“ATTENDANCE MAESTRO”

Student Attendance Management System

INDEX

CASE STUDY

In educational institutions, efficient attendance management is crucial for monitoring student participation and identifying patterns in attendance. This case study explores the development of an Attendance Management System using Python and MySQL, which not only records attendance but also visualizes it through graphs for better insights.

Objective:
The primary objective is to create a robust and user-friendly system for tracking student attendance in classes, storing the data in a MySQL database, and providing visualizations for easy analysis.

Database Design:
MySQL is used to create a database named attendance_system. Tables are created for storing student information (students), teachers' information (tlogin), and attendance records (dailyatt).

User Interface (UI):
Tkinter is employed to design a user-friendly interface. The main screen displays options for managing classes, taking attendance, and viewing attendance statistics.

Attendance Tracking:
The "Take Attendance" feature allows teachers to mark students as present or absent or special permission for a specific class. The attendance data is then stored in the dailyatt table.

Data Visualization:

- Matplotlib is utilized to create interactive graphs.
- The system generates attendance trends over time, allowing users to identify patterns easily.
- Generation of graphical reports using Python libraries such as Matplotlib.
- Graphs displaying class-wise attendance trends, student attendance percentages, and more.
- Based on graphical insights, the school administration identifies patterns of attendance.
- Strategies are formulated to address issues like low attendance in specific classes or among certain students.

Data Security:
User authentication ensures that only authorized individuals can access and modify attendance data. Passwords are securely typed using *.

Integration with MySQL:
Python's MySQL connector is used to establish a connection between the system and the MySQL database. SQL queries are employed to fetch and update attendance data.

Export Data into CSV File:
The export function in Python is not a built-in function, but when referring to exporting data or results, it provides the to_csv functions for exporting data frames to CSV files. This enables seamless data sharing and analysis across different platforms and tools.

Modules used in Python:

- Tkinter is a standard GUI (Graphical User Interface) toolkit in Python, providing a set of tools for creating windows, dialogs, buttons, and other GUI elements, making it an essential module for building desktop applications.
- Matplotlib is a powerful data visualization library in Python, widely used for creating static, animated, and interactive plots and charts, making it an indispensable tool for analysts and scientists to convey insights from their data.
- The datetime module in Python offers classes for working with dates and times, providing functionality to parse, format, and perform arithmetic operations on dates, enabling developers to manage time-related information effectively.
- Messagebox is a part of the Tkinter library that allows the creation of pop-up message boxes in Python GUI applications, providing a simple and standardized way to interact with users through alert messages, prompts, and confirmations.
- The Treeview widget in Tkinter enables the creation of hierarchical and tabular data structures, making it a versatile tool for displaying and organizing data in a structured format within a graphical user interface.
- Pillow is a powerful image processing library in Python, serving as a successor to the Python Imaging Library (PIL). It provides essential functionalities for opening, manipulating, and saving various image formats, making it a go-to choice for image-related tasks in Python.

- Pandas is a powerful open-source data manipulation and analysis library for Python, offering data structures like DataFrame for handling structured data, and providing tools for cleaning, transforming, and analyzing datasets efficiently.
- NumPy is a fundamental numerical computing library for Python, enabling efficient operations on large, multi-dimensional arrays and matrices. Its robust mathematical functions and tools make it essential for scientific computing, data analysis, and machine learning applications in the Python ecosystem.
- csv module in Python is a built-in library that facilitates the reading and writing of Comma-Separated Values (CSV) files, a common format for tabular data. It simplifies the handling of CSV data by providing functions like csv.reader for reading and csv.writer for writing, making it an essential tool for data manipulation and interoperability.
- tkcalendar is a Python library that provides a customizable calendar widget for Tkinter-based graphical user interfaces. It simplifies the integration of date selection features, offering a user-friendly calendar interface for applications, ranging from simple date pickers to complex event scheduling systems.

Benefits:

- Streamlined attendance tracking process.
- Improved accuracy and reliability of attendance data.
- Enhanced decision-making through graphical insights.
- Easy identification of attendance patterns and trends.
- Efficient way to transfer students' data through csv files.

MYSQL TABLE STRUCTURES

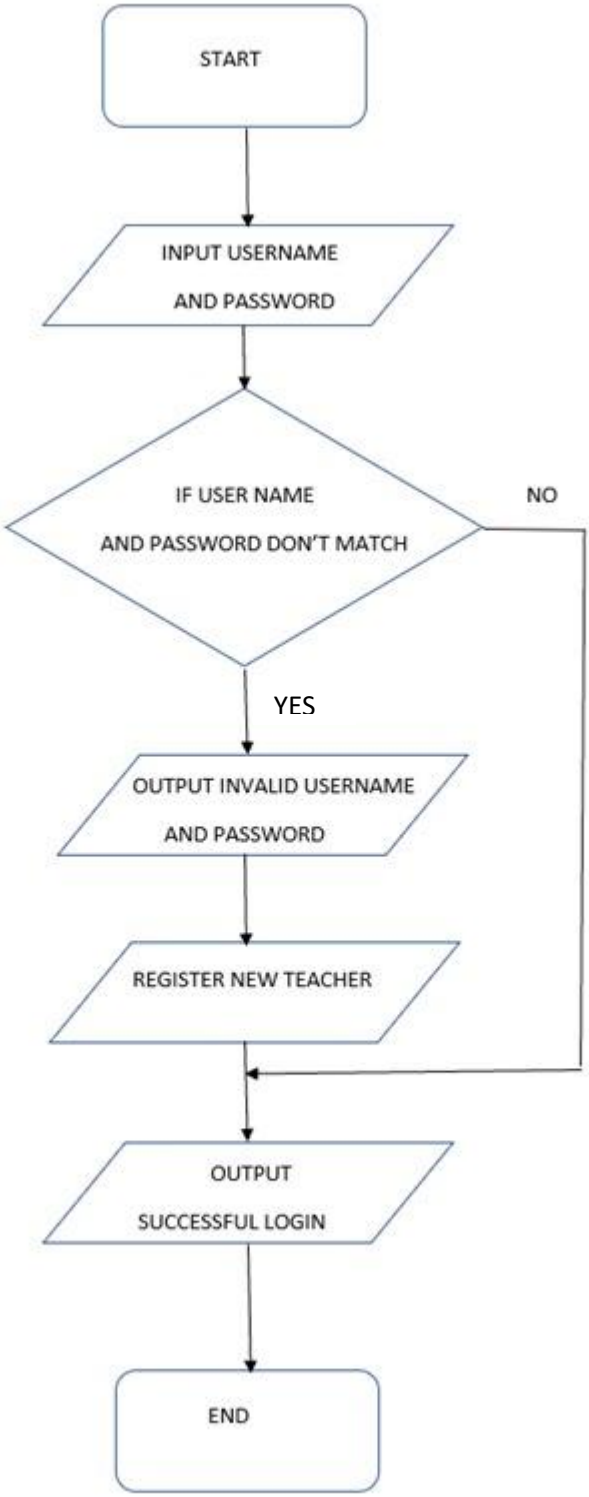
```
mysql> desc tlogin;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| name     | varchar(25) | YES  |     | NULL    |       |
| class    | varchar(3)  | YES  |     | NULL    |       |
| email    | varchar(25) | YES  |     | NULL    |       |
| password | varchar(10) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

