Name: AMEENA PARAYANGATT

Email: anuparayangatt25@gmail.com

STUDENT RESULT MANAGEMENT SYSTEM

ABSTRACT

Nowadays, the advancement of technology and the effect of computers and the internet had an impact on a number of different industries. And today, computers are used to perform practically all tasks. The key to being in business is to swiftly transform knowledge into a product that customers desire. A student result management system is an application designed to organize student academic records, including course, marks obtained and other related information of the student. The system allows teachers and administrators to enter, organize, and report student performance data in a centralized location. The student result management system typically includes several modules that are integrated to provide a complete solution for organizing student records. Some of the modules are:

1. **Student Profile Management**: This module organizes student information such as personal details, academic records, attendance records, and contact information.
2. **Course Management**: This module allows teachers to create, manage, and update course details, including course schedules, course descriptions, and course materials.
3. **Result Management**: This module organizes the grades and marks of students for each course.
4. **Report Management**: This module generates reports on student performance, such as grade reports, attendance reports, and progress reports, and can be customized to suit the needs of the school or institution.

By analysing the result status and applying the standard calculation followed by the University the result is displayed with individual scores and the equivalent percentage. The system is intended for the student. The student can login through their login id and password to check their respective results.

Overall, a student result management system streamlines the process of organizing student records and allows teachers and administrators to make informed decisions based on accurate data.

CONTENTS

CHAPTER 1

1.1 INTRODUCTION

The Student Result Management System (STUDENT RESULT MANAGEMENT SYSTEM) is a web-based tool that primarily focuses on delivering results to students and instructors. The student checks their separate outcomes using their roll number. It is easy for students to retrieve their results through the college website, and it is easier for faculty to assess the pass and fail rates of a given subject. Student, Faculty, and Administrator are the three components that make up the system. The students may examine his results by entering their roll number, and the faculty can view the analysis of pass and failure counts in the selected topic by entering their name, roll number, course. The administrator is responsible for creating and maintaining any current score.

1.2 AIMS AND OBJECTIVES

The main aim of a Student Result Management System is to automate the process of organizing student records and results. This includes storing, retrieving, updating, and deleting student records, as well as calculating and organizing their academic results. The objectives of a Student Result Management System include:

- **Efficient Management:** To provide an efficient and effective system for organizing student records and results, eliminating the need for manual processing and reducing the time and effort required for organizing student records.
- **Accuracy:** To ensure accuracy and reliability in organizing student records and results, reducing the likelihood of errors and discrepancies in the records.
- **Accessibility:** To provide easy access to student records and results for authorized personnel, such as teachers, administrators, and parents, through a secure and user-friendly interface.
- **Automation:** To automate the process of generating reports and calculating results, reducing the workload of teachers and administrators.
- **Transparency:** To provide transparency in the evaluation process, allowing students to view their results and enabling parents to monitor the progress of their children.
- **Scalability:** To design a system that can accommodate a large number of students and records, ensuring that it can be easily scaled to meet the needs of growing institutions.

Overall, the aim and objectives of a Student Result Management System are to improve the management of student records and results, while reducing the workload of teachers and administrators, and providing easy access and transparency to all stakeholders.

1.3 SCOPE OF SYSTEM

The scope of a Student Result Management System (SRMS) typically involves organizing and processing student academic records, including the tracking of student performance and other related information. More specifically, the scope of an SRMS may include the following functionalities:

Student Information Management: This includes capturing and maintaining basic information about students, such as their names, addresses, and contact details.

Course and Curriculum Management: This includes managing information related to the courses and programs offered by the institution, including course schedules, syllabi, and curriculum requirements.

Marks and Assessment: This involves capturing and managing information related to student performance, including grading, assessments, and feedback.

Reporting and Analytics: This involves generating grade reports and analytics related to student performance and academic progress.

Communication and Collaboration: This involves enabling communication and collaboration among stakeholders, such as teachers, students, parents, and administrators, through features such as messaging, notifications, and alerts.

CHAPTER 2

2.1 PROBLEM STATEMENT

A student result management system is a software application that helps educational institutions manage student data related to their academic progress, including grades, attendance, and other relevant information. The system helps administrators, teachers, and students' access and manage data related to student performance, enabling them to make informed decisions and take appropriate actions.

The key features of a student result management system include student registration, course management, grade management, attendance management, and report generation. The system should be able to manage student information, including personal details, academic records, and attendance records. Teachers should be able to enter grades for each student in their class, and the system should automatically calculate the students' overall grades and generate reports. The system should also generate attendance reports for each student in each course and provide alerts if a student is absent frequently.

The student result management system should be easy to use, secure, and scalable to accommodate growing educational institutions. It should provide access to authorized personnel, such as teachers, administrators, and students, and ensure data privacy and security. The system should be able to integrate with other education-related applications, such as learning management systems, student information systems, and assessment systems.

Overall, a student result management system is a critical tool for educational institutions to manage student data and ensure student success. It enables administrators, teachers, and students to monitor and manage academic performance, leading to better outcomes and improved student achievement.

2.2 PROPOSED SOLUTION

To develop a student result management system, a software application can be built using a combination of programming languages, frameworks, and database technologies. Here is a proposed solution for building a student result management system:

1. **Identify user requirements:** The first step is to identify the user requirements for the system. This includes identifying the stakeholders (e.g., administrators, teachers, students) and their needs. For example, administrators may need to manage student records, while teachers may need to enter grades and attendance.
2. **Develop a database schema:** The next step is to develop a database schema that represents the various entities in the system, such as students, courses, grades, and attendance. This can be done using a database management system such as MySQL or PostgreSQL.

3. **Develop a user interface:** The user interface can be developed using a web framework such as Flask or Django, which provides templates for creating web pages. The interface should allow users to access the various functions of the system, such as entering grades, viewing attendance records, and generating reports.
4. **Develop the backend:** The backend can be developed using a programming language such as Python, which provides libraries for working with databases and web frameworks. The backend should include functions for adding and retrieving data from the database, as well as functions for performing calculations, such as calculating grades.
5. **Implement security measures:** To ensure data privacy and security, the system should implement measures such as user authentication and authorization, encryption of sensitive data, and secure connections.
6. **Test and deploy the system:** The system should be tested thoroughly to ensure that it meets the user requirements and functions correctly. Once the system has passed testing, it can be deployed on a server or cloud platform for use by the stakeholders.
7. **Provide maintenance and support:** The system should be maintained and supported by a team of developers to ensure that it remains functional and up-to-date. This includes fixing bugs, updating the system to new versions of software, and providing user support.

Overall, the proposed solution for a student result management system involves identifying user requirements, developing a database schema, developing a user interface and backend, implementing security measures, testing and deploying the system, and providing maintenance and support.

CHAPTER 3

3.1 SYSTEM ARCHITECTURE

We can generate proper program steps to develop a student result management system. The steps may differs depending upon the details of system, such as the programming language used, database management system, and web framework used to develop the application.

1) Create a folder “python” in the computer, in which we can add the necessary files and images to create the student result management system.
2) Open this folder in “pychram” to type the code for creating the student result management system.
3) Create file “login.py” and form the login page of student result management system. Give separate columns for giving the “username” and “password” in a frame. Herr, add two buttons “Login In” and “Forget Password”. In another frame, ask the question “Don’t have an account?”, this option is for the users who have not any account in student result management system yet. And give here a button “Sign Up” for these users to create one.
4) Create file “create_db.py” for adding each database created for the creation of student result management system.
5) Create file “python project.py” in the above folder and give codes to form the main page of student result management system. Here, we can add menus; “course, student, result, view student results (report), logout, exit”.
6) Create file “course.py” in which we can add the necessary codes to add the “course name, duration of course, charges (fee of course), description about course”. And can add buttons for saving, updating, deleting and to clear the details in the page.
7) Create file “student.py” and can add the details of a student registered in student result management system. The details such as roll number, name, email, gender, address, state, city, pin code, date of birth (DOB), contact number, date of admission, course. Also added buttons for searching, saving, updating, deleting and to clear the details. Here, we can add selecting options in “gender” and “course” for an easier access.
8) Create file “result” to add the marks of a particular student. Add an option for selecting the student by roll number in the top of it. And create codes so that the student result management system itself fetch the name and course of the student for the roll number given. Here, then the instructor can add the marks obtained by the student and total marks of the exam by himself. Also give buttons “search”, “submit” and “clear”.
9) Create file “report.py”. Add the codes for the easiness of user to search the student by his/her roll number and so the student result management system itself will fetch the data of this particular student. The data fetched by student result management system are name, course, marks obtained, total marks of the exam. Add button “search”, “delete” and “clear”.
10) Add codes for button menu “logout” and “exit”.

These steps include four classes: **Student**, **Course**, **Result** and **Report**

The **student** class represents a single student and their academic information, including their name, roll number, email, gender, state, city, address, pin code, date of birth (DOB), contact number, admission date and course.

The **Course** class represents a course that students can enrol in, with details such as the course name, duration, charges (fee of course), description about the course.

The **Result** class represents marks obtained by a student in a particular course along with the total marks for the exam.

The **Report** class represents a record of a student's performance in a particular course along with the percentage of passing the exam.

Overall, this flow chart diagram provides a high-level view of the student result management system and its components.

