CONTACT=%s, EMAIL=%s
WHERE STAFFID=%s
""", new_info)
con.commit()
messagebox.showinfo("Success", "Information updated successfully!")
update_window.destroy()

ttk.Button(update_window, text="Update", command=submit_update).pack(pady=20)

except Exception as e:
messagebox.showerror("Error", f"An error occurred: {e}")

def mark_attendance():
  cur.execute(f"SELECT class1, class2, class3 FROM staff WHERE staffid='{username_text}'")
  classes = cur.fetchone()

  students_by_class = {}
  for class_name in classes:
    cur.execute(f"SELECT admno, first_name FROM students WHERE class='{class_name}'")
    students_by_class[class_name] = cur.fetchall()

  cur.execute(f"SELECT subject_taught from staff where staffid='{username_text}';")
  subj=cur.fetchone()[0]
  attendance_window = tk.Toplevel()
  attendance_window.title("Mark Attendance")
  attendance_window.geometry("900x1000")

  ttk.Label(attendance_window, text="Mark Attendance", font=("Segoe UI", 20, "bold")).pack(pady=10)

  ttk.Label(attendance_window, text="Select Date:").pack(pady=5)
  cal = Calendar(attendance_window, selectmode='day', year=datetime.now().year,
           month=datetime.now().month, day=datetime.now().day)
  cal.pack(pady=10)

  attendance_vars = {}
```

Source Code

```
    for class_name, students in students_by_class.items():

        ttk.Label(attendance_window, text=f"Class: {class_name}", font=("Segoe UI", 14, "bold")).pack(pady=5)

        for admno, name in students:
            var = tk.StringVar(value='P')
            attendance_vars[admno] = var
            ttk.Label(attendance_window, text=f"{name} ({admno})").pack(anchor='w')
            ttk.Combobox(attendance_window, textvariable=var, values=['P', 'A', 'L'],
state='readonly').pack(anchor='w')

      def submit_attendance():
        attendance_date = cal.get_date()

        month, day, year = map(int, attendance_date.split('/'))
        formatted_date = f"{year}-{month:02d}-{day:02d}"

        for admno, status_var in attendance_vars.items():
            status_value = status_var.get()
            cur.execute("INSERT INTO std_attendance (admno, adate, status,subject) VALUES (%s, %s, %s,%s)",
                        (admno, formatted_date, status_value,subj))
        con.commit()
        messagebox.showinfo("Success", "Attendance marked successfully!")
        attendance_window.destroy()

    ttk.Button(attendance_window, text="Submit", command=submit_attendance).pack(pady=20)
def view_attendance_parent():

    attendance_window = tk.Toplevel()
    attendance_window.title("View Attendance")
    attendance_window.geometry("600x400")

    ttk.Label(attendance_window, text="Attendance Records", font=("Segoe UI", 16)).pack(pady=10)


    admno = username_text


    cur.execute(f"SELECT adate, status , subject FROM std_attendance WHERE admno='{admno}'")
    attendance_data = cur.fetchall()
```

Source Code

```
    total_classes = len(attendance_data)
    attended_classes = 0
    for i in attendance_data:
        if i[1]=='P' or i[1]=='L':
            attended_classes+=1
    if total_classes:
        attendance_percentage = (attended_classes / total_classes * 100)
    else:
        attendance_percentage=0


    columns = ("Date", "Status","Subject")
    treeview = ttk.Treeview(attendance_window, columns=columns, show='headings')
    for col in columns:
        treeview.heading(col, text=col)
    treeview.pack(expand=True, fill="both", padx=10, pady=10)

    for date, status,subj in attendance_data:
        treeview.insert("", "end", values=(date, status, subj))


    ttk.Label(attendance_window, text=f"Attendance Percentage: {attendance_percentage:.2f}%", font=
("Segoe UI", 14)).pack(pady=10)


    close_button = ttk.Button(attendance_window, text="Close", command=attendance_window.destroy)
    close_button.pack(pady=10)

def manage_users():
    global useredit_window
    useredit_window = tk.Toplevel()
    useredit_window.geometry('600x400')
    useredit_window.title(f"{role.capitalize()} Dashboard")


    header = ttk.Label(useredit_window, text="Manage Users", font=("Segoe UI", 20, "bold"))
    header.pack(pady=10)
    treeview_frame = ttk.Frame(useredit_window)
    treeview_frame.pack(expand=True, fill="both")

    treeview = ttk.Treeview(treeview_frame, selectmode="extended", columns=("Username", "Role"),
height=15)
    treeview.pack(side="left", expand=True, fill="both")
```

Source Code

```
    tree_scroll = ttk.Scrollbar(treeview_frame, orient="vertical", command=treeview.yview)
    treeview.config(yscrollcommand=tree_scroll.set)
    tree_scroll.pack(side="right", fill="y")


    treeview.heading("#0", text="Name", anchor="center")
    treeview.heading("Username", text="Username", anchor="center")
    treeview.heading("Role", text="Role", anchor="center")

    load_users(treeview)

    add_button = ttk.Button(useredit_window, text="Add User", command=lambda: add_user(treeview))
    add_button.pack(pady=10)

    delete_button = ttk.Button(useredit_window, text="Delete User", command=lambda:
delete_user(treeview))
    delete_button.pack(pady=10)

    modify_button = ttk.Button(useredit_window, text="Modify User", command=lambda:
modify_user(treeview))
    modify_button.pack(pady=10)

    close_button = ttk.Button(useredit_window, text="Close", command=useredit_window.destroy)
    close_button.pack(pady=10)

def load_users(treeview):
    """Load users from the database and display them in the Treeview."""
    treeview.delete(*treeview.get_children())
    cur.execute("SELECT name, username, password, role FROM credentials;")
    for row in cur.fetchall():
        treeview.insert("", "end", iid=row[0], text=row[0], values=(row[1], row[3]))
def add_user(treeview):
    def submit():
        nname=name_entry.get()
        nusername = username_entry.get()
        nrole = role_entry.get()
        npassword = password_entry.get()
        if nusername and nrole and npassword:
            cur.execute("INSERT INTO credentials (name,username, role, password) VALUES (%s, %s, %s,%s)",
(nname,nusername, nrole, npassword))
            con.commit()
```

Source Code

```
        if nrole=='PARENT':
            cur.execute(f"INSERT INTO students (admno) value('{nusername}');")
            con.commit()
        else:
            cur.execute(f"INSERT INTO staff (staffid) value('{nusername}');")
            con.commit()
        load_users(treeview)
        add_window.destroy()
    add_window = tk.Toplevel(useredit_window)
    add_window.title("Add User")

    ttk.Label(add_window, text="Name:").pack()
    name_entry = ttk.Entry(add_window)
    name_entry.pack()
    ttk.Label(add_window, text="Username:").pack()
    username_entry = ttk.Entry(add_window)
    username_entry.pack()

    ttk.Label(add_window, text="Role:").pack()
    role_entry = ttk.Entry(add_window)
    role_entry.pack()

    ttk.Label(add_window, text="Password:").pack()
    password_entry = ttk.Entry(add_window, show="*")
    password_entry.pack()

    ttk.Button(add_window, text="Submit", command=submit).pack()
def delete_user(treeview):
    """Delete selected user from the database."""
    selected_item = treeview.selection()
    if not selected_item:
        messagebox.showwarning("Selection Error", "Please select a user to delete.")
        return

    user_id = selected_item[0]
    cur.execute(f"SELECT ROLE FROM credentials WHERE name = %s;", (user_id,))
    role = cur.fetchone()
    cur.execute(f"SELECT USERNAME FROM CREDENTIALS WHERE name='{user_id}'")
    uname=cur.fetchone()
    if role and role[0] == 'PARENT':
        cur.execute(f"DELETE FROM students WHERE admno = '{uname[0]}';")
        con.commit()
```

Source Code