```
mysql> desc students;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| usn    | varchar(10) | NO   | PRI | NULL    |       |
| name   | varchar(25) | YES  |     | NULL    |       |
| rollno | varchar(9)  | YES  |     | NULL    |       |
| class  | varchar(3)  | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
```

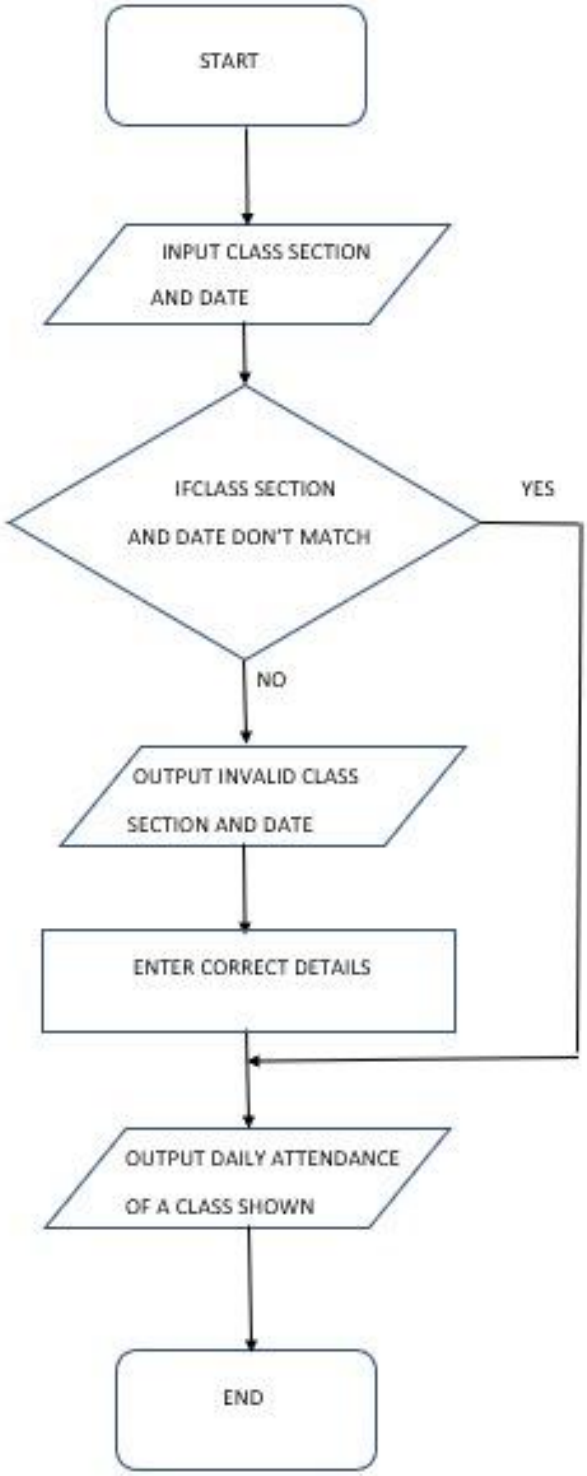
```
mysql> desc dailyatt;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| usn        | varchar(10)  | YES  |     | NULL    |       |
| name       | varchar(25)  | YES  |     | NULL    |       |
| rollno     | varchar(9)   | YES  |     | NULL    |       |
| class      | varchar(3)   | YES  |     | NULL    |       |
| date       | date         | YES  |     | NULL    |       |
| attendance | varchar(100) | YES  |     | NULL    |       |
| shift      | varchar(1)   | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
```

SAMPLE FLOWCHART

login()



daily_attendance()



ALGORITHM

ALGORITHM FOR FUNCTION login():

1. Start
2. Create a tkinter window with labels and columns to collect data.
3. Get the value of the username variable.
4. Get the value of the password variable.
5. Create a MySQL connector using host, username, password, and "attendance_system" as the database.
6. Create a cursor object for the connection.
7. Construct and execute a SQL query string to select all columns from the tlogin table where the condition is matched.
8. Fetch all the results from the query.
9. If the length of the fetched data is not zero, display an information messagebox saying successful login.
10. Otherwise, display an information messagebox with the text "Error" and "Invalid Username or Password".
11. Stop

ALGORITHM FOR FUNCTION register():

1. Start
2. Create a tkinter window with labels and columns to collect data.
3. Get the values for the username, password, email, class variables.
4. Construct and execute a SQL query string to insert the values in all columns in the tlogin table.
5. Display an information messagebox with the text saying successful teacher registration.
6. Stop

ALGORITHM FOR FUNCTION main():

1. Start
2. Create a tkinter window with buttons and background image.
3. Stop

ALGORITHM FOR FUNCTION take_attendance():

1. Start
2. Create a tkinter window with buttons and labels with columns.
3. Take entries for date, class and shift.
4. Create a button to enter the details.
5. Stop

ALGORITHM FOR FUNCTION enter():

1. Start
2. Create a MySQL connector using host, username, password, and "attendance_system" as the database.
3. Create a cursor object for the connection.
4. Construct and execute a SQL query string to select name, rollno and usn from the students table where the condition is matched.
5. Fetch all the results from the query and insert it into a dictionary.
6. Create labels which display the details for each student.
7. Create a button to save the attendance.
8. Mark present or absent or special permission for each student using a radio button.
9. Stop

ALGORITHM FOR FUNCTION save_attendance():

1. Start
2. Construct and execute a SQL query string to insert name, rollno, usn, attendance, date and shift into table dailyatt.
3. Stop

ALGORITHM FOR FUNCTION daily_attendance():

1. Start
2. Create a tkinter window with date, class label and column and button to display attendance.
3. Stop

ALGORITHM FOR FUNCTION daily_attendancenext():

1. Start
2. Create a MySQL connector using host, username, password, and "attendance_system" as the database.
3. Create a cursor object for the connection.
4. Create a tree view table with columns USN, Rollno, Name, Attendance, Shift.
5. Construct and execute a SQL query string to select USN, RollNo, Name, Attendance, shift from the students table where the condition is matched in ascending order of rollno.
6. Fetch all the results from the query.
7. Display all the details in the tree view table
8. Stop

ALGORITHM FOR FUNCTION monthly_attendance():

1. Start
2. Create a tkinter window with month, class label and column and button to display attendance.
3. Stop

ALGORITHM FOR FUNCTION monthly_attendancenext():

1. Start
2. Create a MySQL connector using host, username, password, and "attendance_system" as the database.
3. Create a cursor object for the connection.
4. Take entries for the class and section and month.
5. Create a tree view table with columns USN, RollNo, Name, Number of days present/pstar.
6. Construct and execute a SQL query string to select USN, RollNo, Name and number of days present from the dailyatt table where the condition is matched and grouped by usn, rollno, name in ascending order of rollno.
7. Fetch all the results from the query.
8. Display all the details in the tree view table.
9. Stop

ALGORITHM FOR FUNCTION register_new_student():

1. Start
2. Create a tkinter window with labels and columns to collect data.
3. Get the values for the name, class, rollno, usn variables.
4. Construct and execute a SQL query string to insert the values in all columns in the students table.
5. Display an information messagebox with the text saying successful student registration.
6. Stop

ALGORITHM FOR FUNCTION att_trends():

1. Start
2. Import matplotlib and pandas module
3. Create a MySQL connector using host, username, password, and "attendance_system" as the database.
4. Create a cursor object for the connection.

5. Construct and execute a SQL query string to select required data from dailyatt table.
6. Create a graph with x axis as class and y axis as number of students present by fetching data from dailyatt table.
7. Stop

ALGORITHM FOR FUNCTION schart():

1. Start
2. Import matplotlib and numpy modules.
3. Create a MySQL connector using host, username, password, and "attendance_system" as the database.
4. Create a cursor object for the connection.
5. Construct and execute a SQL query string to select usn, count of attendance when the student is present.
6. Fetch all the results from the query.
7. Create a graph from the fetched results with x axis showing usn of student and y axis showing the number of days present and percentage of attendance.
8. Stop

ALGORITHM FOR FUNCTION exportcsv():

1. Start
2. Create a MySQL connector using host, username, password, and "attendance_system" as the database.
3. Create a cursor object for the connection.
4. Take entry for rollno.
5. Create a button which exports data into the csv file.
6. Open a new csv file and input date, attendance and shift to the file.
7. Construct and execute a SQL query string to select date, attendance, shift from table dailyatt where the condition is matched.
8. Fetch all results from the query.
9. Write all the data fetched from the query into the csv file.
10. Stop

CODE