3.2 SOURCE CODE OF THE ALGORITHM

1. Source code from file "python project.py"

```
from tkinter import *
from course import CourseClass
from student import studentClass
from result import ResultClass
from report import ReportClass
from tkinter import messagebox
import os

class RMS:
    def __init__(self,root):
        self.root=root
        self.root.title("Student Result Management System")
        self.root.geometry("1400x750+0+0")
        self.root.config(bg="white")
        # ===title=====

        title = Label(self.root,text="Student Result Management
System",padx=10,compound=LEFT,font=("goudy old
style",18,"bold"),bg="#033054",fg="white").place(x=0,y=0,relwidth=1,height=50
)
        # ===Menu=====
        M_Frame=LabelFrame(self.root, text="Menus",font=("times new
roman",15),bg="blue")
```

```
        M_Frame.place(x=10,y=80,width=1340,height=60)

        btn_course=Button(M_Frame,text="Course",font=("goudy old
style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add_cou
rse).place(x=20,y=5,width=200,height=40)
        btn_student=Button(M_Frame,text="Student",font=("goudy old
style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add_stu
dent).place(x=240,y=5,width=200,height=40)
        btn_result=Button(M_Frame,text="Result",font=("goudy old
style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add_res
ult).place(x=460,y=5,width=200,height=40)
        btn_view=Button(M_Frame,text="View Student Results",font=("goudy old
style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add_rep
ort).place(x=680,y=5,width=200,height=40)
        btn_logout=Button(M_Frame,text="Logout",font=("goudy old
style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.logout)
.place(x=900,y=5,width=200,height=40)
        btn_exit=Button(M_Frame,text="Exit",font=("goudy old
style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.exit_).
place(x=1120,y=5,width=200,height=40)

        # ===footer====

        footer=Label(self.root,text="STUDENT RESULT MANAGEMENT SYSTEM-Student
Result Management System\nContact Us for any Technical Issue:student result
management system@gmail.com",font=("goudy old
style",12),bg="#033054",fg="white").place(x=0,y=630,relwidth=1,height=50)
        # ===Update details=====
        self.lbl_course=Label(self.root,text="Total Course\n[0]",font=("goudy
old style",20), bd=10, relief=RIDGE, bg="#0b5377",
fg="pink").place(x=400,y=530,width=300,height=80)
        self.lbl_student=Label(self.root, text="Total Student\n[0]",
font=("goudy old style", 20),  bd=10, relief=RIDGE,
bg="#0b5377",fg="pink").place(x=700, y=530, width=300, height=80)
        self.lbl_result=Label(self.root, text="Total Result\n[0]",
font=("goudy old style", 20), bd=10, relief=RIDGE,
bg="#0b5377",fg="pink").place(x=1000, y=530, width=300, height=80)

    def add_course(self):
        self.new_win=Toplevel(self.root)
        self.new_obj=CourseClass(self.new_win)

    def add_student(self):
        self.new_win=Toplevel(self.root)
        self.new_obj=studentClass(self.new_win)

    def add_result(self):
        self.new_win=Toplevel(self.root)
        self.new_obj=ResultClass(self.new_win)

    def add_report(self):
        self.new_win=Toplevel(self.root)
        self.new_obj=ReportClass(self.new_win)
    def logout(self):
        op=messagebox.askyesno("confirm","Do you really want to
logout?",parent=self.root)
        if op==True:
            self.root.destroy()
            os.system("python login.py")

    def exit_(self):
        op=messagebox.askyesno("confirm", "Do you really want to exit?",
parent=self.root)
        if op==True:
            self.root.destroy()
            os.system("python login.py")


if __name__=="__main__":
     root=Tk()
     obj=RMS(root)
     root.mainloop()
```

2. Source code from file "course.py"

```
from tkinter import *
from tkinter import ttk, messagebox
import sqlite3


class CourseClass:
    def __init__(self, root):
        self.root = root
        self.root.title("Student Result Management System")
        self.root.geometry("1200x480+80+170")
        self.root.config(bg="white")
        self.root.focus_force()
        # ===title=====

        title = Label(self.root, text="Manage Course Details", font=("goudy
old style", 18, "bold"), bg="#033054",
                      fg="white").place(x=10, y=15, width=1180, height=35)

        # ========Variables============
        self.var_course = StringVar()
        self.var_duration = StringVar()
        self.var_charges = StringVar()
        # ========Widgets=============
        lbl_courseName = Label(self.root, text="Course Name", font=("goudy
old style", 15, 'bold'), bg='white').place(
            x=10, y=60)
        lbl_duration = Label(self.root, text="Duration", font=("goudy old
style", 15, 'bold'), bg='white').place(x=10,

y=100)
        lbl_charges = Label(self.root, text="Charges", font=("goudy old
style", 15, 'bold'), bg='white').place(x=10,

y=140)
        lbl_description = Label(self.root, text="Description", font=("goudy
old style", 15, 'bold'), bg='white').place(
            x=10, y=180)

        # =====Entry fields========
        self.txt_courseName = Entry(self.root, textvariable=self.var_course,
font=("goudy old style", 15, 'bold'),
                                    bg='lightyellow')
        self.txt_courseName.place(x=150, y=60, width=200)
        txt_duration = Entry(self.root, textvariable=self.var_duration,
font=("goudy old style", 15, 'bold'),
                             bg='lightyellow').place(x=150, y=100, width=200)
        txt_charges = Entry(self.root, textvariable=self.var_charges,
font=("goudy old style", 15, 'bold'),
                            bg='lightyellow').place(x=150, y=140, width=200)
        self.txt_description = Text(self.root, font=("goudy old style", 15,
'bold'), bg='lightyellow')
        self.txt_description.place(x=150, y=180, width=500, height=130)

        # =====Buttons==========

        self.btn_add = Button(self.root, text='Save', font=("goudy old
style", 15, 'bold'), bg="#2196f3", fg='white',
                              cursor="hand2", command=self.add)
        self.btn_add.place(x=150, y=400, width=110, height=40)
        self.btn_update = Button(self.root, text='Update', font=("goudy old
style", 15, 'bold'), bg="#4caf50",
                                 fg='white', cursor="hand2",
command=self.update)
        self.btn_update.place(x=270, y=400, width=110, height=40)
        self.btn_delete = Button(self.root, text='Delete', font=("goudy old
style", 15, 'bold'), bg="#f44336",
                                 fg='white', cursor="hand2",
command=self.delete)
        self.btn_delete.place(x=390, y=400, width=110, height=40)
        self.btn_clear = Button(self.root, text='Clear', font=("goudy old
style", 15, 'bold'), bg="#607d8b", fg='white',
                                cursor="hand2", command=self.clear)
        self.btn_clear.place(x=510, y=400, width=110, height=40)

        # ======Search panel========

        self.var_search = StringVar()
        lbl_search_courseName = Label(self.root, text="Course Name",
font=("goudy old style", 15, 'bold'),
                                      bg='white').place(x=720, y=60)
        txt_search_courseName = Entry(self.root,
textvariable=self.var_course, font=("goudy old style", 15, 'bold'),
                                      bg='lightyellow').place(x=870, y=60,
width=180)
        btn_search = Button(self.root, text='Search', font=("goudy old
style", 15, 'bold'), bg="#03a9f4", fg='white',
                            cursor="hand2",
command=self.search).place(x=1070, y=60, width=120, height=28)

        # ========content=========
        self.C_Frame = Frame(self.root, bd=2, relief=RIDGE)
        self.C_Frame.place(x=720, y=100, width=470, height=340)
        scrolly = Scrollbar(self.C_Frame, orient=VERTICAL)
        scrollx = Scrollbar(self.C_Frame, orient=HORIZONTAL)
        self.CourseTable = ttk.Treeview(self.C_Frame, columns=("cid", "name",
"duration", "charges", "description"),
                                        xscrollcommand=scrollx.set,
yscrollcommand=scrolly.set)

        scrollx.pack(side=BOTTOM, fill=X)
        scrolly.pack(side=RIGHT, fill=Y)
        scrollx.config(command=self.CourseTable.xview)
        scrolly.config(command=self.CourseTable.yview)

        self.CourseTable.heading("cid", text="Course ID")
        self.CourseTable.heading("name", text="Name")
        self.CourseTable.heading("duration", text="Duration")
        self.CourseTable.heading("charges", text="Charges")
        self.CourseTable.heading("description", text="Description")
        self.CourseTable["show"] = 'headings'
        self.CourseTable.column("cid", width=60)
        self.CourseTable.column("name", width=100)
        self.CourseTable.column("duration", width=100)
        self.CourseTable.column("charges", width=90)
        self.CourseTable.column("description", width=145)
        self.CourseTable.pack(fill=BOTH, expand=1)
        self.CourseTable.bind("<ButtonRelease-1>", self.get_data)
        self.show()

    # =========================================
    def clear(self):
        self.show()
        self.var_course.set("")
        self.var_duration.set("")
        self.var_charges.set("")
        self.var_search.set("")
        self.txt_description.delete('1.0',END)
        self.txt_courseName.config(state=NORMAL)

    def delete(self):
        con = sqlite3.connect(database="rms.db")
        cur = con.cursor()
        try:
            if self.var_course.get() == "":
                messagebox.showerror("Error", "Course Name should be
required", parent=self.root)
            else:
                cur.execute("select * from course where name=?",
(self.var_course.get(),))
                row = cur.fetchone()
                if row == None:
                    messagebox.showerror("Error", "please select the course
from list first", parent=self.root)
                else:
                    op = messagebox.askyesno("Confirm", "Do you really want
to delete?", parent=self.root)
                    if op == True:
                        cur.execute("delete from course where name=?",
(self.var_course.get(),))
```

```
                        con.commit()
                        messagebox.showinfo("Delete", "Course deleted
successfully", parent=self.root)
                        self.clear()

        except Exception as ex:
            messagebox.showerror("Error", f"Error due to {str(ex)}")

    def get_data(self, ev):
        self.txt_courseName.config(state='readonly')
        self.txt_courseName
        r = self.CourseTable.focus()
        content = self.CourseTable.item(r)
        row = content["values"]
        # print(row)
        self.var_course.set(row[1])
        self.var_duration.set(row[2])
        self.var_charges.set(row[3])
        # self.var_course.set(row[4])
        self.txt_description.delete('1.0',END)
        self.txt_description.insert(END,row[4])

    def add(self):
        con = sqlite3.connect(database="rms.db")
        cur = con.cursor()
        try:
            if self.var_course.get() == "":
                messagebox.showerror("Error", "Course Name should be
required", parent=self.root)
            else:
                cur.execute("select * from course where name=?",
(self.var_course.get(),))
                row = cur.fetchone()
                if row != None:
                    messagebox.showerror("Error", "Course Name Already
Present", parent=self.root)
                else:
                    cur.execute("insert into course
(name,duration,charges,description) values(?,?,?,?)", (
                        self.var_course.get(),
                        self.var_duration.get(),
                        self.var_charges.get(),
                        self.txt_description.get("1.0", END)
                    ))
                    con.commit()
                    messagebox.showinfo("Success", "Course Added
Successfully", parent=self.root)
                    self.show()
        except Exception as ex:
            messagebox.showerror("Error", f"Error due to {str(ex)}")

    def update(self):
        con = sqlite3.connect(database="rms.db")
        cur = con.cursor()
        try:
            if self.var_course.get() == "":
                messagebox.showerror("Error", "Course Name should be
required", parent=self.root)
            else:
                cur.execute("select * from course where name=?",
(self.var_course.get(),))
                row = cur.fetchone()
                if row == None:
                    messagebox.showerror("Error", "Select course from list",
parent=self.root)
                else:
                    cur.execute("update course set
duration=?,charges=?,description=? where name=?", (
                        self.var_duration.get(),
                        self.var_charges.get(),
                        self.txt_description.get("1.0", END),
                        self.var_course.get()
                    ))
                    con.commit()
                    messagebox.showinfo("Success", "Course Updated
Successfully", parent=self.root)
                    self.show()
        except Exception as ex:
            messagebox.showerror("Error", f"Error due to {str(ex)}")

    def show(self):
        con = sqlite3.connect(database="rms.db")
        cur = con.cursor()
        try:
            cur.execute("select * from course")
            rows = cur.fetchall()
            self.CourseTable.delete(*self.CourseTable.get_children())
            for row in rows:
                self.CourseTable.insert('', END, values=row)
        except Exception as ex:
            messagebox.showerror("Error", f"Error due to {str(ex)}")

    def search(self):
        con = sqlite3.connect(database="rms.db")
        cur = con.cursor()
        try:
            cur.execute(f"select * from course where name LIKE
'%{self.var_search.get()}%'")
            rows = cur.fetchall()
            self.CourseTable.delete(*self.CourseTable.get_children())
            for row in rows:
                self.CourseTable.insert('', END, values=row)
        except Exception as ex:
            messagebox.showerror("Error", f"Error due to {str(ex)}")


if __name__ == "__main__":
    root = Tk()
    obj = CourseClass(root)
    root.mainloop()
```

