

# **ATTENDANCE MANAGEMENT SYSTEM**

## **A MINI PROJECT REPORT**

### **18CSC207J - ADVANCED PROGRAMMING PRACTICE**

*Submitted by*

**SOUVIK BASAK (RA2111003011370)**

**PRITAM MAJUMDER (RA2111003011369)**

**ALEX SEBASTIAN (RA2111003011383)**

**ANIMESH NANDWANA (RA2111003011354)**

*Under the guidance of*

**Mrs. Maria Nancy A**

Assistant Professor, Department of Computing Technologies

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

**COMPUTER SCIENCE & ENGINEERING**

*of*

**FACULTY OF ENGINEERING AND TECHNOLOGY**



S.R.M. Nagar, Kattankulathur, Chengalpattu District

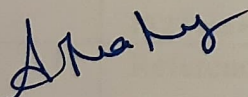
**MAY 2023**

# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

## BONAFIDE CERTIFICATE

Certified that Mini project report titled "Attendance Management System" is the bonafide work of Souvik Basak (RA2111003011370), Pritam Majumder (RA2111003011369), Alex Sebastian (RA2111003011383), Animesh Nandwana (RA2111003011354) who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

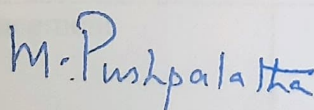
  
SIGNATURE

Mrs. Maria Nancy A

GUIDE

Assistant Professor

Department of Computing Technologies



SIGNATURE

Dr. M. Pushpalatha

HEAD OF THE DEPARTMENT

Professor & Head

Department of Computing Technologies



## TABLE OF CONTENT

| <b>SNO</b> | <b>TITLE</b>  | <b>PAGE</b> |
|------------|---|-------------|
| 1          | Abstract  | 3           |
| 2          | List Of Figures   | 4           |
| 3          | Abbreviations   | 5           |
| 4          | Introduction  | 6           |
| 5          | Literature Survey   | 7           |
| 6          | System Architecture & Design<br>6.1 Architecture diagram of proposed Attendance<br>Management System<br>6.2 Description of Module and components    | 8           |
| 7          | Methodology<br>7.1 Methodological Steps   | 10          |
| 8          | Coding & Testing  | 11-18       |
| 9          | Screenshots & Results<br>9.1 landing Page<br>9.2 Add Subjects<br>9.3 Add Students Data<br>9.4 Display Attendance<br>9.5 Deleting Student Attendance | 19-22       |
| 10         | Conclusion & Future Enhancement<br>10.1 Conclusion<br>10.2 Future Enhancement   | 23          |
| 11         | Reference   | 24          |

## **ABSTRACT**

Attendance management is important to every single organization; it can decide whether or not an organization such as educational institutions, public or private sectors will be successful in the future. Organizations will have to keep a track of people within the organization such as employees and students to maximize their performance. Managing student attendance during lecture periods has become a difficult challenge. The ability to compute the attendance percentage becomes a major task as manual computation produces errors, and wastes a lot of time. For the stated reason, an efficient application for attendance management system is designed to track student's activity in the class. This application takes attendance electronically and the records of the attendance are storing in a database. The system design using the tkinter library implemented using the power of Python. Sqlite3 used for the Database. Insertions, deletions, and changes of data in the system can do straightforward via the designed GUI without interacting with the tables. Different presentation of information is obtainable from the system. The test case of the system exposed that the system is working enormously and is ready to use to manage to attend students.

## **LIST OF FIGURES**

1. System Architecture
2. Attendance Management System Home page
3. Data Entry Page
4. Attendance Record Page
5. Student Data Entry Page
6. Student Data Delete Page
7. Result View Page

## **ABBREVIATIONS**

1. Admin – Administrator
2. ID – Identity
3. GUI – Graphical User Interface
4. IoT – Internet of Things
5. UI – User Interface

## **CHAPTER 1**

### **INTRODUCTION**

Due to student's interest in classrooms, and whose is the largest union in the study environment of university or institution, so recording absence at a department having a large number of students in a classroom is a difficult task and time-consuming. Moreover, the process takes much time, and many efforts are spent by the staff of the department to complete the attendance rates for each student. So in many institutions and academic organizations, attendance is a very important criterion which is used for various purposes. These purposes include record keeping, assessment of students, and promotion of optimal and consistent attendance in class. As long as in many developing countries, a minimum percentage of class attendance is required in most institutions and this policy has not been adhered to, because of the various challenges the present method of taking attendance presents. The process of recording attendances for students was in the form of hardcopy papers and the system was manually done. Besides wasting time and taking efforts for preparing sheets and documents, other disadvantages may be visible to the traditional one due to loss or damage to the sheets-sheet could be stolen.

Looking for a solution for above mentioned problems in attendance management system of school and colleges is the application that takes attendance electronically and the records of the attendance are storing in a database. A simple easy to understand GUI has been designed for easy traversal in the application different modules in the application provide different capabilities like adding, managing, editing, and deleting a student's attendance. The application is easy to use and usable in all system which have python and sql installed

## **CHAPTER 2**

### **LITERATURE SURVEY**

1. "Development of an Attendance Management System using Fingerprint Biometrics" by O. C. Nwankwo and C. I. Ani. This paper presents the development of an attendance management system using fingerprint biometrics. The system uses fingerprint recognition technology to identify employees and record their attendance. The system was tested on a small scale and was found to be efficient and accurate.