```
    else:
        cur.execute(f"DELETE FROM staff WHERE staffid = '{uname[0]}';")
        con.commit()


  if uname:
    cur.execute(f"DELETE FROM credentials WHERE username = '{uname[0]}';")
    con.commit()
    load_users(treeview)


  else:
    messagebox.showwarning("Error", "User not found.")
def modify_user(treeview):
  """Modify selected user information."""
  selected_item = treeview.selection()
  if not selected_item:
    messagebox.showwarning("Selection Error", "Please select a user to modify.")
    return

  user_id = selected_item[0]
  cur.execute("SELECT name, username, role, password FROM credentials WHERE name = %s;", (user_id,))
  user_data = cur.fetchone()

  if user_data is None:
    messagebox.showwarning("Error", "User data not found.")
    return

  def submit():
    new_name = name_entry.get()
    new_username = username_entry.get()
    new_role = role_entry.get()
    new_password = password_entry.get()

    if new_username and new_role:
      cur.execute(
        "UPDATE credentials SET name = %s, username = %s, role = %s, password = %s WHERE name = %s;",
        (new_name, new_username, new_role, new_password, user_id)
      )
      con.commit()
      load_users(treeview)
      modify_window.destroy()
```

Source Code

```
    modify_window = tk.Toplevel(useredit_window)
    modify_window.title("Modify User")

    ttk.Label(modify_window, text="Name:").pack()
    name_entry = ttk.Entry(modify_window)
    name_entry.insert(0, user_data[0])
    name_entry.pack()

    ttk.Label(modify_window, text="Username:").pack()
    username_entry = ttk.Entry(modify_window)
    username_entry.insert(0, user_data[1])
    username_entry.pack()

    ttk.Label(modify_window, text="Role:").pack()
    role_entry = ttk.Entry(modify_window)
    role_entry.insert(0, user_data[2])
    role_entry.pack()

    ttk.Label(modify_window, text="New Password:").pack()
    password_entry = ttk.Entry(modify_window, show="*")
    password_entry.insert(0, user_data[3])
    password_entry.pack()

    ttk.Button(modify_window, text="Submit", command=submit).pack()
def register_student():
    def submit_student():
        admno = entry_admno.get()
        first_name = entry_first_name.get()
        last_name = entry_last_name.get()
        dob = entry_dob.get()
        gender = var_gender.get()
        address = entry_address.get()
        emergency_contact = entry_emergency_contact.get()
        subjects = [entry_subject1.get(), entry_subject2.get(), entry_subject3.get(),
                    entry_subject4.get(), entry_subject5.get()]
        student_class = entry_class.get()
        email = entry_email.get()
```

Source Code

```
        sql = f"""UPDATE students
            SET first_name='{first_name}', last_name='{last_name}', dob='{dob}',
                gender='{gender}', address='{address}', emergency_contact='{emergency_contact}',
                subject1='{subjects[0]}', subject2='{subjects[1]}', subject3='{subjects[2]}',
                subject4='{subjects[3]}', subject5='{subjects[4]}',
                class='{student_class}', email='{email}'
            WHERE admno='{admno}'"""
        cur.execute(sql)
        con.commit()
        messagebox.showinfo("Success", "Student Registered Successfully !")
        student_window.destroy()

    student_window = tk.Toplevel(dashboard_window)
    student_window.title("Student Registration")
    student_window.geometry('700x500')

    ttk.Label(student_window, text="Student Registration").grid(row=0, columnspan=2)

    ttk.Label(student_window, text="Admission No:").grid(row=1, column=0)
    entry_admno = ttk.Entry(student_window)
    entry_admno.grid(row=1, column=1)

    ttk.Label(student_window, text="First Name:").grid(row=2, column=0)
    entry_first_name = ttk.Entry(student_window)
    entry_first_name.grid(row=2, column=1)

    ttk.Label(student_window, text="Last Name:").grid(row=3, column=0)
    entry_last_name = ttk.Entry(student_window)
    entry_last_name.grid(row=3, column=1)

    ttk.Label(student_window, text="DOB (YYYY-MM-DD):").grid(row=4, column=0)
    entry_dob = ttk.Entry(student_window)
    entry_dob.grid(row=4, column=1)

    var_gender = tk.StringVar(value='M')
    ttk.Label(student_window, text="Gender:").grid(row=5, column=0)
    ttk.Radiobutton(student_window, text='M', variable=var_gender, value='M').grid(row=5, column=1,
sticky='w')
    ttk.Radiobutton(student_window, text='F', variable=var_gender, value='F').grid(row=5, column=1)
```

Source Code

```
    ttk.Label(student_window, text="Address:").grid(row=6, column=0)
    entry_address = ttk.Entry(student_window)
    entry_address.grid(row=6, column=1)

    ttk.Label(student_window, text="Emergency Contact:").grid(row=7, column=0)
    entry_emergency_contact = ttk.Entry(student_window)
    entry_emergency_contact.grid(row=7, column=1)

    ttk.Label(student_window, text="Subjects (5):").grid(row=8, column=0)
    entry_subject1 = ttk.Entry(student_window)
    entry_subject1.grid(row=8, column=1)

    entry_subject2 = ttk.Entry(student_window)
    entry_subject2.grid(row=9, column=1)

    entry_subject3 = ttk.Entry(student_window)
    entry_subject3.grid(row=10, column=1)

    entry_subject4 = ttk.Entry(student_window)
    entry_subject4.grid(row=11, column=1)

    entry_subject5 = ttk.Entry(student_window)
    entry_subject5.grid(row=12, column=1)

    ttk.Label(student_window, text="Class:").grid(row=13, column=0)
    entry_class = ttk.Entry(student_window)
    entry_class.grid(row=13, column=1)

    ttk.Label(student_window, text="Email:").grid(row=14, column=0)
    entry_email = ttk.Entry(student_window)
    entry_email.grid(row=14, column=1)

    ttk.Button(student_window, text="Submit", command=submit_student).grid(row=15, columnspan=2)
def register_teacher():
    def submit_teacher():
        staff_id = entry_staff_id.get()
        name = entry_name.get()
        subject_taught = entry_subject_taught.get()
        classes = [entry_class1.get(), entry_class2.get(), entry_class3.get()]
        email = entry_email.get()
        spouse_name = entry_spouse_name.get()
```

Source Code