3. Source code from file "student.py"

```
2. from tkinter import *
   from tkinter import ttk,messagebox
   import sqlite3

   class studentClass:
       def __init__(self,root):
           self.root=root
           self.root.title("Student Result Management System")
           self.root.geometry("1200x480+80+170")
           self.root.config(bg="white")
           self.root.focus_force()
           # ===title=====

           title=Label(self.root,text="Manage Student 
   Details",font=("goudy old 
   style",18,"bold"),bg="#033054",fg="white").place(x=10,y=15,width=1180,h
   eight=35)

           #========Variables============
           self.var_roll=StringVar()
           self.var_name=StringVar()
           self.var_email=StringVar()
           self.var_gender=StringVar()
           self.var_dob=StringVar()
           self.var_contact=StringVar()
           self.var_course=StringVar()
           self.var_admission=StringVar()
           self.var_state=StringVar()
           self.var_city=StringVar()
           self.var_pin=StringVar()

           #========Widgets=============
           #========column 1============
           lbl_rollno=Label(self.root,text="Roll No.",font=("goudy old 
   style",15,'bold'),bg='white').place(x=10,y=60)
           lbl_name=Label(self.root,text="Name",font=("goudy old 
   style",15,'bold'),bg='white').place(x=10,y=100)
           lbl_email=Label(self.root,text="Email",font=("goudy old 
   style",15,'bold'),bg='white').place(x=10,y=140)
           lbl_gender=Label(self.root,text="Gender",font=("goudy old 
   style",15,'bold'),bg='white').place(x=10,y=180)
           lbl_state=Label(self.root,text="State",font=("goudy old 
   style",15,'bold'),bg='white').place(x=10,y=220)

   txt_state=Entry(self.root,textvariable=self.var_state,font=("goudy old 
   style",15,'bold'),bg='lightyellow').place(x=150,y=220,width=150)

           lbl_city=Label(self.root,text="City",font=("goudy old 
   style",15,'bold'),bg='white').place(x=310,y=220)

   txt_city=Entry(self.root,textvariable=self.var_city,font=("goudy old 
   style",15,'bold'),bg='lightyellow').place(x=380,y=220,width=100)

           lbl_pin=Label(self.root,text="Pin",font=("goudy old 
   style",15,'bold'),bg='white').place(x=500,y=220)
```

```
        txt_pin=Entry(self.root,textvariable=self.var_pin,font=("goudy
old style",15,'bold'),bg='lightyellow').place(x=560,y=220,width=120)
        lbl_address=Label(self.root,text="Address",font=("goudy old
style",15,'bold'),bg='white').place(x=10,y=260)

        # =====Entry fields 1========

self.txt_roll=Entry(self.root,textvariable=self.var_roll,font=("goudy
old style",15,'bold'),bg='lightyellow')
        self.txt_roll.place(x=150,y=60,width=200)

txt_name=Entry(self.root,textvariable=self.var_name,font=("goudy old
style",15,'bold'),bg ='lightyellow').place(x=150,y=100,width=200)

txt_email=Entry(self.root,textvariable=self.var_email,font=("goudy old
style",15,'bold'),bg='lightyellow').place(x=150,y=140,width=200)

self.txt_gender=ttk.Combobox(self.root,textvariable=self.var_gender,val
ues=("select","Male","Female","Other"),font=("goudy old
style",15,'bold'),state='readonly',justify=CENTER)
        self.txt_gender.place(x=150,y=180,width=200)
        self.txt_gender.current(0)

        #===============column 2===============
        lbl_dob=Label(self.root,text="DOB",font=("goudy old
style",15,'bold'),bg='white').place(x=360,y=60)
        lbl_contact=Label(self.root,text="Contact",font=("goudy old
style",15,'bold'),bg='white').place(x=360,y=100)
        lbl_admission=Label(self.root,text="Admission",font=("goudy old
style",15,'bold'),bg='white').place(x=360,y=140)
        lbl_course=Label(self.root,text="Course",font=("goudy old
style",15,'bold'),bg='white').place(x=360,y=180)

        #==============Entry fields 2============
        self.course_list=[]
        #function-call to update the list
        self.fetch_course()
        txt_dob=Entry(self.root,textvariable=self.var_dob,font=("goudy
old style",15,'bold'),bg='lightyellow').place(x=480,y=60,width=200)

txt_contact=Entry(self.root,textvariable=self.var_contact,font=("goudy
old style",15,'bold'),bg='lightyellow').place(x=480,y=100,width=200)

txt_admissionn=Entry(self.root,textvariable=self.var_admission,font=("g
oudy old
style",15,'bold'),bg='lightyellow').place(x=480,y=140,width=200)

self.txt_course=ttk.Combobox(self.root,textvariable=self.var_course,val
ues=(self.course_list),font=("goudy old
style",15,'bold'),state='readonly',justify=CENTER)
        self.txt_course.place(x=480,y=180,width=200)
        self.txt_course.set("Select")

        #=================Text Address========
        self.txt_address=Text(self.root,font=("goudy old
style",15,'bold'),bg='lightyellow')
        self.txt_address.place(x=150,y=260,width=540,height=100)
        #=====Buttons==========

        self.btn_add=Button(self.root,text='Save',font=("goudy old 
style",15,'bold'),bg="#2196f3",fg='white',cursor="hand2",command=self.a
dd)
        self.btn_add.place(x=150,y=400,width=110,height=40)
        self.btn_update=Button(self.root,text='Update',font=("goudy old 
style",15,'bold'),bg="#4caf50",fg='white',cursor="hand2",command=self.u
pdate)
        self.btn_update.place(x=270,y=400,width=110,height=40)
        self.btn_delete=Button(self.root,text='Delete',font=("goudy old 
style",15,'bold'),bg="#f44336",fg='white',cursor="hand2",command=self.d
elete)
        self.btn_delete.place(x=390,y=400,width=110,height=40)
        self.btn_clear=Button(self.root,text='Clear',font=("goudy old 
style",15,'bold'),bg="#607d8b",fg='white',cursor="hand2",command=self.c
lear)
        self.btn_clear.place(x=510, y=400, width=110, height=40)

        #======Search panel========

        self.var_search=StringVar()
        lbl_search_roll=Label(self.root,text="Roll No.",font=("goudy 
old style",15,'bold'),bg='white').place(x=720,y=60)

txt_search_roll=Entry(self.root,textvariable=self.var_search,font=("gou
dy old style",15,'bold'),bg='lightyellow').place(x=870,y=60,width=180)
        btn_search=Button(self.root,text='Search',font=("goudy old 
style",15,'bold'),bg="#03a9f4",fg='white',cursor="hand2",command=self.s
earch).place(x=1070,y=60,width=120,height=28)

        #=========content==========
        self.C_Frame=Frame(self.root,bd=2,relief=RIDGE)
        self.C_Frame.place(x=720,y=100,width=470,height=340)

        scrolly=Scrollbar(self.C_Frame,orient=VERTICAL)
        scrollx=Scrollbar(self.C_Frame,orient=HORIZONTAL)

self.CourseTable=ttk.Treeview(self.C_Frame,columns=("roll","name","emai
l","gender","dob","contact","course","state","city","pin","address"),xs
crollcommand=scrollx.set,yscrollcommand=scrolly.set)

        scrollx.pack(side=BOTTOM,fill=X)
        scrolly.pack(side=RIGHT, fill=Y)
        scrollx.config(command=self.CourseTable.xview)
        scrolly.config(command=self.CourseTable.yview)

        self.CourseTable.heading("roll",text="Roll No.")
        self.CourseTable.heading("name",text="Name")
        self.CourseTable.heading("email",text="Email")
        self.CourseTable.heading("gender",text="Gender")
        self.CourseTable.heading("dob",text="D.O.B")
        self.CourseTable.heading("contact",text="Contact")

        self.CourseTable.heading("course",text="Course")
        self.CourseTable.heading("state",text="State")
        self.CourseTable.heading("city",text="City")
        self.CourseTable.heading("pin", text="PIN")
        self.CourseTable["show"]='headings'
        self.CourseTable.column("roll",width=100)
        self.CourseTable.column("name",width=100)
        self.CourseTable.column("email",width=100)
        self.CourseTable.column("gender",width=100)
        self.CourseTable.column("dob",width=100)
        self.CourseTable.column("contact",width=100)

        self.CourseTable.column("course",width=100)
        self.CourseTable.column("state",width=100)
        self.CourseTable.column("city",width=100)
        self.CourseTable.column("pin",width=100)
        self.CourseTable.column("address",width=200)
        self.CourseTable.pack(fill=BOTH,expand=1)
        self.CourseTable.bind("<ButtonRelease-1>",self.get_data)
        self.show()

        #==========================================
    def clear(self):
        self.show()
        self.var_roll.set("")
        self.var_name.set("")
        self.var_email.set("")
        self.var_gender.set("Select")
        self.var_dob.set("")
        self.var_contact.set("")
        self.var_admission.set("")
        self.var_course.set("Select")
        self.var_state.set("")
        self.var_city.set("")
        self.var_pin.set("")
        self.txt_address.delete("1.0",END)
        self.txt_roll.config(state=NORMAL)
        self.var_search.set("")

    def delete(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_roll.get()=="":
                messagebox.showerror("Error","roll Number should be 
required",parent=self.root)
            else:
                cur.execute("select * from student where 
roll=?",(self.var_roll.get(),))
                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","please select student 
from list first",parent=self.root)
                else:
                    op=messagebox.askyesno("Confirm","Do you really 
want to delete?",parent=self.root)
                    if op==True:
                        cur.execute("delete from student where 
roll=?",(self.var_roll.get(),))
```

```
                        con.commit()
                        messagebox.showinfo("Delete","Student deleted 
successfully",parent=self.root)
                        self.clear()

        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")

    def get_data(self,ev):
        self.txt_roll.config(state='readonly')
        r=self.CourseTable.focus()
        content=self.CourseTable.item(r)
        row=content["values"]
        self.var_roll.set(row[0])
        self.var_name.set(row[1])
        self.var_email.set(row[2])
        self.var_gender.set(row[3])
        self.var_dob.set(row[4])
        self.var_contact.set(row[5])
        self.var_admission.set(row[6])
        self.var_course.set(row[7])
        self.var_state.set(row[8])
        self.var_city.set(row[9])
        self.var_pin.set(row[10])
        self.txt_address.delete("1.0",END)
        self.txt_address.insert(END,row[11])

    def add(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_roll.get()=="":
                messagebox.showerror("Error","Roll Number should be 
required",parent=self.root)
            else:
                cur.execute("select * from student where 
roll=?",(self.var_roll.get(),))
                row=cur.fetchone()
                if row!=None:
                    messagebox.showerror("Error","Roll Number Already 
Present",parent=self.root)
                else:
                    cur.execute("insert into 
student(roll,name,email,gender,dob,contact,admission,course,state,city,
pin,address) values(?,?,?,?,?,?,?,?,?,?,?,?)",(
                        self.var_roll.get(),
                        self.var_name.get(),
                        self.var_email.get(),
                        self.var_gender.get(),
                        self.var_dob.get(),
                        self.var_contact.get(),
                        self.var_admission.get(),
                        self.var_course.get(),
                        self.var_state.get(),
                        self.var_city.get(),
                        self.var_pin.get(),
                        self.txt_address.get("1.0",END)
                     ))
                     con.commit()
                     messagebox.showinfo("Success","Student Added
 Successfully",parent=self.root)
                     self.show()
         except Exception as ex:
             messagebox.showerror("Error",f"Error due to {str(ex)}")

     def update(self):
         con=sqlite3.connect(database="rms.db")
         cur=con.cursor()
         try:
             if self.var_roll.get()=="":
                 messagebox.showerror("Error","Roll Number should be
 required",parent=self.root)
             else:
                 cur.execute("select * from student where
 roll=?",(self.var_roll.get(),))
                 row=cur.fetchone()
                 if row==None:
                     messagebox.showerror("Error","Select student from
 list",parent=self.root)
                 else:
                     cur.execute("update student set
 name=?,email=?,gender=?,dob=?,contact=?,admission=?,course=?,state=?,ci
 ty=?,pin=?,address=? where roll=?",(
                         self.var_name.get(),
                         self.var_email.get(),
                         self.var_gender.get(),
                         self.var_dob.get(),
                         self.var_contact.get(),
                         self.var_admission.get(),
                         self.var_course.get(),
                         self.var_state.get(),
                         self.var_city.get(),
                         self.var_pin.get(),
                         self.txt_address.get("1.0", END),
                         self.var_roll.get()
                     ))
                     con.commit()
                     messagebox.showinfo("Success","Student Updated
 Successfully",parent=self.root)
                     self.show()
         except Exception as ex:
             messagebox.showerror("Error",f"Error due to {str(ex)}")

     def show(self):
         con=sqlite3.connect(database="rms.db")
         cur=con.cursor()
         try:
             cur.execute("select * from student")
             rows=cur.fetchall()
             self.CourseTable.delete(*self.CourseTable.get_children())
             for row in rows:
                 self.CourseTable.insert('',END,values=row)
         except Exception as ex:
             messagebox.showerror("Error",f"Error due to {str(ex)}")
     def fetch_course(self):
         con=sqlite3.connect(database="rms.db")
         cur=con.cursor()
         try:
             cur.execute("select name from course")
             rows=cur.fetchall()
             if len(rows)>0:
                 for row in rows:
                    self.course_list.append(row[0])
         except Exception as ex:
                 messagebox.showerror("Error", f"Error due to
 {str(ex)}")

     def search(self):
         con=sqlite3.connect(database="rms.db")
         cur=con.cursor()
         try:
             cur.execute(f"select * from student where
 roll=?",(self.var_search.get(),))
             row=cur.fetchone()
             # print(row)
             if row!=None:

 self.CourseTable.delete(*self.CourseTable.get_children())
                  self.CourseTable.insert('',END,values=row)
             else:
                 messagebox.showerror("Error","No record
 found",parent=self.root)
         except Exception as ex:
             messagebox.showerror("Error",f"Error due to {str(ex)}")

 if __name__=="__main__":
     root=Tk()
     obj=studentClass(root)
     root.mainloop()
```