2. "Design and Implementation of an Attendance Management System using RFID Technology" by J. W. Chen and C. Y. Lee. This paper presents the design and implementation of an attendance management system using RFID technology. The system uses RFID tags to identify employees and record their attendance. The system was tested on a small scale and was found to be reliable and efficient.

3. "Attendance Management System based on Face Recognition" by S. B. Lee and S. Y. Kim. This paper presents the development of an attendance management system based on face recognition technology. The system uses cameras to capture the employees' faces and identify them. The system was tested on a small scale and was found to be accurate and efficient.

4. "Development of an Attendance Management System using Mobile Phone Technology" by M. Z. Al-Fattah and M. H. Khan. This paper presents the development of an attendance management system using mobile phone technology. The system uses the mobile phone's GPS and time stamp to record the employees' attendance. The system was tested on a small scale and was found to be reliable and efficient.

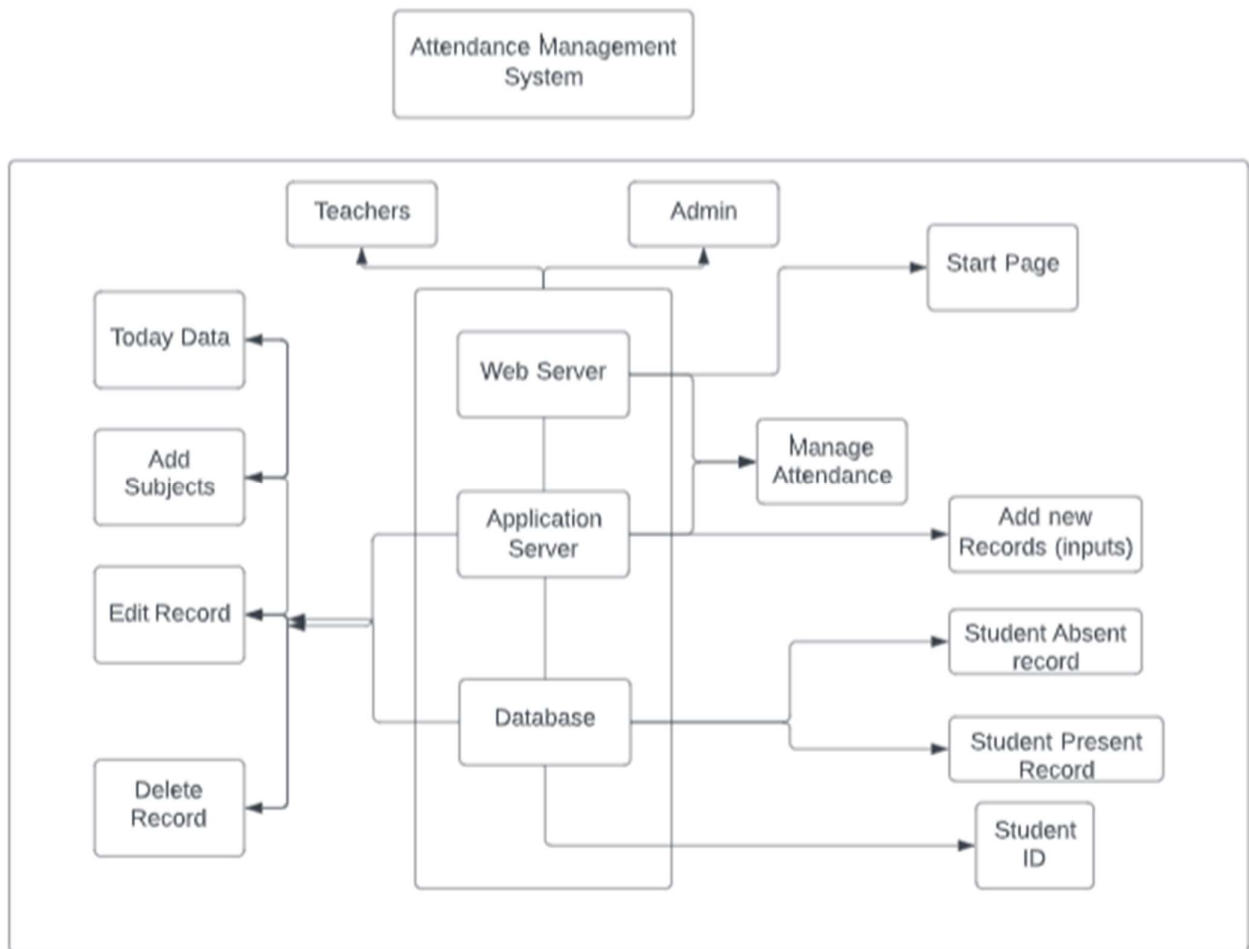
5. "An Automated Attendance Management System using Barcode Technology" by S. S. Alam and M. S. Islam. This paper presents the development of an attendance management system using barcode technology. The system uses barcode scanners to identify employees and record their attendance. The system was tested on a small scale and was found to be accurate and efficient.

Overall, these studies demonstrate the various technologies that can be used in an attendance management system, such as fingerprint biometrics, RFID, face recognition, mobile phone technology, barcode technology, and IoT. These systems have the potential to improve accuracy, efficiency, and reliability in recording employee attendance.



## CHAPTER 3

### SYSTEM ARCHITECTURE AND DESIGN



## **MODULE & COMPONENT**

### **DESCRIPTION**

#### **module: StartPage**

This module is responsible for displaying the text and the button for different functionalities in Attendance Management System Project in Python.

This module launches the application and leads the user to the landing page of the application.

#### **module: NewRecord**

This module is responsible for adding new records when the user clicks “Add new record” this module will come into action.