```
import tkinter as tk
from tkinter import messagebox
import mysql.connector
from datetime import date
import matplotlib.pyplot as plt
from tkinter import ttk

db =
mysql.connector.connect(host="localhost",user="root",password="mysql",datab
ase="attendance_system")
cursor = db.cursor()

def login():
    username = username_entry.get()
    password = password_entry.get()

    query = "SELECT * FROM tlogin WHERE name = %s AND password = %s"
    cursor.execute(query, (username, password))
    result = cursor.fetchone()

    if result:
        messagebox.showinfo("Login Successful", "Welcome, " + username)
        show_options()

    else:
        messagebox.showerror("Login Failed", "Invalid username or password")

def register():
    name = reg_name_entry.get()
    class_val = reg_class_entry.get()
    email = reg_email_entry.get()
    password = reg_password_entry.get()

    query = "INSERT INTO tlogin (name, class, email, password) VALUES (%s, %s,
%s, %s)"
    cursor.execute(query, (name, class_val, email, password))
    db.commit()
    messagebox.showinfo("Registration Successful", "You are now registered.")
```

```
def att_trends():
    import matplotlib.pyplot as plt
    import mysql.connector as mc
    import pandas as pd

    c = mc.connect(host="localhost", user="root", password="mysql",
database="attendance_system")
    cr = c.cursor()

    cr.execute("SELECT class, COUNT(attendance)/2 FROM dailyatt WHERE
date=curdate() and attendance in ('present','pstar') and shift in('1','2') GROUP BY
class")
    rows = cr.fetchall()

    # Create a DataFrame from the fetched data
    df = pd.DataFrame(rows, columns=['Class', 'No of students Present'])

    # Convert 'No of students Present' column to integers
    df['No of students Present'] = df['No of students Present'].astype(int)

    # Plotting
    plt.figure(figsize=(10, 6))  # Adjust the figure size as needed
    plt.bar(df['Class'], df['No of students Present'])
    plt.xlabel('Class')
    plt.ylabel('No of students Present')
    plt.title('Attendance per Class')
    plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility

    # Set y-axis ticks to specific values
    plt.yticks(range(1, df['No of students Present'].max() + 1))  # Set y-axis ticks
from 1 to the maximum present value

    plt.tight_layout()  # Adjust layout for better appearance
    plt.show()

    # Close the connection
    c.close()


def daily_attendancenext(c_var,d_var):
```

```
    root1.destroy()
    root2=Tk()
    root2.title("Attendance System")
    root2.geometry('700x530')

    import mysql.connector as mc
    from tkinter import ttk

c=mc.connect(host="localhost",user="root",password="mysql",database="atten
dance_system")
    cr=c.cursor()


    frame=Frame(root2)
    frame.grid(row=3,column=3)

    tree=ttk.Treeview(frame)
    tree["columns"]=("","","","","")

    tree.heading("0",text="USN",anchor=CENTER)
    tree.heading("1",text="RollNo",anchor=CENTER)
    tree.heading("2",text="Name",anchor=CENTER)
    tree.heading("3",text="Attendance",anchor=CENTER)
    tree.heading("4",text="Shift",anchor=CENTER)

    tree.column("#0",width=0)
    tree.column("0",anchor=CENTER,width=100)
    tree.column("1",anchor=CENTER,width=100)
    tree.column("2",anchor=CENTER,width=100)
    tree.column("3",anchor=CENTER,width=100)
    tree.column("4",anchor=CENTER,width=100)

    q="select usn,rollno,name,attendance,shift from dailyatt where class=%s and
date=%s order by rollno"

    cr.execute(q,[c_var,d_var])
    d=cr.fetchall()

    for x in d:
        tree.insert(parent="",index="end",values=x)
```

```
    tree.pack()
    c.close()
    root1.mainloop()
from tkinter import *
from tkcalendar import DateEntry
def daily_attendance():
    options_window.destroy()
    global root1
    root1 = Tk()
    root1.title("Attendance System")
    root1.geometry('700x530')

    global classsec
    global datee

    classseclabel = Label(root1, text="Class and Section")
    classseclabel.grid(row=2, column=2)
    classsec = Entry(root1)
    classsec.grid(row=2, column=3)

    l_date = tk.Label(root1, text="Select Date:")
    l_date.grid(row=3,column=2)
    e_date = DateEntry(root1, width=12, background='darkblue',
foreground='white', borderwidth=2)
    e_date.grid(row=3,column=3)

    b1 = Button(root1, text="Show attendance", fg="red",
command=lambda:daily_attendancenext(classsec.get(),e_date.get_date()))
    b1.grid(row=4, column=2)

    root1.mainloop()

def monthly_attendancenext(class__sec,month):
    root3.destroy()
    root4=Tk()
    root4.title("Attendance System")
    root4.geometry('700x530')

    import mysql.connector as mc
```

```
    from tkinter import ttk

c=mc.connect(host="localhost",user="root",password="mysql",database="atten
dance_system")
    cr=c.cursor()

    lbl = Label(root4, text = "Class:"+class__sec.get())
    lbl.grid(column=2,row=1)
    lbl.config(font=('Helvetica bold',15))
    lb2 = Label(root4, text = "Month:"+month.get())
    lb2.grid(column=3,row=1)
    lb2.config(font=('Helvetica bold',15))

    frame=Frame(root4)
    frame.grid(row=3,column=3)

    tree=ttk.Treeview(frame)
    tree["columns"]=("","","","")

    tree.heading("0",text="USN",anchor=CENTER)
    tree.heading("1",text="RollNo",anchor=CENTER)
    tree.heading("2",text="Name",anchor=CENTER)
    tree.heading("3",text="No of days present",anchor=CENTER)

    tree.column("#0",width=0)
    tree.column("0",anchor=CENTER,width=150)
    tree.column("1",anchor=CENTER,width=100)
    tree.column("2",anchor=CENTER,width=150)
    tree.column("3",anchor=CENTER,width=200)

    q="SELECT usn, rollno, name, round((COUNT(attendance)/2),2) FROM dailyatt
WHERE attendance in ('present','pstar') and shift in('1','2') AND class = %s AND
MONTH(date) = %s GROUP BY usn, rollno, name ORDER BY rollno"

    cr.execute(q,[class__sec.get(),month.get()])
    d=cr.fetchall()
    for x in d:
        tree.insert(parent="",index="end",values=x)

    tree.pack()
```

```
    c.close()
    root4.mainloop()

from tkinter import *
def monthly_attendance():
    options_window.destroy()
    global root3
    root3=Tk()
    root3.title("Attendance System")
    root3.geometry('700x530')

    class__seclabel=Label(root3,text="Enter class and
Section").grid(row=2,column=2)
    class__sec=StringVar()
    class__secentry=Entry(root3,textvariable=class__sec).grid(row=2,column=3)

    dateelabel=Label(root3,text="Enter month number").grid(row=3,column=2)
    month=StringVar()
    dateeentry=Entry(root3,textvariable=month).grid(row=3,column=3)

    b1=Button(root3,text="Show
attendance",fg="red",command=lambda:monthly_attendancenext(class__sec,m
onth))
    b1.grid(row=4,column=2)

    root3.mainloop()

def take_attendance():
    options_window.destroy()

    import tkinter as tk
    from tkinter import messagebox
    from tkcalendar import DateEntry
    import mysql.connector

    # Establish MySQL connection
    mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="mysql",
```

```
        database="attendance_system"
    )
    mycursor = mydb.cursor()

    std = None

    student_radios = {}
    global root7
    # Create Tkinter window
    root7 = tk.Tk()
    root7.geometry('700x530')
    root7.title("Attendance System")