4. Source code from file “result.py”

```
from tkinter import *
from tkinter import ttk,messagebox
import sqlite3

class ResultClass:
    def __init__(self,root):
        self.root=root
        self.root.title("Student Result Management System")
        self.root.geometry("1200x480+80+170")
        self.root.config(bg="white")
        self.root.focus_force()
        # ===title=====
        title=Label(self.root,text="Add Student Results",font=("goudy old
style",18,"bold"),bg="blue",fg="white").place(x=10,y=15,width=1180,height=50)
        #============widgets======
        #===========variables=========
        self.var_roll=StringVar()
        self.var_name=StringVar()
        self.var_course=StringVar()
        self.var_marks=StringVar()
        self.var_full_marks=StringVar()
        self.roll_list=[]
        self.fetch_roll()


        lbl_select=Label(self.root,text="Select Student",font=("goudy old
style",15,'bold'),bg='white').place(x=50,y=100)
        lbl_name=Label(self.root,text="Name",font=("goudy old
style",15,'bold'),bg='white').place(x=50,y=160)
        lbl_course=Label(self.root,text="Course",font=("goudy old
style",15,'bold'),bg='white').place(x=50,y=220)
        lbl_marks=Label(self.root,text="Marks",font=("goudy old
style",15,'bold'),bg='white').place(x=50,y=280)
        lbl_full_marks=Label(self.root,text="Full Marks",font=("goudy old
style",15,'bold'),bg='white').place(x=50,y=340)


self.txt_student=ttk.Combobox(self.root,textvariable=self.var_roll,values=sel
f.roll_list,font=("goudy old
style",15,'bold'),state='readonly',justify=CENTER)
        self.txt_student.place(x=280,y=100,width=200)
        self.txt_student.set("Select")
        btn_search=Button(self.root,text='Search',font=("goudy old
style",15,'bold'),bg="#03a9f4",fg='white',cursor="hand2",command=self.search)
.place(x=500,y=100,width=100,height=28)

        txt_name=Entry(self.root,textvariable=self.var_name,font=("goudy old
style",20,'bold'),bg='lightyellow',state='readonly').place(x=280,y=160,width=
320)
        txt_course=Entry(self.root,textvariable=self.var_course,font=("goudy
old style",20,'bold'),bg='lightyellow').place(x=280,y=220,width=320)
        txt_marks=Entry(self.root,textvariable=self.var_marks,font=("goudy
old style",20,'bold'),bg='lightyellow').place(x=280,y=280,width=320)

txt_full_marks=Entry(self.root,textvariable=self.var_full_marks,font=("goudy
old style",20,'bold'),bg='lightyellow').place(x=280,y=340,width=320)
        #=========Button=================
        btn_add=Button(self.root,text="submit",font=("times new
roman",15),bg="lightgreen",activebackground="lightgreen",cursor="hand2",comma
nd=self.add).place(x=300,y=420,width=120,height=35)
        btn_clear=Button(self.root,text="Clear",font=("times new
roman",15),bg="lightgrey",activebackground="lightgreen",cursor="hand2",comman
d=self.clear).place(x=430,y=420,width=120,height=35)
        #==========================================================
    def fetch_roll(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            cur.execute("select roll from student")
            rows=cur.fetchall()
            if len(rows)>0:
                for row in rows:
                    self.roll_list.append(row[0])
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")

    def search(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            cur.execute("select name,course from student where
roll=?",(self.var_roll.get(),))
            row=cur.fetchone()
            if row!=None:
                self.var_name.set(row[0])
                self.var_course.set(row[1])
            else:
                messagebox.showerror("Error","No record
found",parent=self.root)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")

    def add(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_name.get()=="":
                messagebox.showerror("Error","Please first search student
record",parent=self.root)
            else:
                cur.execute("select * from result where roll=? and
course=?",(self.var_roll.get(),self.var_course.get(),))
                row=cur.fetchone()
                if row!=None:
                    messagebox.showerror("Error","Result Already
Present",parent=self.root)
                else:

per=(int(self.var_marks.get())*100)/int(self.var_full_marks.get())
                    cur.execute("insert into result
(roll,name,course,marks_ob,full_marks,per) values(?,?,?,?,?,?)",(
                        self.var_roll.get(),
                        self.var_name.get(),
                        self.var_course.get(),
                        self.var_marks.get(),
                        self.var_full_marks.get(),
                        str(per)
                    ))
                    con.commit()
                    messagebox.showinfo("Success","Result Added
Successfully",parent=self.root)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")

    def clear(self):
        self.var_roll.set("Select")
        self.var_name.set("")
        self.var_course.set("")
        self.var_marks.set("")
        self.var_full_marks.set("")


if __name__=="__main__":
    root=Tk()
    obj=ResultClass(root)
    root.mainloop()
```