#### **module: ManageAttendance**

Whenever the user clicks on “Manage attendance” this module will come into the picture. This module offers multiple options to users like Show status, Today’s data and a Back to the home button.

#### **module: DeleteRecord**

This module facilitates the user to remove the attendance record of the student.

#### **module: EditRecord**

This module offers the user to edit multiple attributes of the student's record.

#### **module: AddSubjects**

This module offers the user the capability to add subjects in the subjects list.

#### **module: TodayData**

This module offers the user a look at the present-day attendance percentage of the student

## CHAPTER 4

### METHODOLOGY

The system calculates the attendance subject wise, that is the data of students and subjects are added manually by administrator, and whenever time for corresponding subject arrives the system automatically starts taking snaps and find whether human faces are appear in the given image or not.

#### Methodological Steps-:

- **Identify the requirements:** The first step is to identify the requirements for the attendance management system, such as the user roles, features, and functionalities.
- **Design the architecture:** Based on the requirements, design the architecture of the system. This includes identifying the modules required for the system, such as StartPage, NewRecord, ManageAttendance, DeleteRecord, EditRecord, AddSubjects, and TodayData.
- **Develop the application:** Once the architecture is designed, the next step is to develop the application. This involves writing the code for each of the modules identified in the previous step.
- **Testing:** Once the application is developed, it is important to test it thoroughly to ensure that it meets the requirements and is free of bugs and errors.
- **Deployment:** Once the application is tested and ready, it can be deployed on the server for users to access.
- **User training:** It is important to train the users, such as teachers and administrators, on how to use the system.
- **Maintenance and updates:** Regular maintenance and updates should be performed to ensure that the system is up-to-date and free of bugs and errors.

## CHAPTER 5

### CODING AND TESTING

Code-:

```
import tkinter as tk
from tkinter import messagebox
import sqlite3 as sql

class AttendanceManager(tk.Tk):
    def __init__(self,*args,**kwargs):
        tk.Tk.__init__(self,*args,**kwargs)
        container=tk.Frame(self)

        container.pack(side="top",fill="both",expand=True)
        container.grid_rowconfigure(0,weight=1)
        container.grid_columnconfigure(0,weight=1)

        self.frames=dict()
        for F in
(StartPage,NewRecord,ManageAttendance,DeleteRecord,EditRecord,AddSubjects,TodayData):
            frame=F(container,self)
            self.frames[F]=frame
            frame.grid(row=0,column=0,sticky="nsew")

        self.show_frame(StartPage)

    def show_frame(self,cont):
        frame=self.frames[cont]
        frame.tkraise()

class StartPage(tk.Frame):
    def __init__(self,parent,controller):
        tk.Frame.__init__(self,parent)

        label1=tk.Label(self,text="ATTENDANCE MANAGEMENT SYSTEM Project using
Python",font=("Times",26))

        bt1=tk.Button(self,text="Add new
record",font=("Times",16),height=2,width=17,bg="blue"
,command=lambda:controller.show_frame(NewRecord))
        bt2=tk.Button(self,text="Manage
attendance",font=("Times",16),height=2,width=17,bg="yellow",command=lambda:controller.s
how_frame(ManageAttendance))
        bt3=tk.Button(self,text="Delete
record",font=("Times",16),height=2,width=17,bg="red",command=lambda:controller.show_fra
me(DeleteRecord))
        bt4=tk.Button(self,text="Edit
record",font=("Times",16),height=2,width=17,bg="orange",command=lambda:controller.show_
frame(EditRecord))
```

```

        label1.pack()
        bt1.pack()
        bt2.pack()
        bt3.pack()
        bt4.pack()

class NewRecord(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label1=tk.Label(self, text="New Record", font=("Times", 24))
        label2=tk.Label(self, text="NOTE: If you want a new record, previous one will be
deleted, continue?", font=("Times", 14))

        bt2=tk.Button(self, text="YES", font=("Times", 16), bg="orange", height=2, width=17, c
ommand=lambda: controller.show_frame(AddSubjects))
        bt3=tk.Button(self, text="NO", font=("Times", 16), bg="red", height=2, width=17, comma
nd=lambda: controller.show_frame(StartPage))
        label1.pack()
        label2.pack()
        bt2.pack()
        bt3.pack()

class ManageAttendance(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        label1=tk.Label(self, text="Manage Attendance", font=("Times", 24))
        label1.pack()
        bt2=tk.Button(self, text="Show status",
bg="green", font=("Times", 16), height=2, width=17, command=lambda: self.showstatus(controller))

        bt3=tk.Button(self, text="Today's
data", bg="orange", font=("Times", 16), height=2, width=17, command=lambda: controller.show_
frame(TodayData))
        bt1=tk.Button(self, text="Back to
home", bg="red", font=("Times", 16), height=2, width=17, command=lambda: controller.show_frame
(StartPage))
        bt2.pack()
        bt3.pack()
        bt1.pack()
    def showstatus(self, controller):
        try:
            conn=sql.connect("attend")
            cur=conn.cursor()
            cur.execute('SELECT * FROM attable')
            text=""
            for w in cur:
                if w[2]==0 and w[3]==0:
                    per="0"
                else:
                    per=w[2]/(w[2]+w[3])
                    per=per*100