```
    gender = var_gender.get()
    emergency_contact = entry_emergency_contact.get()
    address = entry_address.get()
    dob = entry_dob.get()
    date_of_joining = entry_date_of_joining.get()


    sql = f"""UPDATE STAFF
            SET NAME='{name}', SUBJECT_TAUGHT='{subject_taught}',
                CLASS1='{classes[0]}', CLASS2='{classes[1]}', CLASS3='{classes[2]}',
                EMAIL='{email}', SPOUSE_NAME='{spouse_name}',
                GENDER='{gender}', EMERGENCY_CONTACT='{emergency_contact}',
                ADDRESS='{address}', DOB='{dob}', DATE_OF_JOINING='{date_of_joining}'
            WHERE STAFFID='{staff_id}'"""

    cur.execute(sql)
    con.commit()
    messagebox.showinfo("Success", "Teacher Registered Successfully!")
    teacher_window.destroy()
teacher_window = tk.Toplevel(dashboard_window)
teacher_window.title("Teacher Registration")
teacher_window.geometry('700x500')

ttk.Label(teacher_window, text="Teacher Registration").grid(row=0, columnspan=2)

ttk.Label(teacher_window, text="Staff ID:").grid(row=1, column=0)
entry_staff_id = ttk.Entry(teacher_window)
entry_staff_id.grid(row=1, column=1)

ttk.Label(teacher_window, text="Name:").grid(row=2, column=0)
entry_name = ttk.Entry(teacher_window)
entry_name.grid(row=2, column=1)

ttk.Label(teacher_window, text="Subject Taught:").grid(row=3, column=0)
entry_subject_taught = ttk.Entry(teacher_window)
entry_subject_taught.grid(row=3, column=1)

ttk.Label(teacher_window, text="Class 1:").grid(row=4, column=0)
entry_class1 = ttk.Entry(teacher_window)
entry_class1.grid(row=4, column=1)
```

Source Code

```
 ttk.Label(teacher_window, text="Class 2:").grid(row=5, column=0)
entry_class2 = ttk.Entry(teacher_window)
entry_class2.grid(row=5, column=1)

ttk.Label(teacher_window, text="Class 3:").grid(row=6, column=0)
entry_class3 = ttk.Entry(teacher_window)
entry_class3.grid(row=6, column=1)

ttk.Label(teacher_window, text="Email:").grid(row=7, column=0)
entry_email = ttk.Entry(teacher_window)
entry_email.grid(row=7, column=1)

ttk.Label(teacher_window, text="Spouse Name:").grid(row=8, column=0)
entry_spouse_name = ttk.Entry(teacher_window)
entry_spouse_name.grid(row=8, column=1)

var_gender = tk.StringVar(value='M')
ttk.Label(teacher_window, text="Gender:").grid(row=9, column=0)
ttk.Radiobutton(teacher_window, text='Male', variable=var_gender, value='M').grid(row=9, column=1,
sticky='w')
ttk.Radiobutton(teacher_window, text='Female', variable=var_gender, value='F').grid(row=9, column=1)

ttk.Label(teacher_window, text="Emergency Contact:").grid(row=10, column=0)
entry_emergency_contact = ttk.Entry(teacher_window)
entry_emergency_contact.grid(row=10, column=1)

ttk.Label(teacher_window, text="Address:").grid(row=11, column=0)
entry_address = ttk.Entry(teacher_window)
entry_address.grid(row=11, column=1)

ttk.Label(teacher_window, text="DOB (YYYY-MM-DD):").grid(row=12, column=0)
entry_dob = ttk.Entry(teacher_window)
entry_dob.grid(row=12, column=1)

ttk.Label(teacher_window, text="Date of Joining (YYYY-MM-DD):").grid(row=13, column=0)
entry_date_of_joining = ttk.Entry(teacher_window)
entry_date_of_joining.grid(row=13, column=1)

ttk.Button(teacher_window, text="Submit", command=submit_teacher).grid(row=14, columnspan=2)
```

Source Code

```
def update_photo():
    cur.execute(f"SELECT ID,PICTURE FROM PROFILE WHERE id='{username_text}';")
    picstat=cur.fetchall()
    global img_name
    if picstat:
        img_name=[]
        temp=filedialog.askopenfilenames()
        for i in temp:
            img_name.append(i.replace("\\" , "\\\\"))
        imgfile = open(img_name[0], 'rb').read()
        imgfile = base64.b64encode(imgfile).decode('utf-8')
        args = (username_text, imgfile)
        q=f"UPDATE PROFILE SET picture='{imgfile}' where id='{username_text}';"
        cur.execute(q)
        con.commit()
        messagebox.showinfo("Success","Image Successfully saved ! Please login again to view Changes.")
    else:
        img_name=[]
        temp=filedialog.askopenfilenames()
        for i in temp:
            img_name.append(i.replace("\\" , "\\\\"))
        imgfile = open(img_name[0], 'rb').read()
        imgfile = base64.b64encode(imgfile).decode('utf-8')
        args = (username_text, imgfile)
        cur.execute('INSERT INTO PROFILE values(%s,%s);',args)
        con.commit()
        messagebox.showinfo("Success","Image Successfully saved ! Please login again to view Changes.")
def search_records():
    query_type = search_type.get()
    input_value = search_entry.get()
    if query_type == "student_adm_no":
            query = "SELECT admno,first_name,last_name,dob,address, emergency_contact, class FROM
students WHERE admno = %s"
            cur.execute(query, (input_value,))
            records = cur.fetchall()
            display_data(records, ['admno', 'first_name', 'last_name', 'dob', 'address', 'emergency_contact', 'class'])

    elif query_type == "student_class":
            query = "SELECT admno,first_name,last_name,dob,address,emergency_contact,class FROM students
WHERE class LIKE %s"
```

Source Code

```
            cur.execute(query, (input_value + "%",))
            records = cur.fetchall()
            display_data(records, ['admno', 'first_name', 'last_name', 'dob', 'address', 'emergency_contact', 'class'])

        elif query_type == "staff_id":
            query = "SELECT staffid,name,subject_taught,email,address FROM staff WHERE STAFFID = %s"
            cur.execute(query, (input_value,))
            records = cur.fetchall()
            display_data(records, ['STAFFID', 'NAME', 'SUBJECT_TAUGHT', 'EMAIL', 'ADDRESS'])

        elif query_type == "staff_subject":
            query = "SELECT staffid,name,subject_taught,email,address FROM staff WHERE SUBJECT_TAUGHT
= %s"
            cur.execute(query, (input_value,))
            records = cur.fetchall()
            display_data(records, ['STAFFID', 'NAME', 'SUBJECT_TAUGHT', 'EMAIL', 'ADDRESS'])


def display_data(data, columns):
    display_view.destroy()
    data_viewer = tk.Toplevel()
    data_viewer.title("Data Viewer")
    data_viewer.geometry('1000x1000')

    tree = ttk.Treeview(data_viewer, columns=columns, show='headings')
    for col in columns:
        tree.heading(col, text=col)
        tree.column(col, anchor="center")

    for row in data:
        tree.insert("", "end", values=row)

    tree.pack(expand=True, fill='both')

def show_data():
    global search_entry, search_type,display_view

    display_view = tk.Toplevel()
    display_view.title("Search Records")
```

Source Code