5. Source code from file “report.py”

```
from tkinter import *
from tkinter import ttk,messagebox
import sqlite3

class ReportClass:
    def __init__(self,root):
        self.root=root
        self.root.title("Student Result Management System")
        self.root.geometry("1200x480+80+170")
        self.root.config(bg="white")
        self.root.focus_force()
        # ===title=====
        title=Label(self.root,text="View Student Results",font=("goudy old
style",18,"bold"),bg="blue",fg="white").place(x=10,y=15,width=1180,height=50)
        #=================search===============
        self.var_search=StringVar()
        self.var_id=""
        lbl_search=Label(self.root,text="Select By Roll No",font=("goudy old
style",15,'bold'),bg='white').place(x=280,y=100)
        txt_search=Entry(self.root,textvariable=self.var_search,font=("goudy
old style",15,'bold'),bg='lightgreen').place(x=500,y=100,width=150)
        btn_search=Button(self.root,text='Search',font=("goudy old
style",15,'bold'),bg="#03a9f4",fg='white',cursor="hand2",command=self.search)
.place(x=680,y=100,width=100,height=36)
        btn_clear=Button(self.root,text='Clear',font=("goudy old
style",15,'bold'),bg="orange",fg='white',cursor="hand2",command=self.clear).p
lace(x=800,y=100,width=100,height=36)

        lbl_roll=Label(self.root,text="Roll No.",font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE).place(x=150,y=230,width=150,
height=50)
        lbl_name=Label(self.root,text="Name",font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE).place(x=300,y=230,width=150,
height=50)
        lbl_course=Label(self.root,text="Course",font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE).place(x=450,y=230,width=150,
height=50)
        lbl_marks=Label(self.root,text="Marks obtained",font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE).place(x=600,y=230,width=150,
height=50)
        lbl_full_marks=Label(self.root,text="Full Marks",font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE).place(x=750,y=230,width=150,
height=50)
        lbl_per=Label(self.root,text="Percentage",font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE).place(x=900,y=230,width=150,
height=50)

        self.roll=Label(self.root,font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE)
        self.roll.place(x=150,y=280,width=150,height=50)
        self.name=Label(self.root,font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE)
        self.name.place(x=300,y=280,width=150,height=50)
        self.course=Label(self.root,font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE)
        self.course.place(x=450,y=280,width=150,height=50)
        self.marks=Label(self.root,font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE)
        self.marks.place(x=600,y=280,width=150,height=50)
        self.full=Label(self.root, font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE)
        self.full.place(x=750,y=280,width=150,height=50)
        self.per=Label(self.root, font=("goudy old
style",15,'bold'),bg='white',bd=2,relief=GROOVE)
        self.per.place(x=900,y=280,width=150,height=50)

        #========button delete===========
        btn_delete=Button(self.root,text='Delete',font=("goudy old
style",15,'bold'),bg="red",fg='white',cursor="hand2",command=self.delete).pla
ce(x=500,y=350,width=150,height=36)
#=============================================================
    def search(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_search.get()=="":
                messagebox.showerror("Error","Roll No. should be
required",parent=self.root)
            else:
                cur.execute("select * from result where
roll=?",(self.var_search.get(),))
                row=cur.fetchone()
                if row!=None:
                    self.var_id=row[0]
                    self.roll.config(text=row[1])
                    self.name.config(text=row[2])
                    self.course.config(text=row[3])
                    self.marks.config(text=row[4])
                    self.full.config(text=row[5])
                    self.per.config(text=row[6])
                else:
                   messagebox.showerror("Error","No record
found",parent=self.root)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to {str(ex)}")
    def clear(self):
        self.var_id=""
        self.roll.config(text="")
        self.name.config(text="")
        self.course.config(text="")
        self.marks.config(text="")
        self.full.config(text="")
        self.per.config(text="")
        self.var_search.set("")

    def delete(self):
        con=sqlite3.connect(database="rms.db")
        cur=con.cursor()
        try:
            if self.var_course.get() == "":
                messagebox.showerror("Error","search Student Result 
first",parent=self.root)
            else:
                cur.execute("select * from result where 
rid=?",(self.var_id,))
                row=cur.fetchone()
                if row==None:
                    messagebox.showerror("Error","Invalid Student 
Result",parent=self.root)
                else:
                    op=messagebox.askyesno("Confirm","Do you really want to 
delete?",parent=self.root)
                    if op==True:
                        cur.execute("delete from result where 
rid=?",(self.var_id,))
                        con.commit()
                        messagebox.showinfo("Delete","Result deleted 
successfully",parent=self.root)
                        self.clear()

        except Exception as ex:
            messagebox.showerror("Error", f"Error due to {str(ex)}")


if __name__=="__main__":
     root=Tk()
     obj=ReportClass(root)
     root.mainloop()
```

6. Source code from file “login.py”

```
from tkinter import*
from tkinter import messagebox
import sqlite3
import os

class Login_Window:
    def __init__(self,root):
        self.root=root
        self.root.title("Login System")
        self.root.geometry("1350x700+0+0")
        self.root.config(bg="lightgreen")
        #=====login frame=======
        login_frame=Frame(self.root,bd=2,relief=RIDGE)
        login_frame.place(x=650,y=90,width=350,height=460)

        title=Label(login_frame,text="Login System",font=("times new
roman",20,'bold'),bg="orange").place(x=0,y=30,relwidth=1)


        lbl_user=Label(login_frame,text="Username",font=("times new
roman",20,'bold'),bg="yellow",fg="blue").place(x=50,y=100)
        self.username = StringVar()
        self.password = StringVar()


txt_username=Entry(login_frame,textvariable=self.username,font=("times new
roman",20,'bold'),bg="yellow",fg="blue").place(x=50,y=140,width=250)
        lbl_pass=Label(login_frame,text="Password",font=("times new
roman",20,'bold'),bg="yellow",fg="blue").place(x=50,y=190)
        txt_pass=Entry(login_frame,textvariable=self.password,font=("times
new roman",20,'bold'),bg="yellow",fg="blue").place(x=50,y=240,width=250)

        btn_login=Button(login_frame,text="Log In",font=("times new
roman",18),bg="yellow",fg="blue",activebackground="yellow",cursor="hand2",com
mand=self.login).place(x=50,y=300,width=250,height=36)

hr=Label(login_frame,bg="lightgray").place(x=50,y=370,width=250,height=2)
        or_=Label(login_frame,text="OR",font=("times new
roman",15)).place(x=145,y=360)

        btn_forget=Button(login_frame,text="Forget Password",font=("times new
roman",13),fg="blue").place(x=100,y=390)

        #========Frame2=======
        register_frame=Frame(self.root,bd=2,relief=RIDGE,bg="white")
        register_frame.place(x=650,y=570,width=350, height=60)

        lbl_reg=Label(register_frame,text="Don't have an
account?",font=("times new roman",12),bg="yellow",fg="blue").place(x=40,y=20)
        btn_signup=Button(register_frame,text="Sign Up",font=("times new
roman",14,'bold'),fg="blue").place(x=210,y=13)

    def login(self):
        if self.username.get()=="" or self.password.get()=="":
            messagebox.showerror("Error","All fields are required")
        elif self.username.get()!="sfaf" or self.password.get()!="231241":
            messagebox.showerror("Error","Invalid username or password\nTry
again")
        else:
            messagebox.showinfo("Informations",f"Welcome :
{self.username.get()}\nYour Password : {self.password.get()}")
if __name__=="__main__":
     root=Tk()
     obj=Login_Window(root)
     root.mainloop()
```