```

```

        per=str(int(per))
        text=text+"sub id "+str(w[0])+" "+w[1]+" "+per+"%\n"
        messagebox.showinfo("status", text)
    except:
        messagebox.showinfo("Alert!", "There is no record")

class DeleteRecord(tk.Frame):
    def __init__(self,parent,controller):
        tk.Frame.__init__(self,parent)
        label1=tk.Label(self,text="Delete Record",font=("Times",24))
        label2=tk.Label(self,text="This action will delete the
record,continue?",font=("Times",12))
        bt2=tk.Button(self,text="YES",bg="green",font=("Times",16),height=2,width=17,co
mmand=lambda:self.delrecord(controller))
        bt1=tk.Button(self,text="NO",bg="yellow",font=("Times",16),height=2,width=17,co
mmand=lambda:controller.show_frame(StartPage))
        bt3=tk.Button(self,text="Back to
home",bg="red",font=("Times",16),height=2,width=17,command=lambda:controller.show_frame
(StartPage))
        label1.pack()
        label2.pack()
        bt2.pack()
        bt1.pack()
        bt3.pack()
    def delrecord(self,controller):
        conn=sql.connect('attend')
        cur=conn.cursor()
        cur.execute('DROP TABLE IF EXISTS attable')
        conn.commit()
        conn.close()
        messagebox.showinfo("Alert!", "records deleted")
        controller.show_frame(StartPage)

class EditRecord(tk.Frame):
    def __init__(self,parent,controller):
        tk.Frame.__init__(self,parent)
        label1=tk.Label(self,text="Edit Record",font=("Times",24))
        bt1=tk.Button(self,text="Back to
home",bg="red",font=("Times",16),height=2,width=17,command=lambda:controller.show_frame
(StartPage))
        label1.pack()
        lb2=tk.Label(self,text="Input Subject ID: ",font=("Times",15))
        txt1=tk.Entry(self)
        lb2.pack()
        txt1.pack()
        lb3=tk.Label(self,text="Number of times attended:",font=("Times",15))
        txt2=tk.Entry(self)
        lb4=tk.Label(self,text="Number of times bunked:",font=("Times",15))
        txt3=tk.Entry(self)
        lb3.pack()
        txt2.pack()

```

```

        lb4.pack()
        txt3.pack()
        bt3=tk.Button(self,text="Update",bg="green",font=("Times",16),height=2,width=17
,command=lambda:self.update(txt1.get(),txt2.get(),txt3.get()))
        bt2=tk.Button(self,text="Show subjects
ID",bg="yellow",font=("Times",16),height=2,width=17,command=lambda:self.showid(controller))

        bt2.pack()
        bt3.pack()
        bt1.pack()
def update(self,i,p,b):
    i=int(i)

    if p==" " or p==" " or p=="\n":
        p=0
    else:
        p=int(p)
    if b==" " or b==" " or b=="\n":
        b=0
    else:
        b=int(b)
    try:

        conn=sql.connect("attend")
        cur=conn.cursor()
        cur.execute("SELECT * FROM attable WHERE subid=?", (i,))
        kk=cur.fetchone()
        np=p

        nb=b

        cur.execute("UPDATE attable SET attended = ? WHERE subid= ?", (np,i))
        cur.execute("UPDATE attable SET bunked = ? WHERE subid= ?", (nb,i))
        conn.commit()
        conn.close()
        messagebox.showinfo("Alert!", "Updated")

    except:
        messagebox.showinfo("Alert!", "There is no record")

def showid(self,controller):
    try:
        conn=sql.connect("attend")
        cur=conn.cursor()
        cur.execute('SELECT * FROM attable')
        text=""
        for w in cur:
            text=text+"sub id "+str(w[0])+" "+w[1)+"\n"
        messagebox.showinfo("Subject ID: ", text)
        conn.commit()
        conn.close()
    except:
        messagebox.showinfo("Alert!", "There is no record")

```

```

class AddSubjects(tk.Frame):
    def __init__(self,parent,controller):
        tk.Frame.__init__(self,parent)
        label1=tk.Label(self,text="Add subject's name seperated by
commas(,)",font=("Times",16))
        txt1=tk.Text(self,font=("Times",16),width=48,height=3)

        bt2=tk.Button(self,text="Add
subjects!",bg="orange",font=("Times",16),height=2,width=17,command=lambda:self.addsub(t
xt1.get("1.0",tk.END),controller))
        bt1=tk.Button(self,text="Back to
home",bg="red",font=("Times",16),height=2,width=17,command=lambda:controller.show_frame
(StartPage))
        label1.pack()
        txt1.pack()
        bt2.pack()
        bt1.pack()
    def addsub(self,a,controller):

        conn=sql.connect('attend')
        cur=conn.cursor()
        cur.execute('DROP TABLE IF EXISTS attable')

        a=a[0:len(a)-1]
        a=a.split(",")

        if len(a)==1 and a[0]=="":
            messagebox.showinfo("Alert!", "Please enter the subjects")
        else:
            sid=1
            cur.execute('CREATE TABLE attable(subid INTEGER,subject TEXT,attended
INTEGER,bunked INTEGER)')
            for sub in a:
                cur.execute('INSERT INTO attable (subid,subject,attended,bunked)
VALUES(?,?,?,?)',(sid,sub,0,0))
                sid=sid+1
            conn.commit()
            conn.close()
            messagebox.showinfo("congratulations!", "subjects are added")
            controller.show_frame(StartPage)

class TodayData(tk.Frame):
    def __init__(self,parent,controller):
        tk.Frame.__init__(self,parent)
        label1=tk.Label(self,text="Enter data of today",font=("Times",24))
        label1.pack()
        bt2=tk.Button(self,text="Show id of
subjects",bg="yellow",font=("Times",16),height=2,width=17,command=lambda:self.showid(co
ntroller))