```
search_type = tk.StringVar(value="student_adm_no")
ttk.Label(display_view, text="Select Search Type:").grid(row=0, column=0)
search_options = ["student_adm_no", "student_class", "staff_id", "staff_subject"]
ttk.OptionMenu(display_view, search_type, search_options[0], *search_options).grid(row=0, column=1)


ttk.Label(display_view, text="Enter Value:").grid(row=1, column=0)
search_entry = ttk.Entry(display_view)
search_entry.grid(row=1, column=1)


ttk.Button(display_view, text="Search", command=search_records).grid(row=2, columnspan=2)

def open_dashboard(role):
    """
    Opens a new window with widgets and functionalities specific to the user role.
    """
    root.destroy()
    global dashboard_window
    dashboard_window = tk.Tk()
    dashboard_window.geometry('700x500')
    dashboard_window.title(f"{role.capitalize()} Dashboard")

    dashboard_window.tk.call('source', 'forest-dark.tcl')


    ttk.Style().theme_use('forest-dark')


    header = ttk.Label(dashboard_window, text=f"Welcome, {creds[0]}!", font=("Segoe UI", 20, "bold"))
    header.pack(pady=10)
    cur.execute(f'SELECT PICTURE FROM PROFILE WHERE ID="{username_text}";')
    pic = cur.fetchall()

    if not pic:
        cur.execute("SELECT PICTURE FROM PROFILE WHERE ID='blank.jpg';")
        pic=cur.fetchall()
    image = pic[0][0]
```

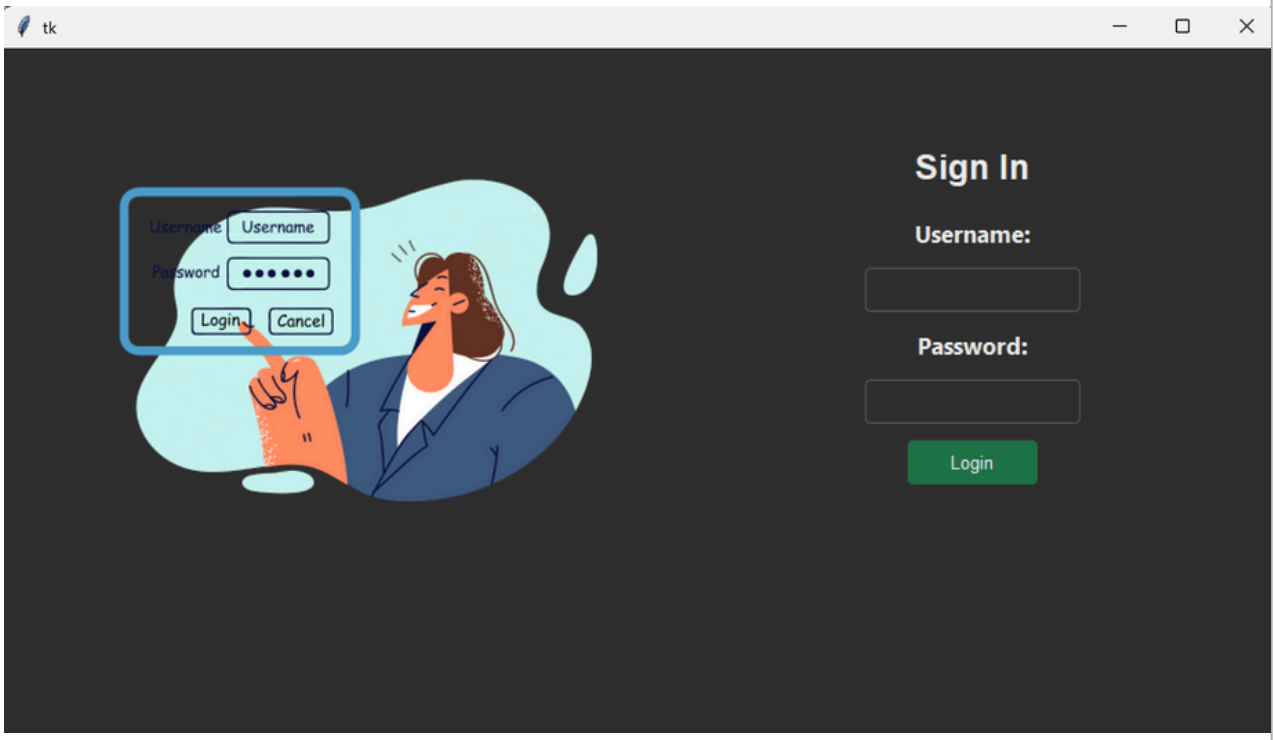
Source Code

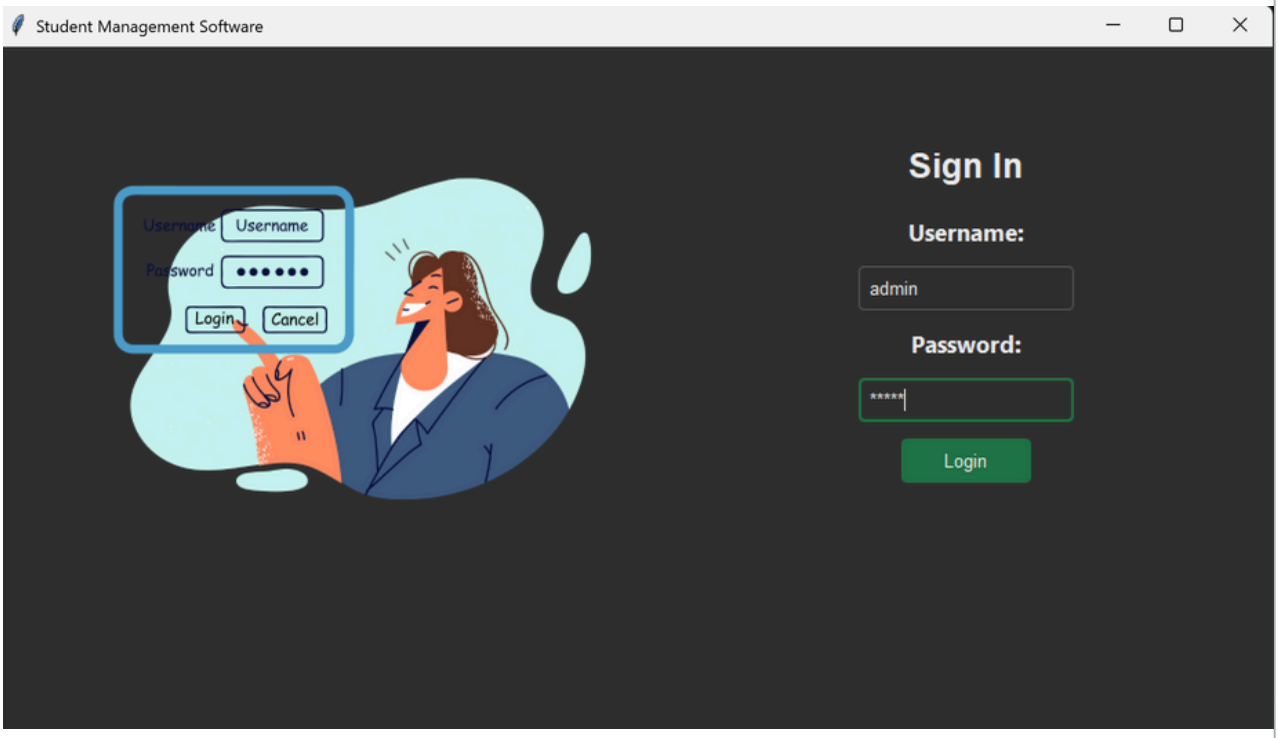
```
    binary_data = base64.b64decode(image)
    image = Image.open(io.BytesIO(binary_data))
    image = image.resize((100, 100), Image.LANCZOS)