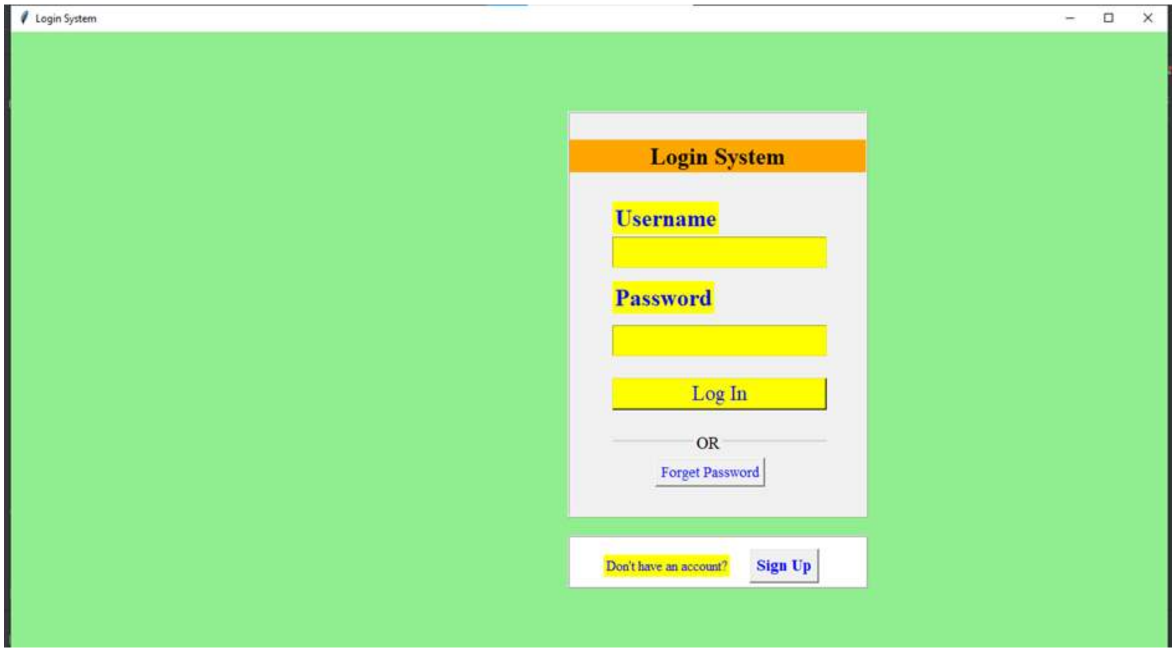
7. Source code from file “create_db.py”

```
import sqlite3
def create_db():
    con=sqlite3.connect(database="rms.db")
    cur=con.cursor()

    cur.execute("CREATE TABLE IF NOT EXISTS course(cid INTEGER PRIMARY KEY
AUTOINCREMENT,name text,duration text,charges text,description text)")
    con.commit()

    cur.execute("CREATE TABLE IF NOT EXISTS student(roll INTEGER PRIMARY KEY
AUTOINCREMENT,name text,email text,gender text,dob text,contact
text,admission text,course text,state text,city text,pin text,address,text)")
    con.commit()

    cur.execute("CREATE TABLE IF NOT EXISTS result(rid INTEGER PRIMARY KEY
AUTOINCREMENT,roll text,nametext,course text,marks_ob text,full_marks
text,per text)")
    con.commit()

    cur.execute("CREATE TABLE IF NOT EXISTS employee(eid INTEGER PRIMARY KEY
AUTOINCREMENT,f_name text,l_name,contact text,email text,question text,answer
text, password text)")
    con.commit()


    con.close()
create_db()
```

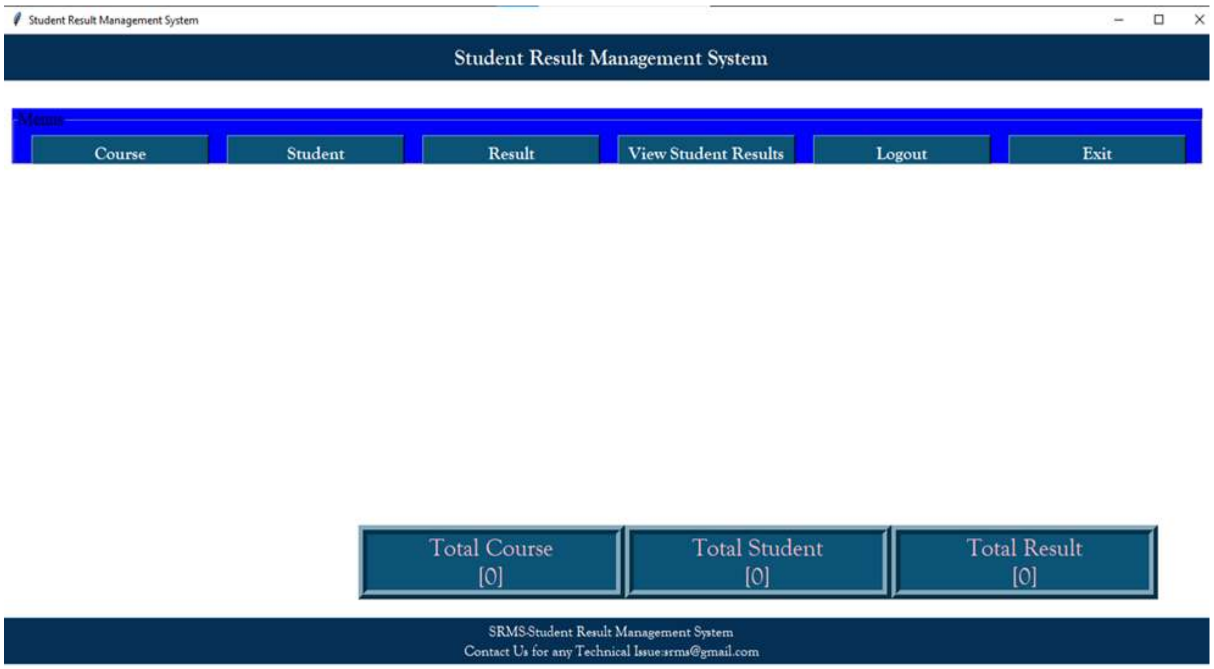
CHAPTER 4

4.1 SCREENSHOTS OF PROJECT OUTCOMES

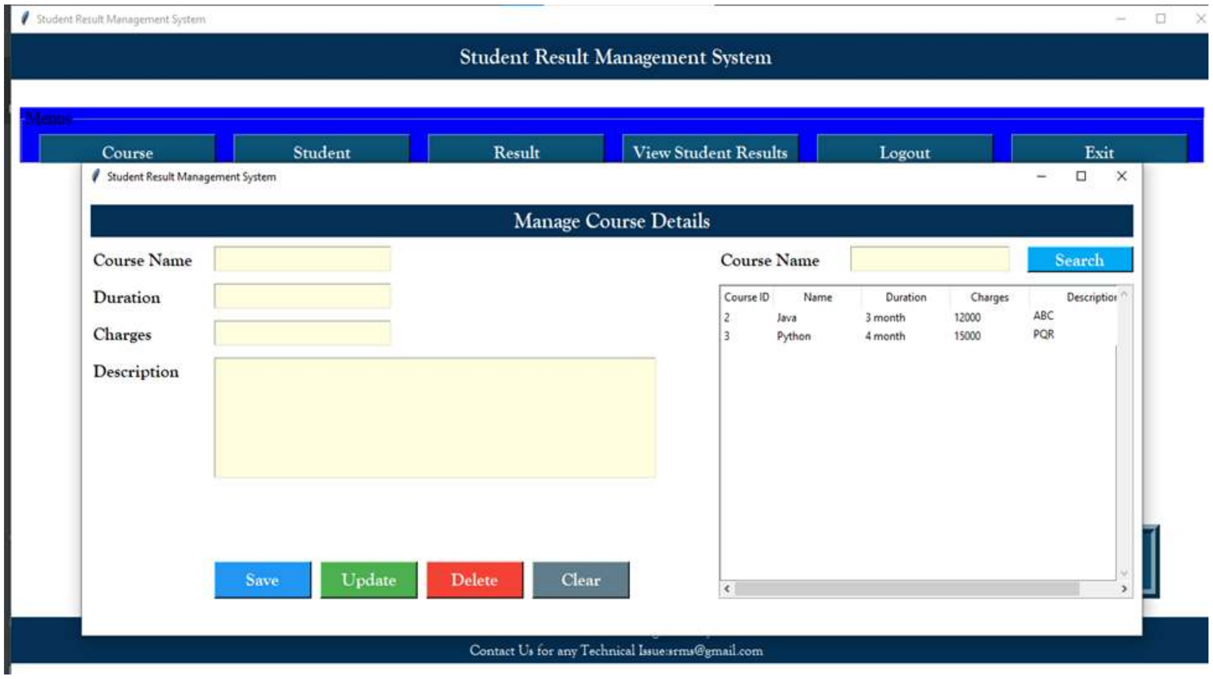
a) Login page of the application



b) Main page of the application


Student Result Management System
Student Result Management System
Menus
Course
Student
Result
View Student Results
Logout
Exit
Total Course
[0]
Total Student
[0]
Total Result
[0]
SRMS-Student Result Management System
Contact Us for any Technical Issue:srms@gmail.com