    # Date selection with calendar
    label_date = tk.Label(root7, text="Select Date:")
    label_date.pack()
    entry_date = DateEntry(root7, width=12, background='darkblue',
foreground='white', borderwidth=2)
    entry_date.pack()

    label_std = tk.Label(root7, text="Class:")
    label_std.pack()
    entry_std = tk.Entry(root7)
    entry_std.pack()

    label_shift = tk.Label(root7, text="Shift:")
    label_shift.pack()
    entry_shift = tk.Entry(root7)
    entry_shift.pack()

    button = tk.Button(root7, text="Enter Attendance", command=lambda:
enter(entry_date, entry_std, entry_shift, student_radios, mycursor))
    button.pack()

    root7.mainloop()
def enter(entry_date, entry_std, entry_shift, student_radios, mycursor):
    std = entry_std.get()
    mycursor.execute("SELECT name, rollno, usn FROM students WHERE
class=%s", [std])
    students = mycursor.fetchall()
```

```
    for student in students:
        frame = tk.Frame(root7)
        frame.pack()

        student_name = student[0]
        label1 = tk.Label(frame, text=student_name)
        label1.pack(side=tk.LEFT, padx=5)

        student_rollno = student[1]
        label2 = tk.Label(frame, text=student_rollno)
        label2.pack(side=tk.LEFT, padx=5)

        student_usn = student[2]
        label3 = tk.Label(frame, text=student_usn)
        label3.pack(side=tk.LEFT, padx=5)

        var = tk.StringVar(value="Present")
        radio_present = tk.Radiobutton(frame, text="Present", variable=var,
value="Present")
        radio_present.pack(side=tk.LEFT, padx=5)
        radio_absent = tk.Radiobutton(frame, text="Absent", variable=var,
value="Absent")
        radio_absent.pack(side=tk.LEFT, padx=5)
        radio_pstar = tk.Radiobutton(frame, text="Special permission", variable=var,
value="Pstar")
        radio_pstar.pack(side=tk.LEFT, padx=5)

        student_radios[(student_name, student_rollno, student_usn)] = var  # Store
Radiobutton variable in the dictionary



    button = tk.Button(root7, text="Save Attendance", command=lambda:
save_attendance(entry_date, entry_std, entry_shift, student_radios, mycursor))
    button.pack()

def save_attendance(entry_date, entry_std, entry_shift, student_radios,
mycursor):
    date = entry_date.get_date()  # Get selected date from DateEntry widget
```

```
    class1 = entry_std.get()
    sh = entry_shift.get()
    attendance_data = []
    import mysql.connector
    mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="mysql",
        database="attendance_system"
    )
    mycursor = mydb.cursor()
    for (student_name, student_rollno, student_usn), var in
student_radios.items():
        attendance_status = var.get()  # Get the value of the RadioButton for each
student
        attendance_data.append((student_name, student_rollno, student_usn,
attendance_status, date, class1, sh))


    insert_query = "INSERT INTO dailyatt (name, rollno, usn, attendance, date,
class, shift) VALUES (%s,%s,%s, %s, %s,%s,%s)"
    mycursor.executemany(insert_query, attendance_data)
    mydb.commit()
    messagebox.showinfo("Take attendance", "Attendance saved successfully!")


import mysql.connector
import numpy as np
import matplotlib.pyplot as plt

def schart():
    mydb = mysql.connector.connect(host="localhost", user="root",
password="mysql", database="attendance_system")
    mycursor = mydb.cursor()

    mycursor.execute("SELECT usn, COUNT(*)/2 FROM dailyatt WHERE
attendance IN ('present', 'pstar') AND shift IN ('1', '2') GROUP BY usn")
    result = mycursor.fetchall()

    Names = []
```

```
    pre = []

    for i in result:
        Names.append(i[0])
        pre.append(float(i[1]))  # Convert to float here

    mycursor.execute('SELECT COUNT(*) FROM dailyatt WHERE rollno=120101')
    total_days = mycursor.fetchone()[0]  # Fetch the count directly

    # Calculate percentages
    total_days = int(total_days)
    percentages = [(count / total_days) * 100 for count in pre]

    # Visualizing Data using Matplotlib
    plt.figure(figsize=(10, 6))  # Adjust figure size as needed
    plt.bar(Names, pre)

    max_value = max(pre) + 5  # Add a margin to the maximum value for better
visualization
    plt.ylim(0, max_value)  # Set y-axis limits based on the maximum value

    plt.xlabel("Usn of Students")
    plt.ylabel("Number of days present")
    plt.title("Student's Information")

    # Display percentages above the chart
    for i, percentage in enumerate(percentages):
        plt.text(i, pre[i] + 0.1, f"{percentage:.2f}%", ha='center', va='bottom',
fontsize=8)

    plt.xticks(rotation=90)  # Rotate x-axis labels for better readability if needed
    plt.tight_layout()  # Adjust layout for better spacing
    plt.show()


import tkinter as tk
from tkinter import messagebox
import mysql.connector
from datetime import date
from tkinter import ttk
```

```
from tkinter import Label, Entry, StringVar

db =
mysql.connector.connect(host="localhost",user="root",password="mysql",datab
ase="attendance_system")
cursor = db.cursor()

def registers2():
    nameeg = namee.get()
    class1g = class1.get()
    Rg = R.get()
    usnng = usnn.get()

    query = "INSERT INTO students (name, class, usn, rollno) VALUES (%s, %s, %s,
%s)"
    cursor.execute(query, (nameeg,class1g,usnng,Rg))
    db.commit()
    messagebox.showinfo("Registration Successful", "Student registered.")

def register_new_student():
    options_window.destroy()
    global root5
    root5=tk.Tk()
    root5.title("Register student")
    root5.geometry("700x530")
    global usnn
    global namee
    global R
    global class1

    usnnlabel=Label(root5,text="Enter usn").grid(row=2,column=2)
    usnn=StringVar()
    usnnentry=Entry(root5,textvariable=usnn).grid(row=2,column=3)

    nameelabel=Label(root5,text="Enter name").grid(row=3,column=2)
    namee=StringVar()
    nameeentry=Entry(root5,textvariable=namee).grid(row=3,column=3)

    Rlabel=Label(root5,text="Enter rollno").grid(row=4,column=2)
    R=StringVar()
```

```
    Rentry=Entry(root5,textvariable=R).grid(row=4,column=3)

    class1label=Label(root5,text="Enter class").grid(row=5,column=2)
    class1=StringVar()
    class1entry=Entry(root5,textvariable=class1).grid(row=5,column=3)

    sr_attendance_btn = tk.Button(root5, text="Register", command=registers2)
    sr_attendance_btn.grid(row=6,column=3)

    root5.mainloop()


import csv
import mysql.connector as mc
from tkinter import *
def exportcsv():

    options_window.destroy()
    root6 = Tk()
    root6.title("Export data")
    root6.geometry('700x530')

    def export():
        c = mc.connect(host="localhost", user="root", password="mysql",
database="attendance_system")
        cr = c.cursor()

        rget = R1.get()
        filename = rget + '.csv'
        with open(filename, 'w', newline="") as f:
            wo = csv.writer(f)
            wo.writerow(['date', 'attendance', 'shift'])
            q = 'SELECT date, attendance, shift FROM dailyatt WHERE rollno=%s'
            cr.execute(q, [rget])
            data = cr.fetchall()
            for row in data:
                wo.writerow(row)
        messagebox.showinfo("Export successful", "Data stored in csv file")

        cr.close()
```

```
        c.close()

    R1label = Label(root6, text="Enter rollno")
    R1label.grid(row=1, column=1)
    R1 = StringVar()
    R1entry = Entry(root6, textvariable=R1)
    R1entry.grid(row=1, column=2)

    export_btn = Button(root6, text="Export data", command=export, width=10)
    export_btn.grid(row=4, column=1)

    root6.mainloop()

def show_options():
    root.destroy()
    global options_window
    options_window = tk.Tk()
    options_window.geometry('700x530')
    options_window.title("Attendance Management System")
    image1 = Image.open("class1.png")
    photo = ImageTk.PhotoImage(image1)

# Create a canvas with the image as the background
    canvas = tk.Canvas(options_window, width=image1.width,
height=image1.height)
    canvas.pack()

    canvas.create_image(0, 0, anchor=tk.NW, image=photo)

# Create a frame inside the canvas
    frame1 = tk.Frame(canvas, bg='white')
    frame1.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