    mask = Image.new('L', (100, 100), 0)
    draw = ImageDraw.Draw(mask)
    draw.ellipse((0, 0, 100, 100), fill=255)


    image.putalpha(mask)


    photo = ImageTk.PhotoImage(image)


    label = tk.Label(dashboard_window, image=photo)
    label.image = photo
    label.place(x=10, y=10)


    if role == 'PARENT':
        ttk.Label(dashboard_window, text="Parent Dashboard", font=("Segoe UI", 16, "bold")).pack(pady=10)
        ttk.Button(dashboard_window, text="View Attendance",
command=view_attendance_parent).pack(pady=10)
        ttk.Button(dashboard_window, text="Update Contact Information",
command=update_info_parent).pack(pady=10)
        ttk.Button(dashboard_window, text="Update Photo", command=update_photo).pack(pady=10)
        ttk.Button(dashboard_window, text="Logout", command=log_out
,style='Accent.TButton').pack(pady=10)

    elif role == 'TEACHER':
        ttk.Label(dashboard_window, text="Teacher Dashboard", font=("Segoe UI", 16, "bold")).pack(pady=10)
        ttk.Button(dashboard_window, text="Email Students", command=email_student).pack(pady=10)
        ttk.Button(dashboard_window, text="Mark Attendance", command=mark_attendance).pack(pady=10)
        ttk.Button(dashboard_window, text="Update Contact Information",
command=update_info_teacher).pack(pady=10)
        ttk.Button(dashboard_window, text="Update Photo", command=update_photo).pack(pady=10)
        ttk.Button(dashboard_window, text="Logout", command=log_out
,style='Accent.TButton').pack(pady=10)
```

Source Code

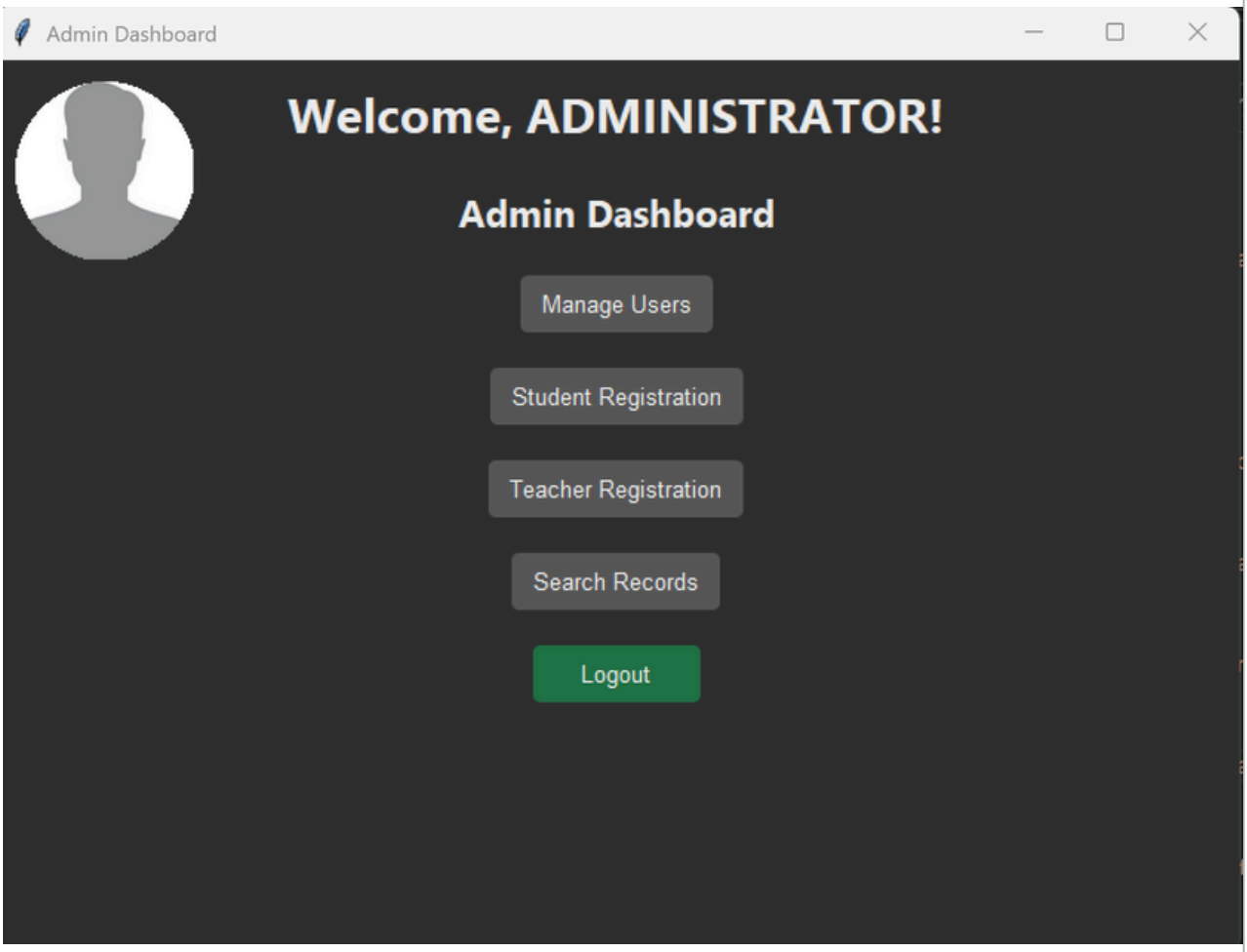
```
    elif role == 'ADMIN':
        ttk.Label(dashboard_window, text="Admin Dashboard", font=("Segoe UI", 16, "bold")).pack(pady=10)
        ttk.Button(dashboard_window, text="Manage Users", command=manage_users).pack(pady=10)
        ttk.Button(dashboard_window, text="Student Registration", command=register_student).pack(pady=10)
        ttk.Button(dashboard_window, text="Teacher Registration", command=register_teacher).pack(pady=10)
        ttk.Button(dashboard_window, text="Search Records", command=show_data).pack(pady=10)
        ttk.Button(dashboard_window, text="Logout", command=log_out
,style='Accent.TButton').pack(pady=10)



root = tk.Tk()
root.geometry('925x500+300+200')
root.title("Student Management Software")


root.tk.call('source',"forest-dark.tcl")


ttk.Style().theme_use('forest-dark')
img = PhotoImage(file='login.PNG')
img_label = tk.Label(root, image=img,)
img_label.place(x=50,y=50)

frame=ttk.Frame(root,width=350,height=350)
frame.place(x=625,y=60)

header2 = ttk.Label(frame, text="Sign In", anchor=tk.CENTER, font=("Open Sans", 19, "bold"))
header2.pack(pady=10)

lab1 = ttk.Label(frame, text="Username:", anchor=tk.CENTER, font=("Segoe UI", 13, "bold"))
username = ttk.Entry(frame)
lab1.pack(pady=10)
username.pack(pady=2)