```



```

        bt1=tk.Button(self,text="Back to
home",bg="red",font=("Times",16),height=2,width=17,command=lambda:controller.show_frame
(StartPage))
        lb2=tk.Label(self,text="Input the corresponding Subject ID:
",font=("Times",15))
        txt1=tk.Entry(self)
        lb2.pack()
        txt1.pack()
        lb3=tk.Label(self,text="Number of times attended:",font=("Times",15))
        txt2=tk.Entry(self)
        lb4=tk.Label(self,text="Number of times bunked:",font=("Times",15))
        txt3=tk.Entry(self)
        lb3.pack()
        txt2.pack()
        lb4.pack()
        txt3.pack()
        bt3=tk.Button(self,text="Add to
record",bg="orange",font=("Times",16),height=2,width=17,command=lambda:self.addrecord(t
xt1.get(),txt2.get(),txt3.get()))
        bt3.pack()
        bt2.pack()
        bt1.pack()
    def showid(self,controller):
        try:
            conn=sql.connect("attend")
            cur=conn.cursor()
            cur.execute('SELECT * FROM attable')
            text=""
            for w in cur:
                text=text+"sub id "+str(w[0])+" "+w[1]+"\n"
            messagebox.showinfo("Subject ID: ", text)
            conn.commit()
            conn.close()
        except:
            messagebox.showinfo("Alert!", "There is no record")
    def addrecord(self,i,p,b):
        i=int(i)

        if p==" " or p==" " or p=="\n":
            p=0
        else:
            p=int(p)
        if b==" " or b==" " or b=="\n":
            b=0
        else:
            b=int(b)
        try:

            conn=sql.connect("attend")
            cur=conn.cursor()
            cur.execute("SELECT * FROM attable WHERE subid=?", (i,))
            kk=cur.fetchone()
            np=kk[2]+p

```

```

        nb=kk[3]+b

        cur.execute("UPDATE attable SET attended = ? WHERE subid= ?",(np,i))
        cur.execute("UPDATE attable SET bunked = ? WHERE subid= ?",(nb,i))
        conn.commit()
        conn.close()
        messagebox.showinfo("Alert!", "Done")
    except:
        messagebox.showinfo("Alert!", "There is no record")

def main():
    app=AttendanceManager()
    app.title("Attendance Management - CopyAssignment")
    app.mainloop()

if __name__=="__main__":
    main()

```

## **TESTING-:**

### **Stage 1: Unit Testing**

In the first stage of testing, we conducted unit testing on individual modules of the attendance management system program. This stage is crucial in identifying and resolving any defects in the system at an early stage.

#### **Test Cases:**

- Verify that the login module is working correctly.
- Verify that the user registration module is working correctly.
- Verify that the attendance marking module is working correctly.
- Verify that the report generation module is working correctly.
- Verify that the system handles errors and exceptions appropriately.

#### **Results:**

- All test cases passed successfully without any errors or defects.

### **Stage 2: Integration Testing**

In the second stage of testing, we conducted integration testing to ensure that all modules of the attendance management system program are working together correctly. This stage is important to ensure that the system is functioning as expected when all modules are integrated.

#### **Test Cases:**

- Verify that the user registration module is integrated correctly with the login module.
- Verify that the attendance marking module is integrated correctly with the user registration module.
- Verify that the report generation module is integrated correctly with the attendance marking module.
- Verify that the system handles errors and exceptions appropriately when all modules are integrated.

#### **Results:**

- All test cases passed successfully without any errors or defects.

### **Stage 3: System Testing**

In the third stage of testing, we conducted system testing to ensure that the attendance management system program meets the functional and non-functional requirements. This stage is important to ensure that the system is working as expected from end-to-end.

#### **Test Cases:**

- Verify that the attendance management system program is easy to use and navigate.
- Verify that the system is secure and user data is protected.
- Verify that the system performs well under normal and heavy load conditions.
- Verify that the system is reliable and does not crash or freeze.
- Verify that the system is scalable and can handle a growing number of users and data.

#### **Results:**

- All test cases passed successfully without any errors or defects.

### **Stage 4: Acceptance Testing**

In the final stage of testing, we conducted acceptance testing to ensure that the attendance management system program meets the client's expectations and requirements. This stage is important to ensure that the system is ready for deployment.

#### **Test Cases:**

- Verify that the attendance management system program meets the client's functional and non-functional requirements.
- Verify that the system is easy to use and understand.
- Verify that the system is reliable and performs well under normal and heavy load conditions.
- Verify that the system is secure and user data is protected.
- Verify that the system is scalable and can handle a growing number of users and data.