# Create buttons inside the frame
    take_attendance_btn = tk.Button(frame1, text="Take Attendance",
command=take_attendance, width=30)
    take_attendance_btn.grid(row=0, column=0)

    daily_attendance_btn = tk.Button(frame1, text="Daily Attendance",
command=daily_attendance, width=30)
```

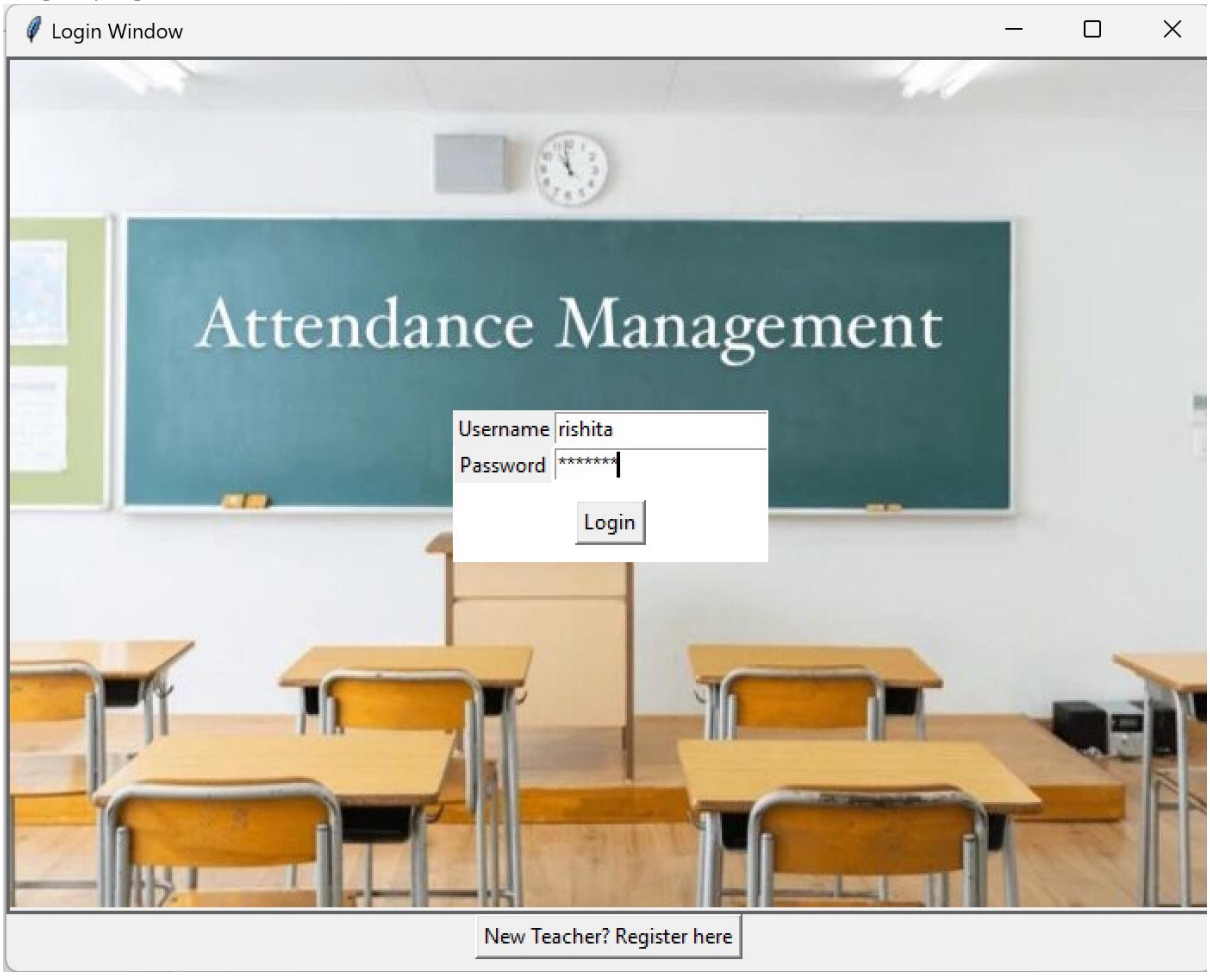
```
    daily_attendance_btn.grid(row=1, column=0)

    monthly_attendance_btn = tk.Button(frame1, text="Monthly Attendance",
command=monthly_attendance, width=30)
    monthly_attendance_btn.grid(row=2, column=0)

    register_new_student_btn = tk.Button(frame1, text="Register new student",
command=register_new_student, width=30)
    register_new_student_btn.grid(row=3, column=0)

    schartt_btn = tk.Button(frame1, text="Student chart", command=schart,
width=30)
    schartt_btn.grid(row=4, column=0)

    att_trends_btn=tk.Button(frame1,text="Attendance trends",
command=att_trends, width=30)
    att_trends_btn.grid(row=5, column=0)

    export_btn=tk.Button(frame1,text="Export into csv", command=exportcsv,
width=30)
    export_btn.grid(row=6, column=0)



    options_window.mainloop()

# Create main login window
import tkinter as tk
from PIL import Image, ImageTk

# Create the main window
root = tk.Tk()
root.geometry('700x530')
root.title("Login Window")

# Load the image
image1 = Image.open("class1.png")
photo = ImageTk.PhotoImage(image1)

# Create a canvas with the image as the background
```

```
canvas = tk.Canvas(root, width=image1.width, height=image1.height)
canvas.pack()

# Place the image on the canvas
canvas.create_image(0, 0, anchor=tk.NW, image=photo)

# Create the login frame
login_frame = tk.Frame(canvas, bg='white')  # Set the background color of the
frame
login_frame.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

# Add labels, entry widgets, and button to the login frame
username_label = tk.Label(login_frame, text="Username")
username_label.grid(row=0, column=0)
username_entry = tk.Entry(login_frame)
username_entry.grid(row=0, column=1)

password_label = tk.Label(login_frame, text="Password")
password_label.grid(row=1, column=0)
password_entry = tk.Entry(login_frame, show="*")
password_entry.grid(row=1, column=1)


login_btn = tk.Button(login_frame, text="Login", command=login)
login_btn.grid(row=2, columnspan=2, pady=10)

# Register Frame
register_frame = tk.Frame(root)
register_frame.pack(padx=20, pady=20)

reg_name_label = tk.Label(register_frame, text="Name")
reg_name_label.grid(row=0, column=0)
reg_name_entry = tk.Entry(register_frame)
reg_name_entry.grid(row=0, column=1)

reg_class_label = tk.Label(register_frame, text="Class")
reg_class_label.grid(row=1, column=0)
reg_class_entry = tk.Entry(register_frame)
reg_class_entry.grid(row=1, column=1)
```

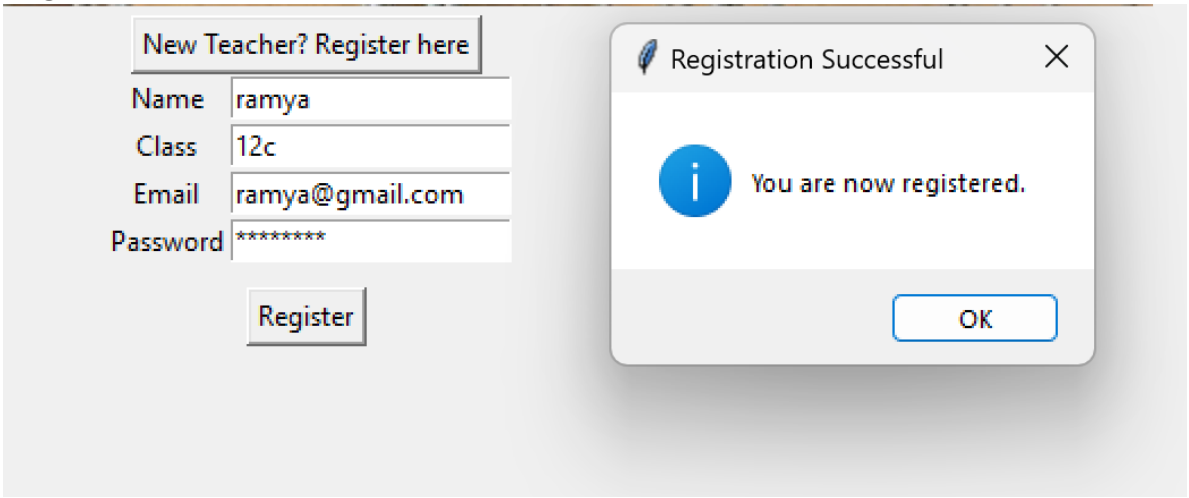
```
reg_email_label = tk.Label(register_frame, text="Email")
reg_email_label.grid(row=2, column=0)
reg_email_entry = tk.Entry(register_frame)
reg_email_entry.grid(row=2, column=1)