lab2 = ttk.Label(frame, text="Password:", anchor=tk.CENTER, font=("Segoe UI", 13, "bold"))
pwd = ttk.Entry(frame, show="*")
lab2.pack(pady=10)
pwd.pack(pady=2)

button = ttk.Button(frame, text='Login', style='Accent.TButton', command=handle_login)
button.pack(pady=10)
root.mainloop()
```

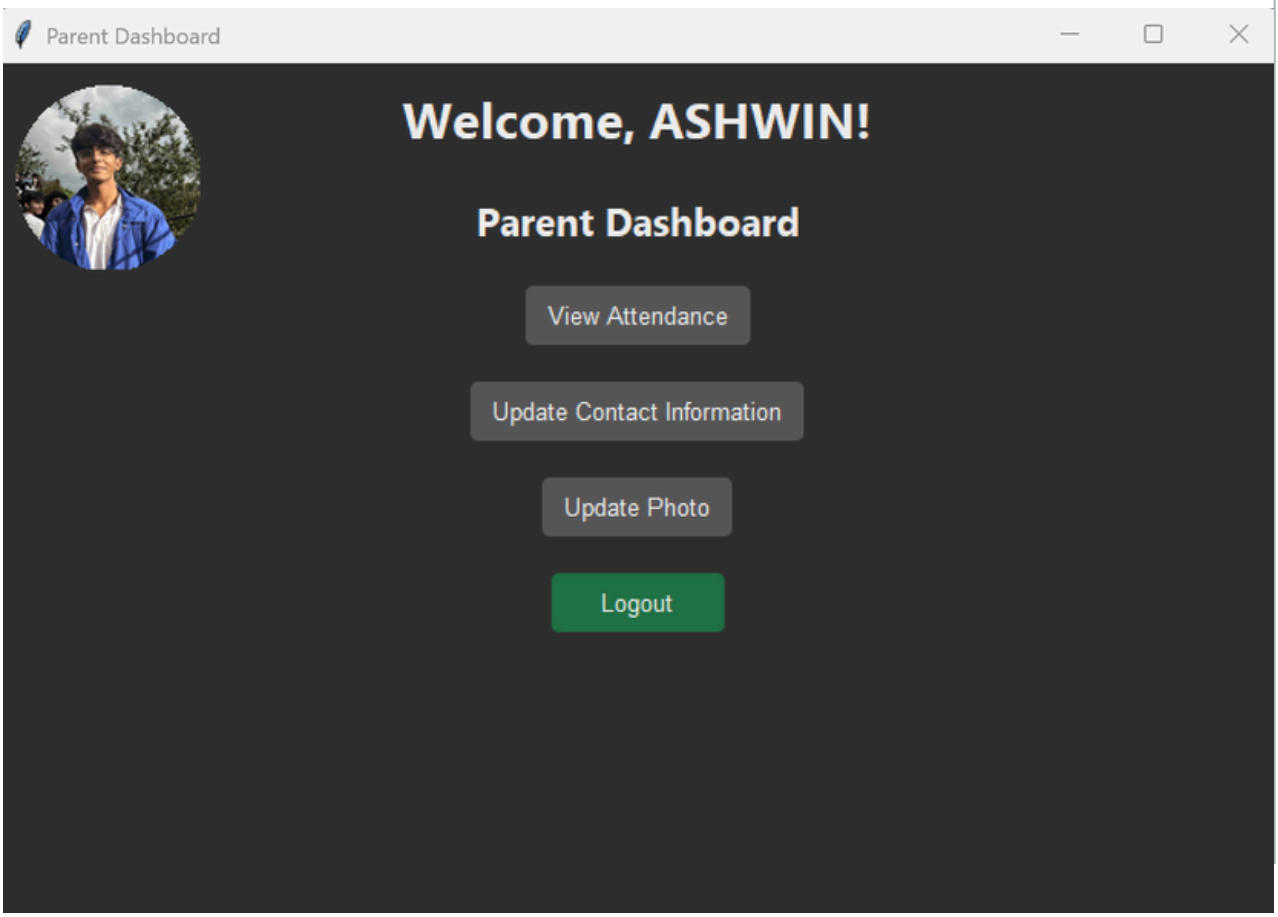
Output


tk
Username
Password
Login
Cancel
Sign In
Username:
Password:
Login

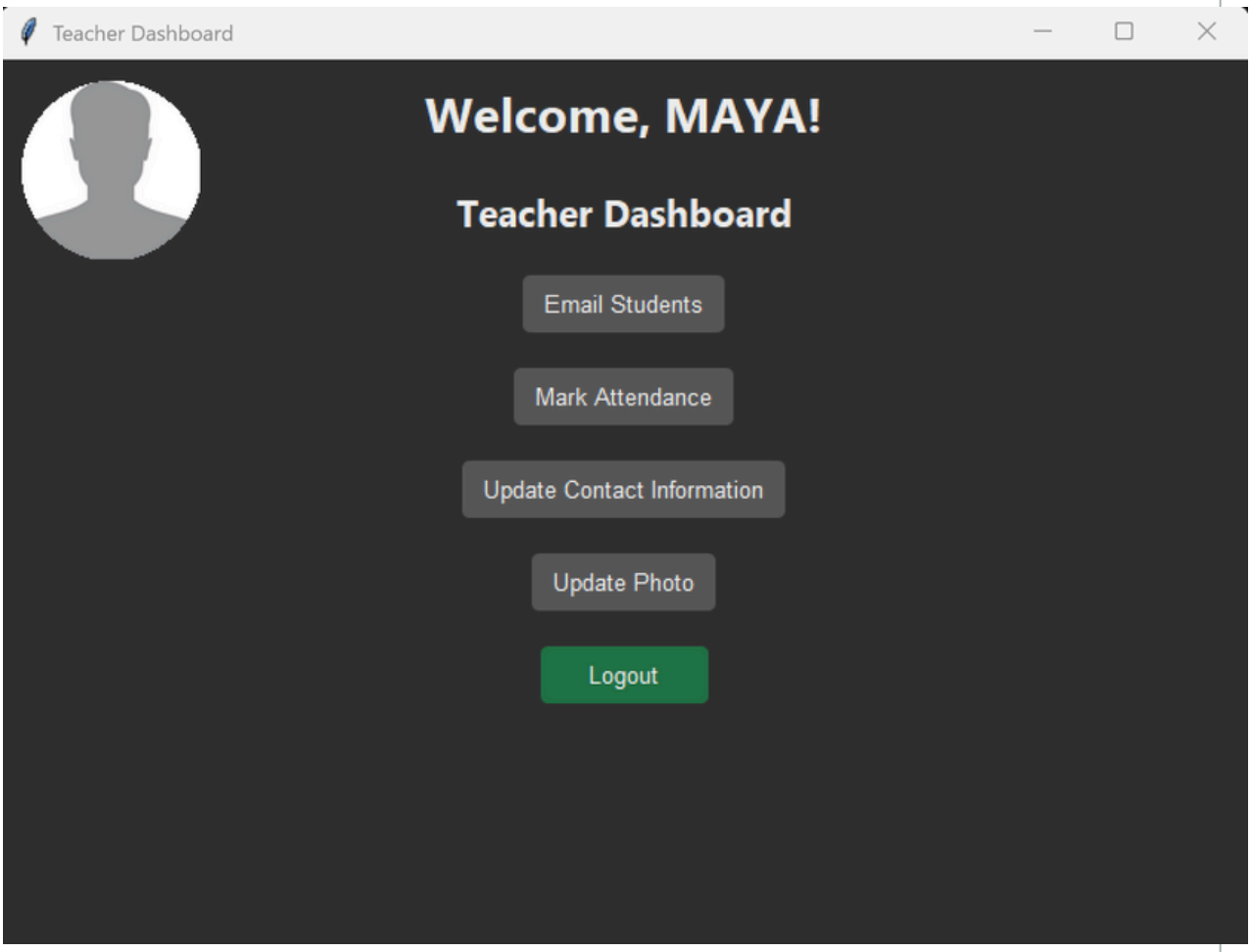

Student Management Software
Username
Password
Login
Cancel
Sign In
Username:
admin
Password:
*****
Login

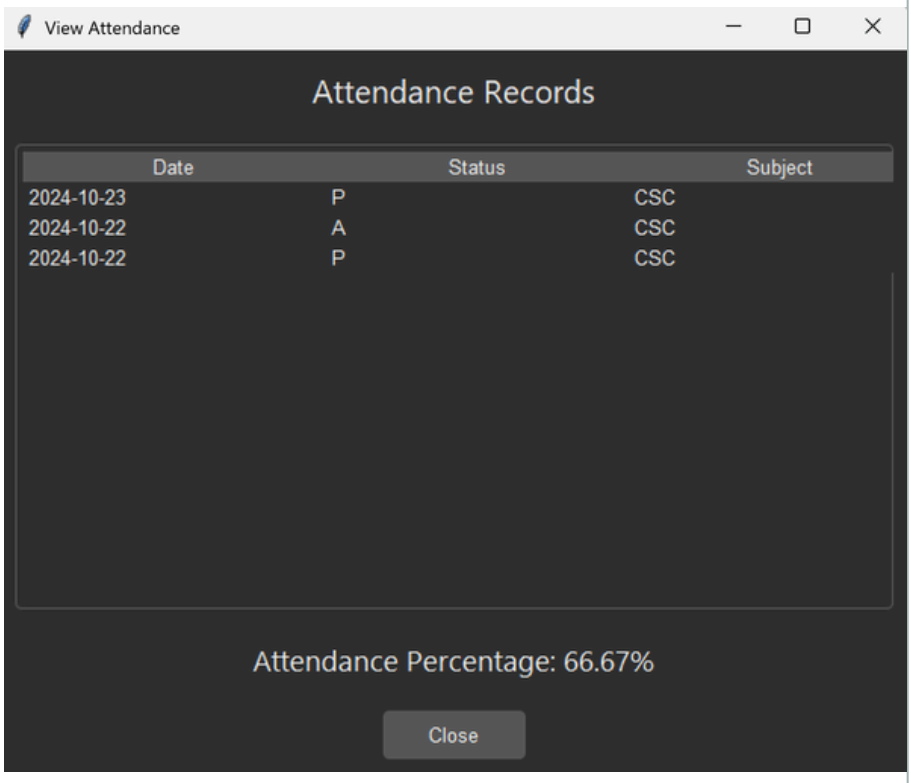
Output


Admin Dashboard
Welcome, ADMINISTRATOR!
Admin Dashboard
Manage Users
Student Registration
Teacher Registration
Search Records
Logout

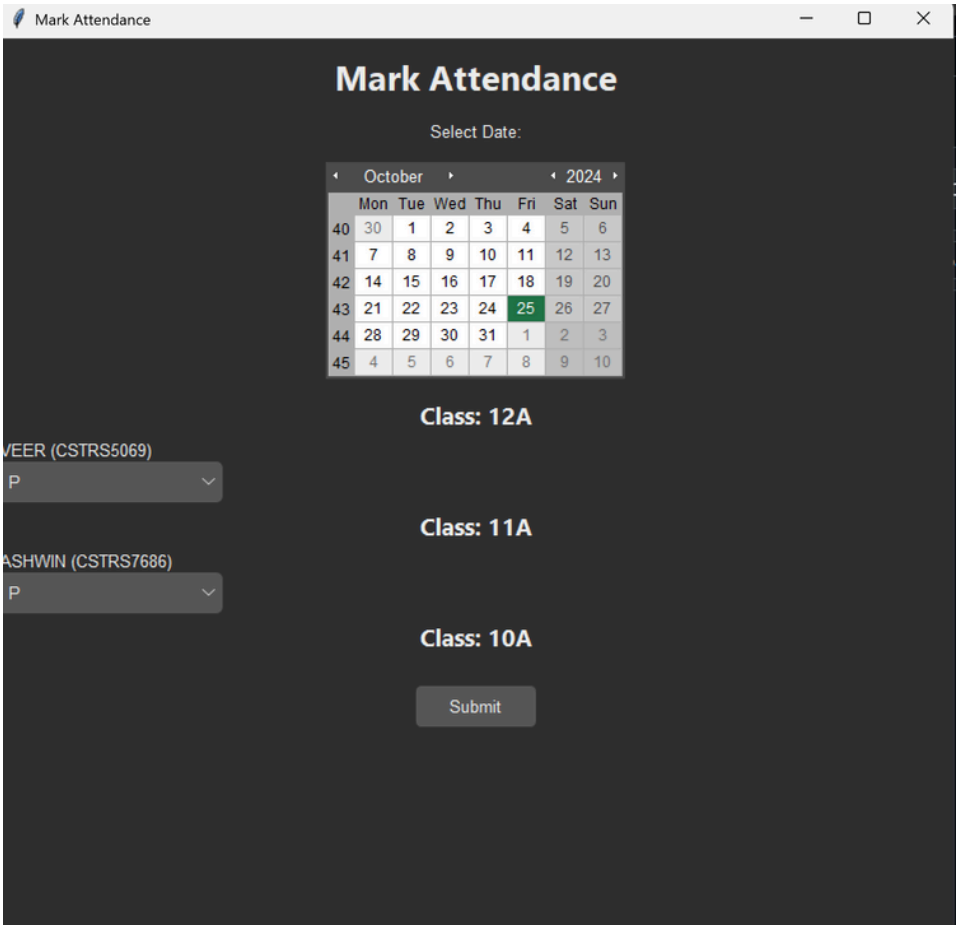

Parent Dashboard