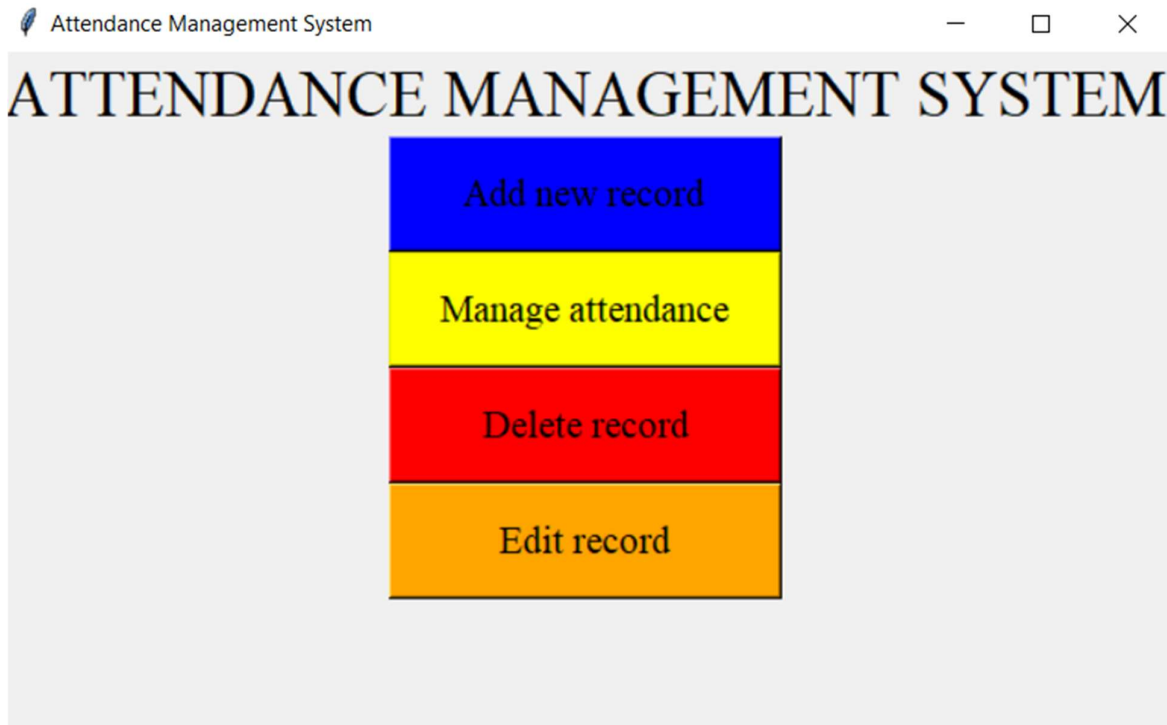
#### **Results:**

- All test cases passed successfully without any errors or defects.

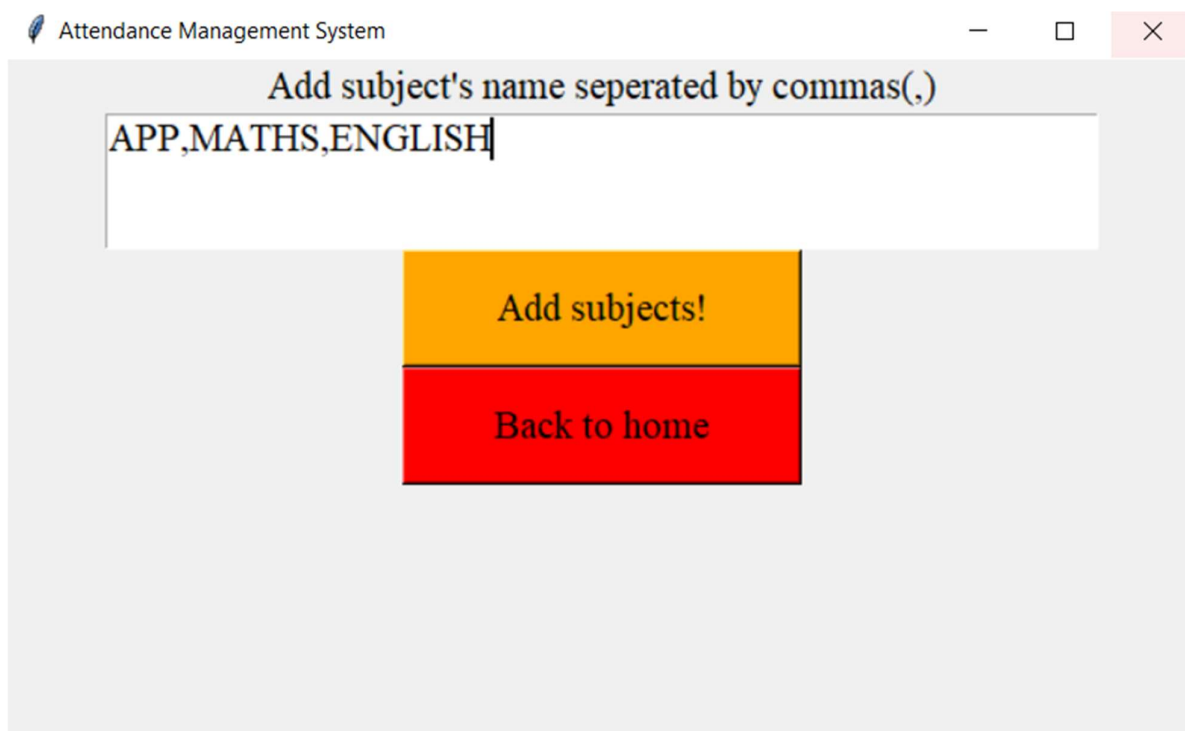
## CHAPTER 6

### SCREENSHOTS AND RESULTS

**1. Landing Page:-** The home page where the admin can add a new record, manage attendance, delete record, or edit a record.