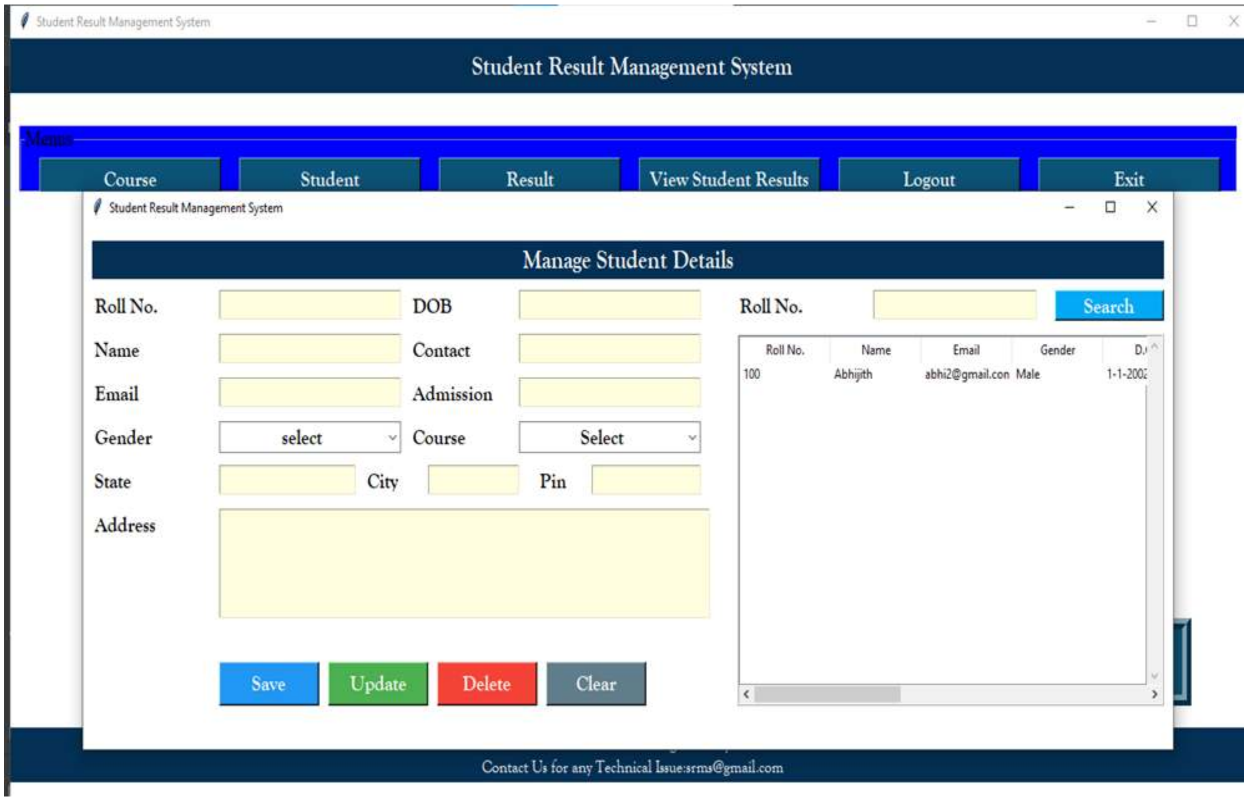
c) Menu 1: Course details page of the application



Course ID	Name	Duration	Charges	Descriptior
2	Java	3 month	12000	ABC
3	Python	4 month	15000	PQR

Contact Us for any Technical Issue:srms@gmail.com

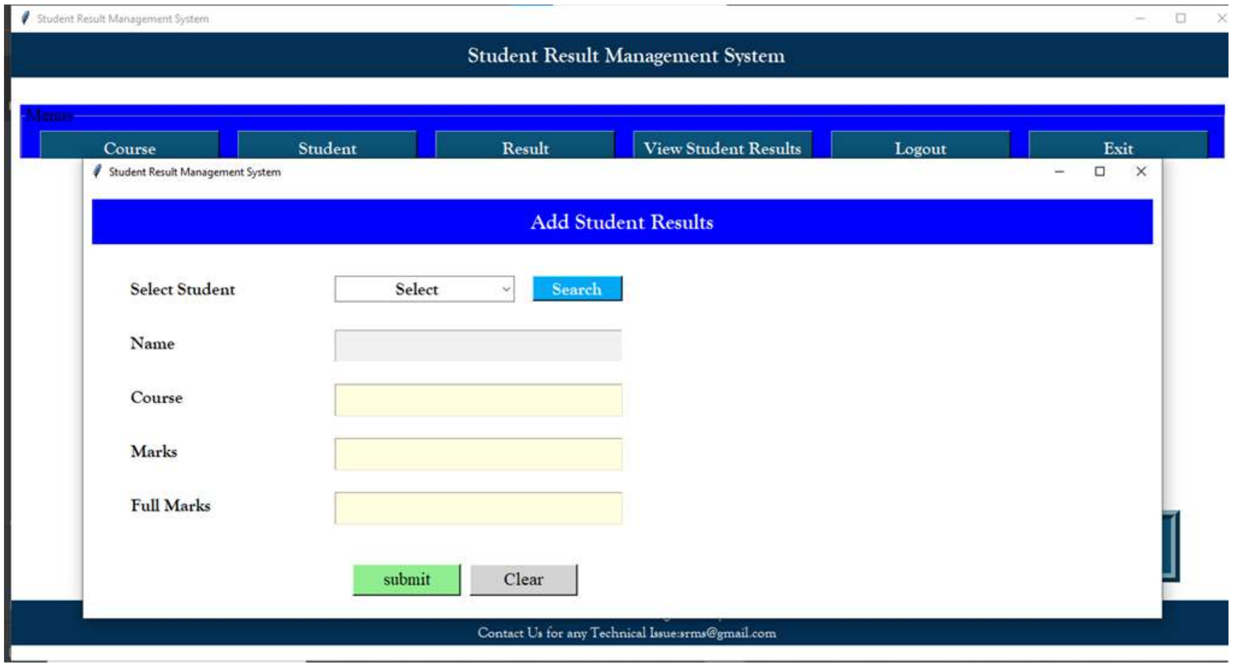
d) Menu 2: Student details page of the application



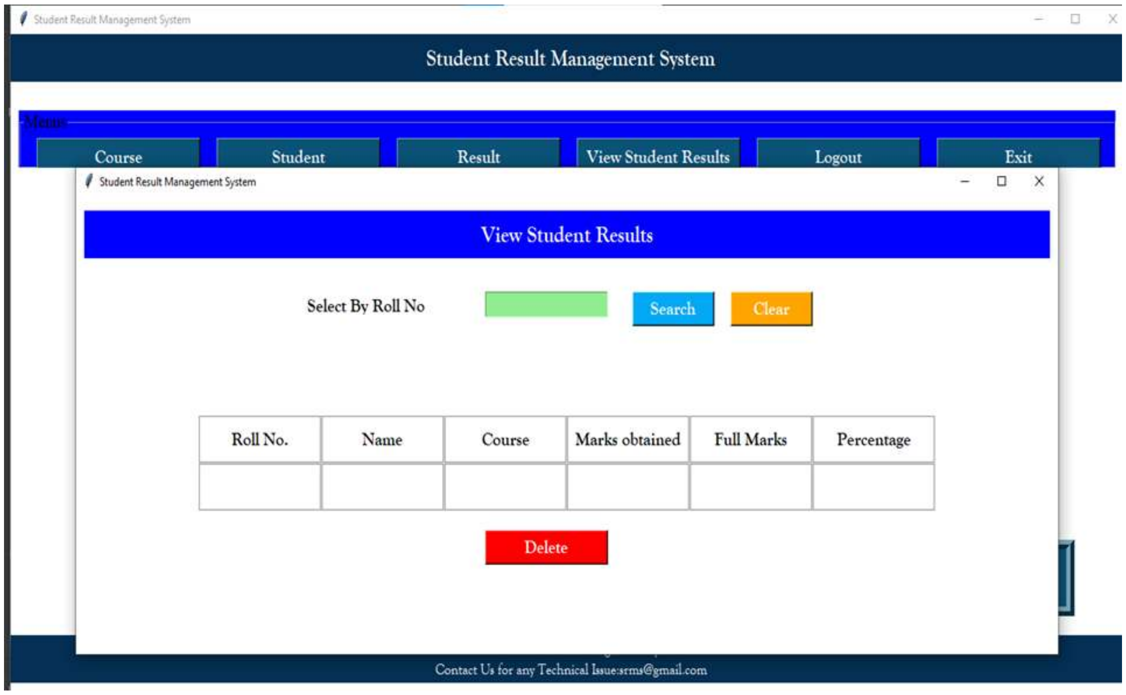
Roll No.	Name	Email	Gender	D.(
100	Abhijith	abhi2@gmail.con	Male	1-1-2002