Welcome, ASHWIN!
Parent Dashboard
View Attendance
Update Contact Information
Update Photo
Logout

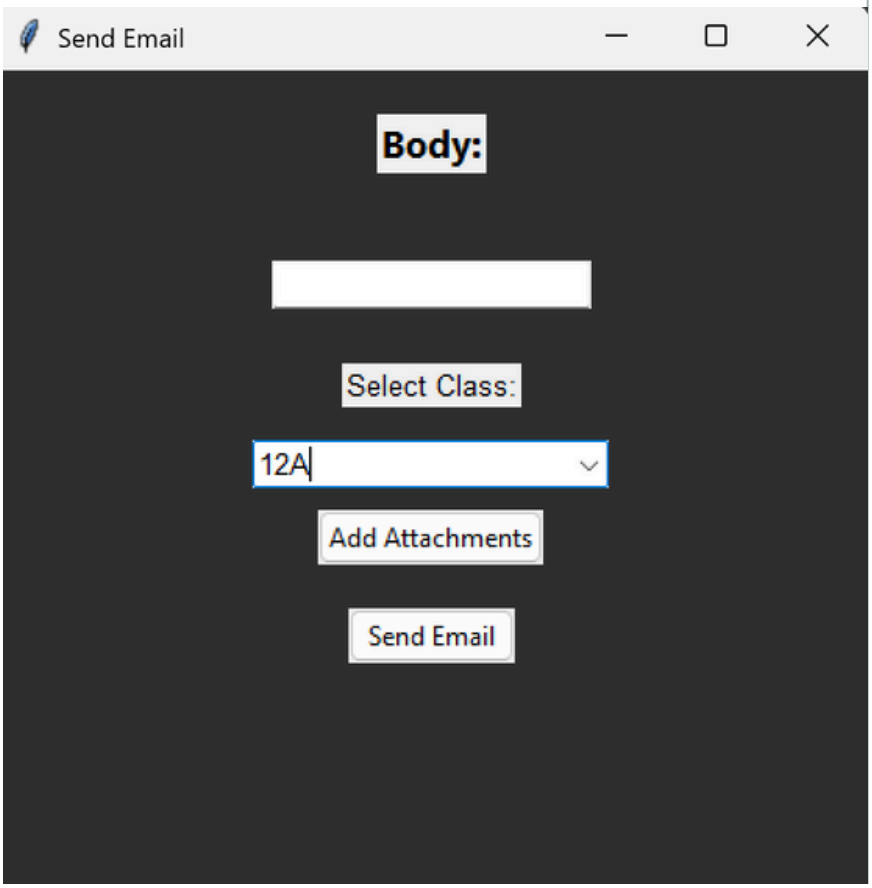
Output


Teacher Dashboard
Welcome, MAYA!
Teacher Dashboard
Email Students
Mark Attendance
Update Contact Information
Update Photo
Logout


View Attendance
Attendance Records
Date
Status
Subject
2024-10-23
P
CSC
2024-10-22
A
CSC
2024-10-22
P
CSC
Attendance Percentage: 66.67%
Close

Output


Mark Attendance
Mark Attendance
Select Date:
October
2024
Mon Tue Wed Thu Fri Sat Sun
40 30 1 2 3 4 5 6
41 7 8 9 10 11 12 13
42 14 15 16 17 18 19 20
43 21 22 23 24 25 26 27
44 28 29 30 31 1 2 3
45 4 5 6 7 8 9 10
Class: 12A
VEER (CSTRS5069)
P
Class: 11A
ASHWIN (CSTRS7686)
P
Class: 10A
Submit


Send Email
Body:
Select Class:
12A
Add Attachments
Send Email

SQL Tables