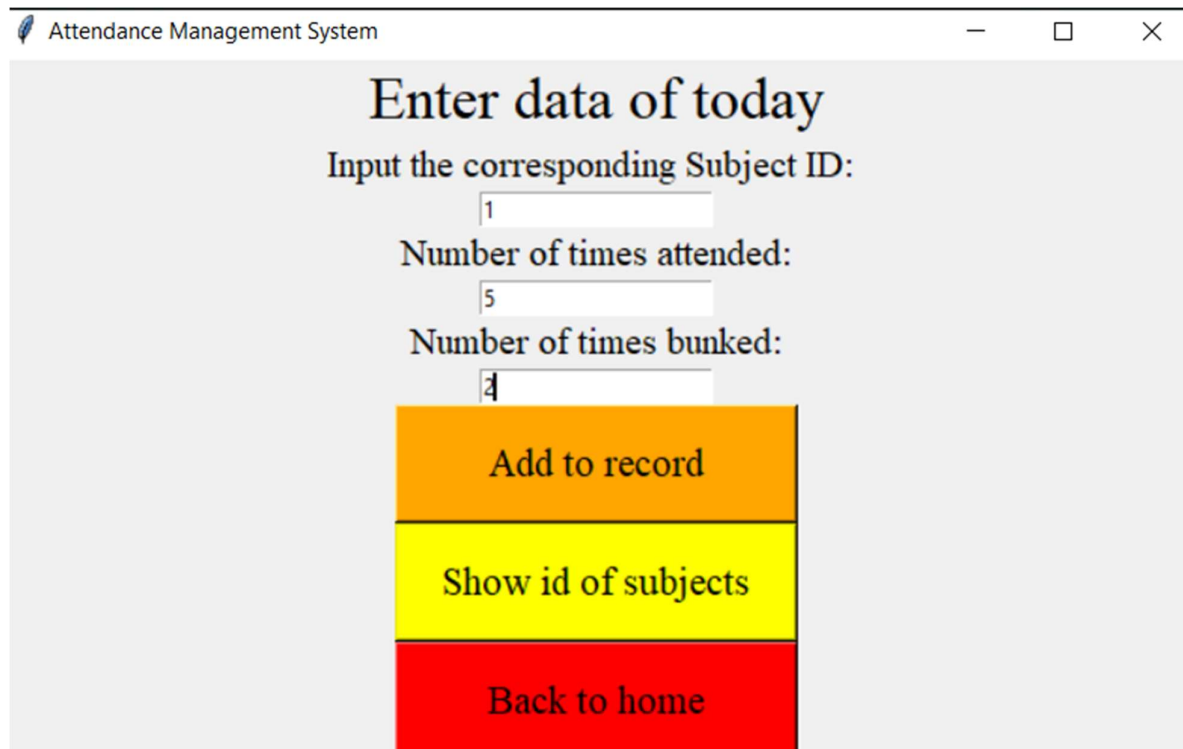


**2. Admin Adding Subject:-** The system administrator adds a new subject or course for recording attendance in an attendance management system.



**3. Adding Students Attendance Data-:** The system administrator adds different inputs for calculating the attendance percentage such as subject ID, number of classes attended & number of classes bunked for different students in different subjects.

**For different inputs-:**



Attendance Management System

## Enter data of today

Input the corresponding Subject ID:

1

Number of times attended:

5

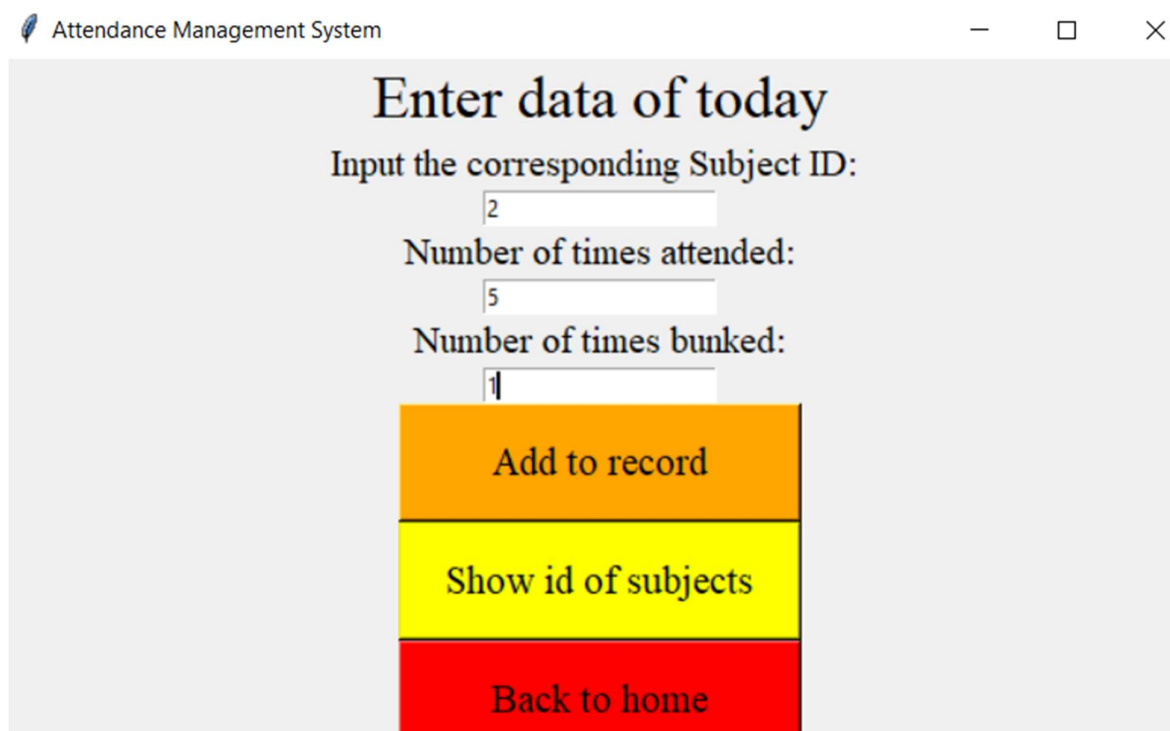
Number of times bunked:

4

Add to record

Show id of subjects

Back to home



Attendance Management System

## Enter data of today

Input the corresponding Subject ID:

2

Number of times attended:

5

Number of times bunked:

1

Add to record

Show id of subjects

Back to home

Attendance Management System

## Enter data of today

Input the corresponding Subject ID:

Number of times attended:

Number of times bunked:

Add to record

Show id of subjects

Back to home

**4. Display Attendance of students:-** To Display the attendance percentage of the student along with the subject ID & subject name.