reg_password_label = tk.Label(register_frame, text="Password")
reg_password_label.grid(row=3, column=0)
reg_password_entry = tk.Entry(register_frame, show="*")
reg_password_entry.grid(row=3, column=1)

register_btn = tk.Button(register_frame, text="Register", command=register)
register_btn.grid(row=4, columnspan=2, pady=10)

# Hide register frame initially
register_frame.pack_forget()

# Function to toggle between login and register frames
def toggle_frame():
    if login_frame.winfo_ismapped():
        login_frame.pack_forget()
        register_frame.pack()
    else:
        register_frame.pack_forget()
        login_frame.pack()

# Button to toggle between login and register frames
toggle_btn = tk.Button(root, text="New Teacher? Register here",
command=toggle_frame)
toggle_btn.pack()

root.mainloop()
db.close()
```

SAMPLE SCREENSHOTS

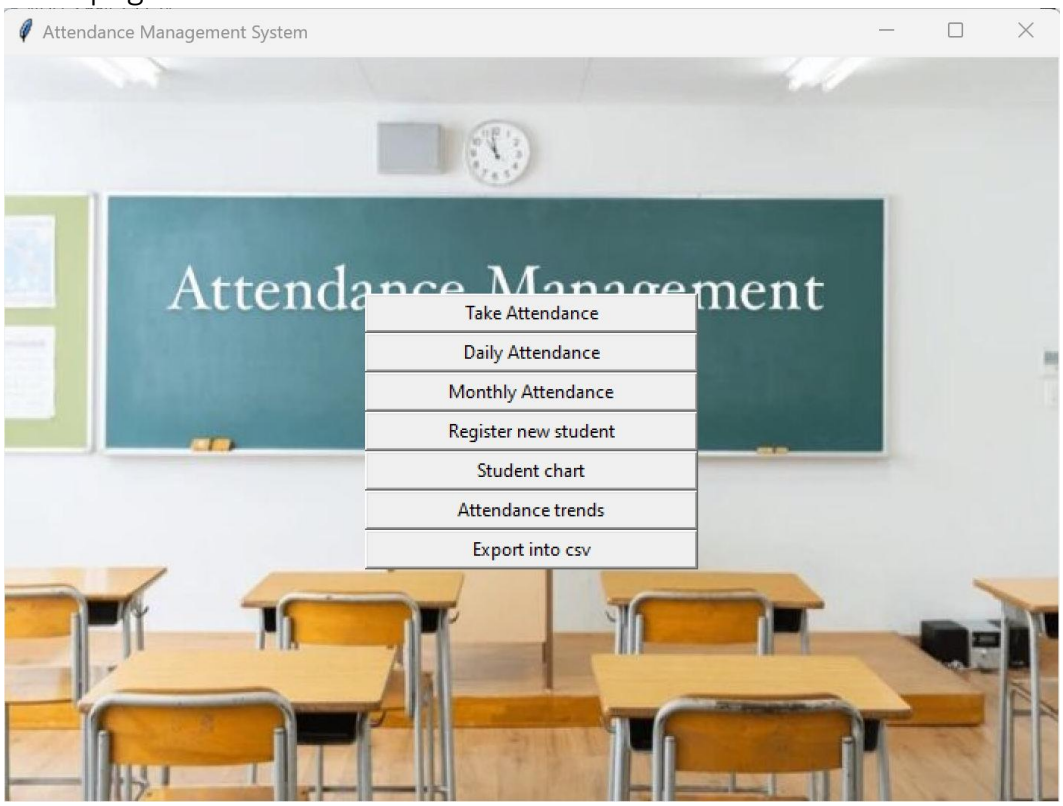
Login page:



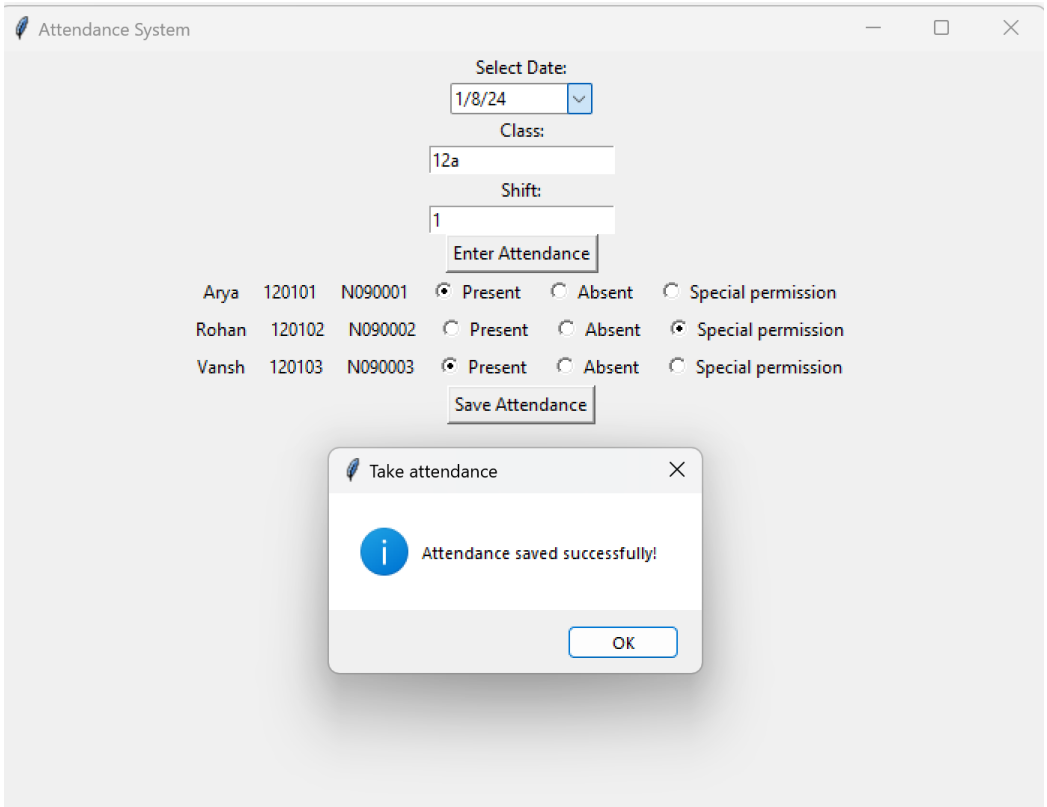
Register new teacher:



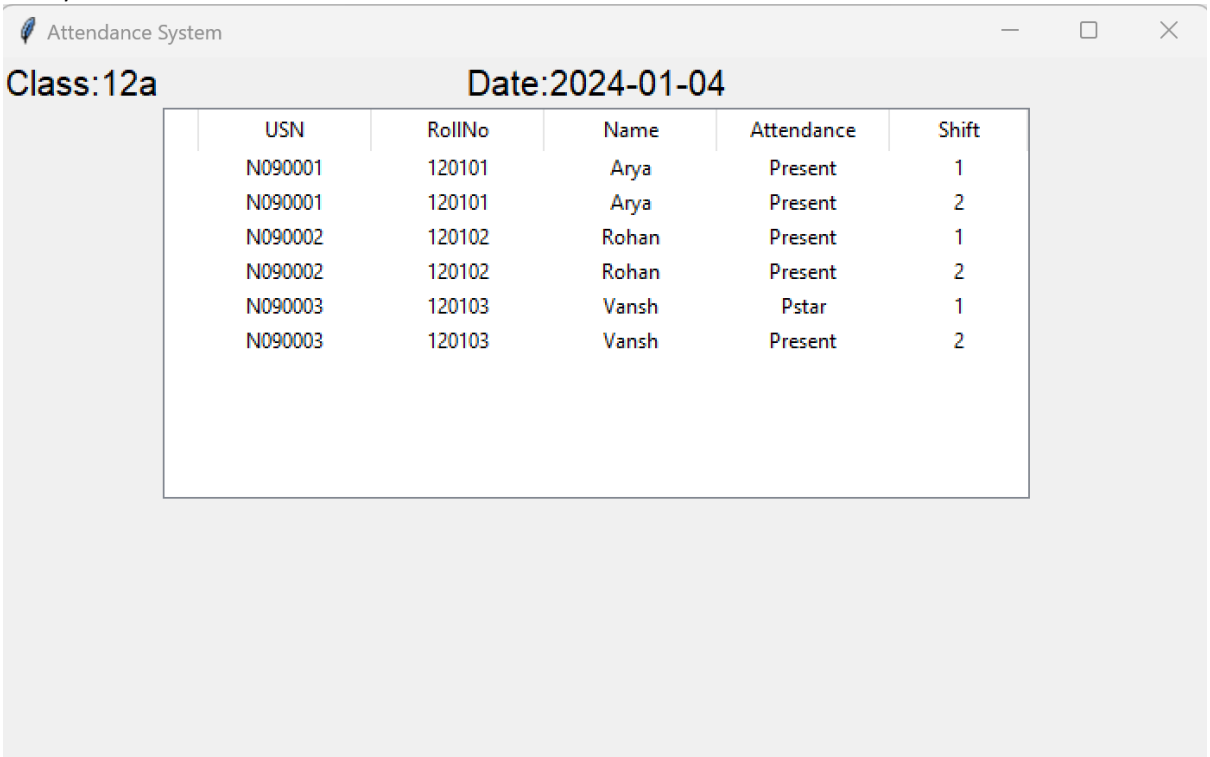
Main page:



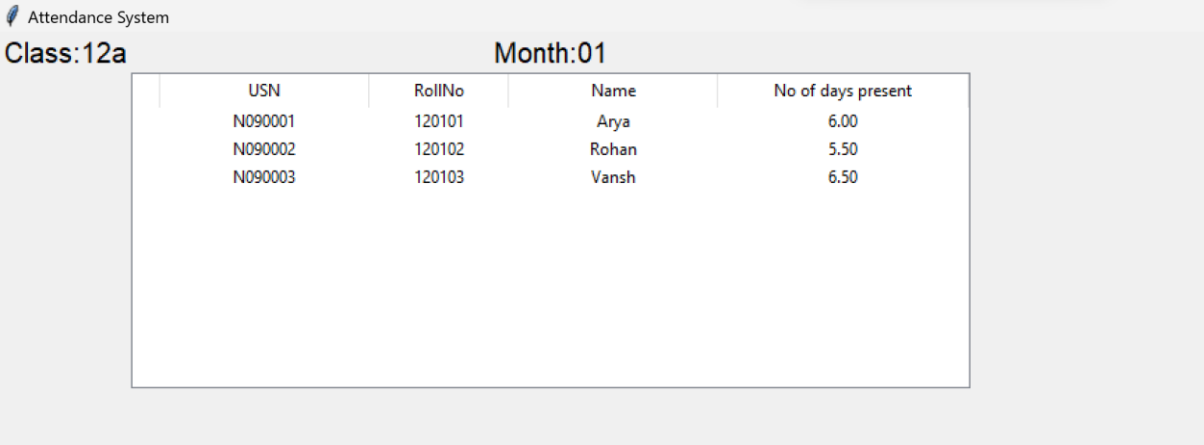
Take attendance:



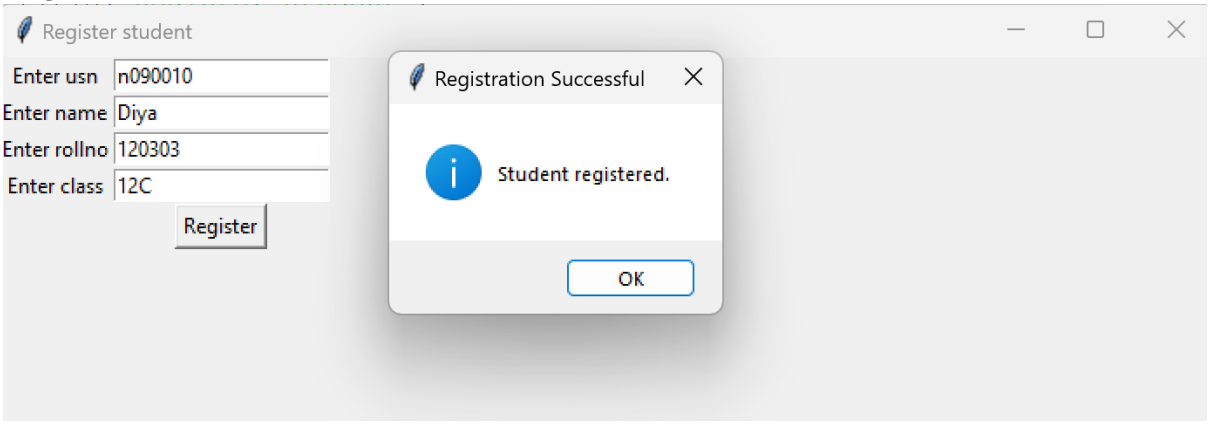
Daily attendance:



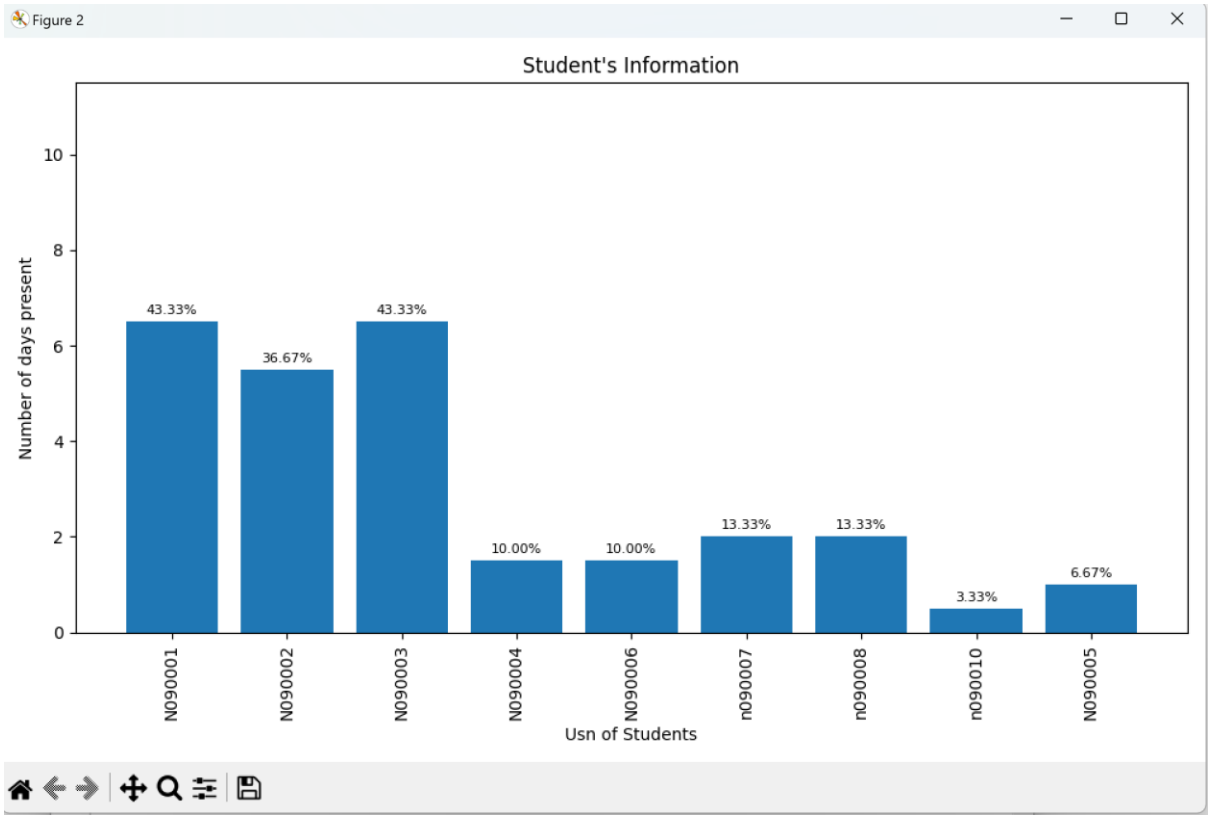
Monthly attendance:



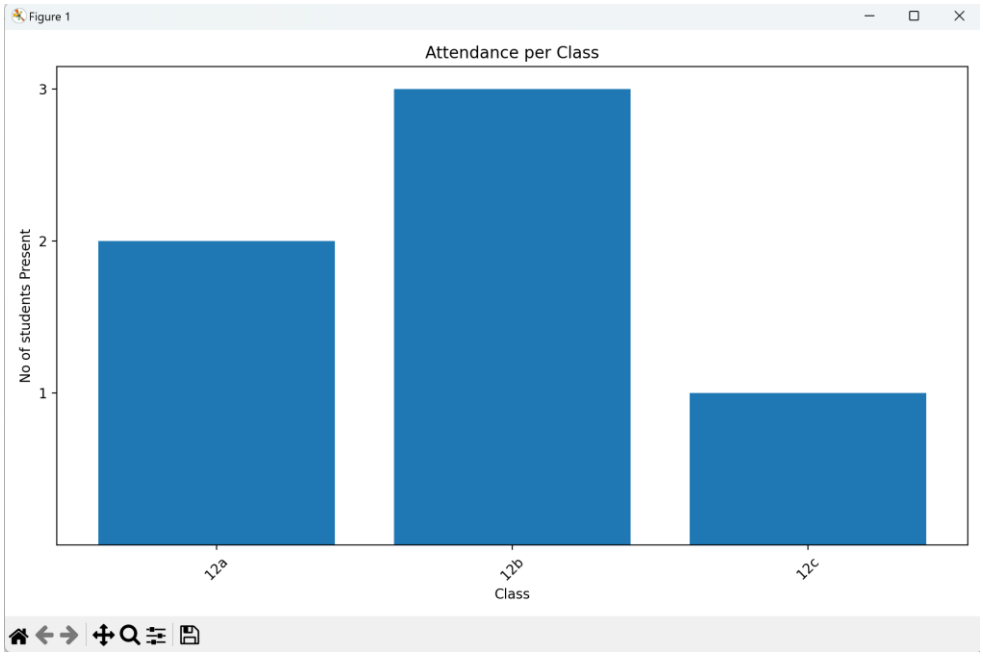
Register new student:



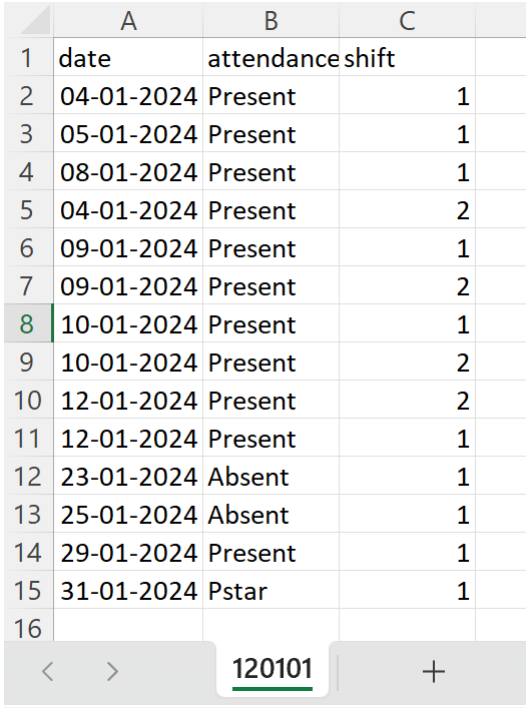
Students chart:



Attendance trends:



Exporting and saving data in csv file:



	A	B	C
1	date	attendance	shift