```
mysql> select * from credentials;
+---------------+----------+----------+---------+
| name          | username | password | role    |
+---------------+----------+----------+---------+
| ADMINISTRATOR | admin    | admin    | ADMIN   |
| MAYA          | CS002    | 1234     | TEACHER |
| VEER          | CSTRS5069 | 1709    | PARENT  |
| JOLLY GEORGE  | CS003    | JOLLY123 | TEACHER |
| PREETHA       | CS004    | 12345    | TEACHER |
| ASHWIN        | CSTRS7686 | 9302    | PARENT  |
+---------------+----------+----------+---------+
6 rows in set (0.00 sec)
```

```
mysql> select * from std_attendance;
+-----------+------------+--------+---------+
| admno     | adate      | status | subject |
+-----------+------------+--------+---------+
| CSTRS5069 | 2024-10-23 | P      | CSC     |
| CSTRS7686 | 2024-10-23 | P      | CSC     |
| CSTRS5069 | 2024-10-22 | P      | CSC     |
| CSTRS7686 | 2024-10-22 | A      | CSC     |
| CSTRS5069 | 2024-10-22 | A      | CSC     |
| CSTRS7686 | 2024-10-22 | P      | CSC     |
| CSTRS5069 | 2024-10-23 | P      | CHEM    |
+-----------+------------+--------+---------+
7 rows in set (0.01 sec)
```

SQL Tables

```
mysql> select * from students;
+-----------+------------+-------------+------------+--------+---------------------+-----------------+----------+----------+----------+----------+----------+-------+---------------------------+
| admno     | first_name | last_name   | dob        | gender | address             | emergency_contact | subject1 | subject2 | subject3 | subject4 | subject5 | class | email                     |
+-----------+------------+-------------+------------+--------+---------------------+-----------------+----------+----------+----------+----------+----------+-------+---------------------------+
| CSTRS5069 | VEER       | HEMAN KURUWA | 2007-04-20 | M      | XYZ LANE HOUSE ABC | 9895587774      | ENG      | PHY      | CHEM     | MATH     | CSC      | 12A   | cstrs5069@choiceschool.com |
| CSTRS7686 | ASHWIN     | ANIL        | 2007-03-19 | M      | THRIPUNITHURAA      | 898989875625    | ENG      | PHY      | CHEM     | MATH     | ENG      | 11A   | ashanil005@gmail.com      |
+-----------+------------+-------------+------------+--------+---------------------+-----------------+----------+----------+----------+----------+----------+-------+---------------------------+
2 rows in set (0.00 sec)
```

```
mysql> select * from staff;
+---------+-----------------+-----------------+---------+---------+---------+-------------------------------+-------------+--------+-------------------+-----------------+------------+-----------------+
| STAFFID | NAME            | SUBJECT_TAUGHT  | CLASS1  | CLASS2  | CLASS3  | EMAIL                         | SPOUSE_NAME | GENDER | EMERGENCY_CONTACT | ADDRESS         | DOB        | DATE_OF_JOINING |
+---------+-----------------+-----------------+---------+---------+---------+-------------------------------+-------------+--------+-------------------+-----------------+------------+-----------------+
| CS002   | MAYA MONEYKUMAR | CSC             | 12A     | 11A     | 10A     | maya.moneykumar@choiceschool.com | NULL        | F      | 94569565986       | ABC LANE XYZ HOUSE | 1984-03-19 | 2024-08-02      |
| CS003   | JOLLY GEORGE    | CHEM            | 12A     | 11B     | 10C     | jolly@example.com             | XYZ         | M      | 9895878587456     | THEVARA         | 1965-05-21 | 2002-05-21      |
| CS004   | PREETHA         | ENG             | 12A     | 12E     | 11A     | -                             | -           | M      | -                 | -               | 2004-01-20 | 2005-01-20      |
+---------+-----------------+-----------------+---------+---------+---------+-------------------------------+-------------+--------+-------------------+-----------------+------------+-----------------+
3 rows in set (0.00 sec)
```

Bibliography

1. Sumita Arora Class 12 Computer Science Textbook
2. GeeksforGeeks.com
3. Stack Overflow
4. Advanced Guide to Python 3 Programming by John Hunt

FIN.

Done by :

1. Veer Heman Kuruwa
2. Ahilesh M
3. Vedant Somesh Nambiar

https://github.com/veerkuruwa20/py-schoolmanagement