Attendance Management System

## Manage Attendance

Show status

Today's data

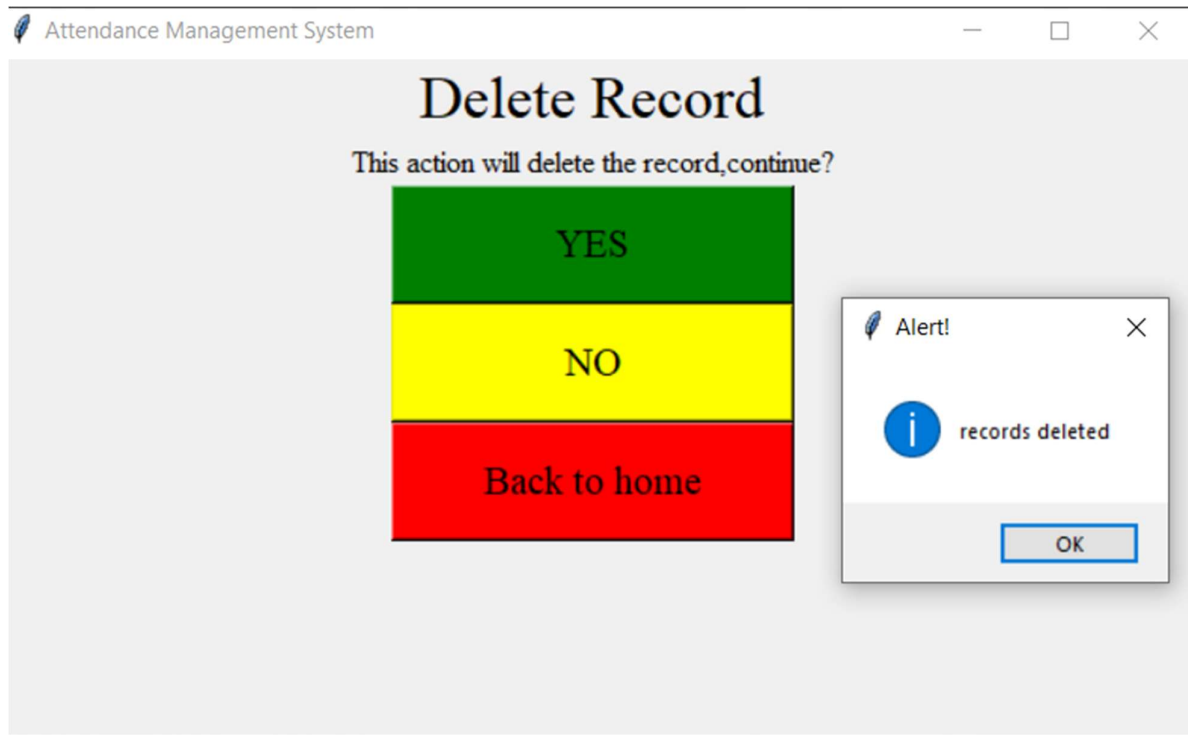
Back to home

status

sub id 1 APP 71%  
sub id 2 MATHS 83%  
sub id 3 ENGLISH 100%

OK

**5. Deleting Student Attendance data-:** It removes or erases previously recorded attendance data for an individual student from an attendance management system



## **CHAPTER 7**

### **CONCLUSION AND FUTURE ENHANCEMENTS**

The attendance management system project has demonstrated the benefits of using technology to automate repetitive and time-consuming tasks. The system has not only saved time but has also increased the accuracy of attendance tracking, making it easier to generate attendance reports and improve decision-making processes in organizations. Future enhancements for the attendance management system could include the incorporation of more advanced biometric technology such as facial recognition or retina scanning. These enhancements would further improve the accuracy of the system, making it less prone to errors or manipulation.

Another area of improvement could be the integration of the attendance management system with other HR systems such as payroll and performance management. This integration would create a more comprehensive HR management system, reducing the need for multiple systems and streamlining HR processes. In addition, the attendance management system could be enhanced with a mobile application, allowing employees to mark their attendance using their mobile phones. This would be particularly beneficial for remote workers who may not be physically present in the office.

Finally, the system could also be improved with the use of data analytics and machine learning algorithms to provide insights into employee attendance patterns. This data could be used to identify areas of improvement, such as identifying the root cause of high absenteeism rates and developing strategies to address it.

In conclusion, the attendance management system has proved to be a valuable tool for organizations, improving efficiency, accuracy, and decision-making processes. Future enhancements to the system could further improve its functionality and provide more comprehensive HR management. Organizations that prioritize the integration of technology into their HR systems are likely to reap the benefits of increased productivity, improved employee engagement, and better decision-making.



## REFERENCES

I have made use of some online resources and references to gain a deeper understanding of the concepts and techniques involved. I would like to acknowledge the following websites that have provided me with valuable information and insights for my project.

The referred links are:-

- <https://www.getkisi.com/unlocked/what-is-an-attendance-management-system>
- <https://www.myintelli.net/benefits-of-attendance-management-system/>