2	04-01-2024	Present	1
3	05-01-2024	Present	1
4	08-01-2024	Present	1
5	04-01-2024	Present	2
6	09-01-2024	Present	1
7	09-01-2024	Present	2
8	10-01-2024	Present	1
9	10-01-2024	Present	2
10	12-01-2024	Present	2
11	12-01-2024	Present	1
12	23-01-2024	Absent	1
13	25-01-2024	Absent	1
14	29-01-2024	Present	1
15	31-01-2024	Pstar	1
16			

120101

LIMITATIONS

- Dependency on User Input:
  Attendance systems often rely on manual input, and accuracy is contingent upon users consistently entering data. Inaccuracies may arise due to forgetfulness, intentional manipulation, or errors during data entry.
- Limited Analytical Capabilities:
  While basic attendance systems record and display attendance data, they may lack advanced analytical features. Institutions seeking in-depth insights may need to supplement these systems with additional analytics tools.
- No provision to mark late.

SCOPE FOR IMPROVEMENT

Automation and Integration:
Explore automation opportunities, such as integrating the system with biometric devices or RFID technology for seamless and accurate attendance tracking. Integration with other educational systems (e.g., student information systems) can streamline data flow.

Real-Time Updates:
Enhance the system to provide real-time updates for both administrators and users. This can include instant notifications for low attendance, late arrivals, or other relevant information.

Mobile Accessibility:
Develop a mobile application or ensure the system is responsive for mobile devices. This allows users to take attendance on the go and provides convenient access for both students and faculty.

Customization Options:
Provide customization options to accommodate diverse attendance tracking needs. This includes flexibility in defining attendance rules, class schedules, and data presentation.

REFERENCES

1. https://www.peoplehum.com/glossary/attendance-management
2. https://www.zoho.com/people/lp/attendance-management-system.html?network=g&device=c&campaignid=20596317785&adgroup=152434923285&keyword=attendance%20management%20system&matchtype=e&adposition=&creative=675422881819&gclid=CjwKCAiA-bmsBhAGEiwAoaQNmlUEHuqa3GEUCny601Y068Gl3CG7DrzeoNvKA5Fa4wxA7tNKmumxyBoCOzQQAvD_BwE
3. https://sumsapplication.com/sums-online-attendance-management/?utm_source=google&utm_medium=cpc&utm_campaign=Sums_Leads&gad_source=1&gclid=CjwKCAiA-bmsBhAGEiwAoaQNmslyAi-5BTNacKir5y8LrUkVh1FIC9a6eZa2SokWXd3Y7SNvJo7yZRoCxxYQAvD_BwE
4. https://www.peoplehrindia.com/hr-management-system/?utm_term=attendance%20management%20system&utm_campaign=People+Hr+Lead+1&utm_source=adwords&utm_medium=ppc&hsa_acc=5357314825&hsa_cam=1677490755&hsa_grp=69601781270&hsa_ad=659994746836&hsa_src=g&hsa_tgt=kwd-1328113491&hsa_kw=attendance%20management%20system&hsa_mt=e&hsa_net=adwords&hsa_ver=3&gad_source=1&gclid=CjwKCAiA-bmsBhAGEiwAoaQNmptk1ZsnrPeT1ulcNPkTkJlzmJvFFmnsuGLuQVQvNOg-x7ElnyuCLRoCMfQQAvD_BwE