Contact Us for any Technical Issue:srms@gmail.com

e) Menu 3: Result details page of the application

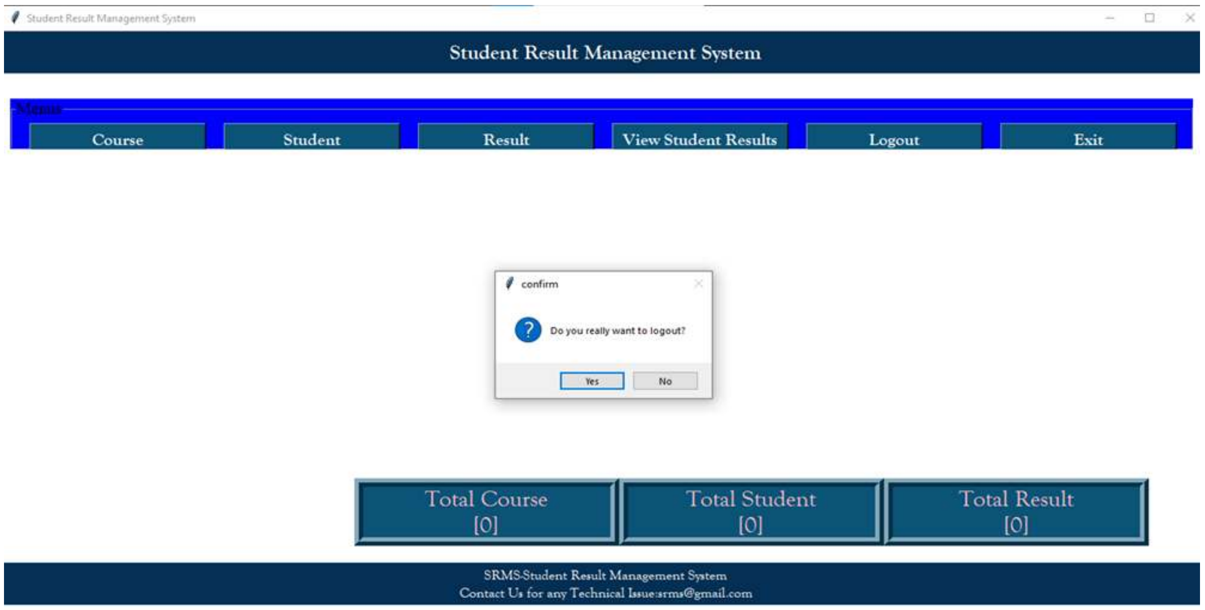

Student Result Management System
Student Result Management System
Menus
Course
Student
Result
View Student Results
Logout
Exit
Student Result Management System
Add Student Results
Select Student
Select
Search
Name
Course
Marks
Full Marks
submit
Clear
Contact Us for any Technical Issue:srms@gmail.com

f) Menu 4: Report details page of the application


Student Result Management System
Student Result Management System
Menus
Course
Student
Result
View Student Results
Logout
Exit
Student Result Management System
View Student Results
Select By Roll No
Search
Clear
Delete
Contact Us for any Technical Issue:srms@gmail.com

Roll No.	Name	Course	Marks obtained	Full Marks	Percentage

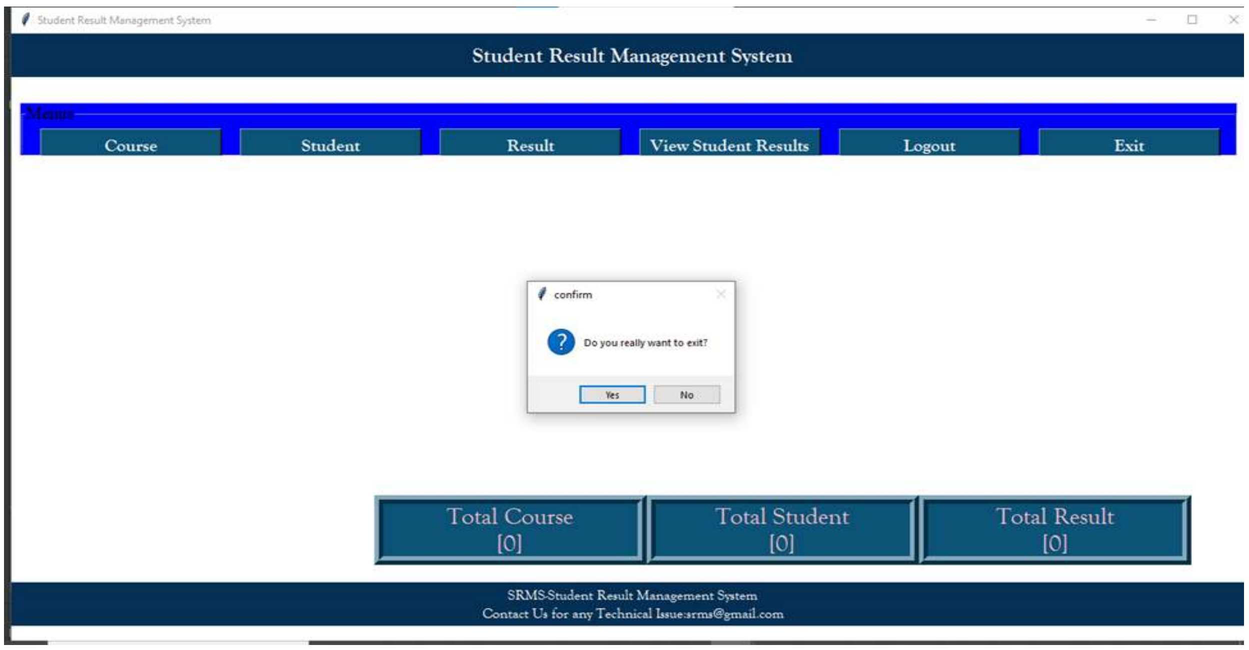
g) Logout



Do you really want to logout?

SRMS-Student Result Management System
Contact Us for any Technical Issue:srms@gmail.com

h) Exit



Do you really want to exit?

SRMS-Student Result Management System
Contact Us for any Technical Issue:srms@gmail.com

4.2 ANALYSIS AND FINDINGS

The Student Result Management System is designed to manage student data, including personal details, course details, grades, and attendance records. The system is designed to help administrators, teachers, and students easily access and manage this information.

One of the main benefits of the system is that it allows for easy retrieval of student data. This can help teachers and administrators quickly find and analyse student information, including grades and attendance records. The system also makes it easy to generate reports on student performance, which can be used to identify areas where students may need additional support. Another benefit of the system is that it can help improve communication between teachers, administrators, and students. Teachers can use the system to communicate with students about their performance, while administrators can use it to communicate with teachers and parents about student progress. One challenge of the system is ensuring the security and privacy of student data. This requires implementing strong security measures, such as user authentication and authorization, encryption of sensitive data, and secure connections. It also requires ensuring that only authorized users can access student data.

In near future, the system interface could be improved, with more attractive, interactive and meaningful images; enhance the system with an email and SMS or email notifications. Enhance the current system by computerizing almost all the services provided by the institution

Overall, the Student Result Management System can help improve the efficiency and effectiveness of organizing student data. It can also help improve student performance by providing teachers and administrators with the information they need to identify areas where students may need additional support. However, ensuring the security and privacy of student data is essential for the system to be successful.

CONCLUSION

The Student Result Management System is a valuable tool for educational institutions to manage and analyse student performance. In this chapter, we will discuss the main conclusions of the project and potential future work. Through the development of the Student Result Management System, we have demonstrated the importance of system design and implementation in creating a reliable and efficient tool for organizing student performance data. The system design phase helped to ensure that the system met the requirements of its stakeholders, while the implementation phase ensured that the system was developed using appropriate programming languages and tools. The system's functionality was thoroughly tested to ensure that it worked as intended. While the current version of the Student Result Management System provides a valuable tool for organizing and analysing student performance data, there is always room for improvement. In conclusion, the development of the Student Result Management System has provided a valuable tool for educational institutions to manage and analyse student performance data. With further development and enhancements, the system has the potential to become an even more powerful tool for supporting student success.

REFERENCES

1. Dhruv G. Patel and Harshit P. Modi (07 |July 2022) : A Review on Student Result Management System, Graduated Student of Department of computer application, Shri Sarvajanik B.C.A. & P.G.D.C.A. College, Gujarat, India and Graduated Student of Department of Mechanical Engineering, U.V. Patel College of Engineering., Ganpat University, Gujarat, India, e-ISSN: 2395-0056, p-ISSN: 2395-0072, ISO 9001:2008 Certified Journal, Volume: 09 Issue, page no: 2963-2965, publisher: www.irjet.net

2. L Varun Ramesh, R Sai Anusha Priyanka, SNSS Venkata Lakshmi, V Mounika (July 2021): Student Result Management System, Department of Computer Science and Engineering Andhra Loyola Institute of Engineering and Technology Jawaharlal Nehru Technological University Kakinada, ISSN NO:0377-9254, Vol 12, Issue 7, page no 329-332, publisher: www.jespublication.com

Code Repo Link

https://github.com/AmeenaParayangatt/python-srms