

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018



**A Database Management System Mini Project Report
on**

“STUDENT RESULT MANAGEMENT SYSTEM”

Submitted in Partial fulfilment of the Requirements for the V Semester of the Degree of

**Bachelor of Engineering
In
Computer Science & Engineering
By**

**SOURITA PODDAR
(1CR18CS162)**

**SRADDHA CHALLAGUNDLA
(1CR18CS164)**

Under the Guidance of

**Mrs. Danthuluri Sudha
Assoc Professor, Dept. of CSE**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CMR INSTITUTE OF TECHNOLOGY**

**#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,
BANGALORE-560037**

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,

BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the Database Management System Project work entitled “**STUDENT RESULT MANAGEMENT SYSTEM**” has been carried out by **Sourita Poddar (1CR18CS162)** and **Sraddha Challagundla (1CR18CS164)** bona fide students of CMR Institute of Technology in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2020-2021**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. This DBMS Project Report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Signature of Guide

Mrs. Danthuluri Sudha
Assoc Professor
Dept. of CSE, CMRIT

Signature of HOD

Dr. Prem Kumar Ramesh
Professor, Head
Dept. of CSE, CMRIT

External Viva

Name of the examiners

Signature with date

1.

2.

ABSTRACT

The main aim of the project is to provide the internal assessment result to the student in a simple and accurate way.

The purpose of the Student Result Management System is to automate the existing manual system by the help of computer software fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same as well as to provide a common platform for students to be able to access their result/concerned data in an easy manner.

The back end development of this System will be based on Python as the programming language, front end development based on a Tkinter Graphic User Interface and MySQL server as the database of the System. Python language is advantageous due to its ease of use and validation properties while MySQL has better advanced features and properties, has good security, is open source and has cross platform operability and the Tkinter GUI used brings the design together with numerous available widgets.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible, success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So with gratitude I acknowledge all those whose guidance and encouragement served as beacon of light and crowned our effort with success.

I would like to thank **Dr. Sanjay Jain**, Principal, CMRIT, Bangalore for providing an excellent academic environment in the college and his never-ending support for the B.E program.

I would also like to thank **Dr. Prem Kumar R**, Professor & Head - Dept. of Computer Science & Engineering , CMRIT, Bangalore who shared her opinions and experiences through which I received the required information crucial for the project.

I consider it a privilege and honour to express my sincere gratitude to my internal guide and Co-guide **Mrs. Sudha D**, Associate Professor, Dept. of Computer Science & Engineering, CMRIT, Bangalore for their valuable guidance throughout the tenure of this project work.

I would also like to thank all the faculty members who have always been very Cooperative and generous. Conclusively, I also thank all the non-teaching staff and all others who have done immense help directly or indirectly during my project

SOURITA PODDAR (1CR18CS162)
SRADDHA CHALLAGUNDLA (1CR18CS164)

TABLE OF CONTENTS

Certificate	Page No.
Abstract	I
Acknowledgement	II
Table of contents	III
List of tables and figures	IV
1 INTRODUCTION	
1.1 Problem Statement	1
1.2 Relevance of the Project	1-2
1.3 Objective	2
2 SYSTEM REQUIREMENTS	
2.1 Software Requirements	4
2.2 Hardware Requirements	4
3 DESIGN	
3.1 E-R Diagram	5
3.2 Database SCHEMA	6
4 IMPLEMENTATION	7
5 SCREENSHOTS	24
6 FUTURE SCOPE & CONCLUSION	
6.1 Future scope	35
6.2 Conclusion	36
BIBLIOGRAPHY	37

LIST OF TABLES AND FIGURES

Table		Page No.
1	Database schema : department table	6
2	Database schema : course table	6
3	Database schema : student table	6
4	Database schema : attendance table	6
5	Database schema : marks table	6
6	Database schema : semester table	6
7	Database schema : studentcount table	6
Figure		
1	E-R Diagram	5

Chapter 1

INTRODUCTION

1.1 Problem Statement

Student Result Management System is a technological opportunity for institutions searching for a simple and alternative solution to the conventional paper-based exam results evaluation, reporting and distribution. Like any other software, the system comes with certain advantages and disadvantages.

The software application unravels and quickens the result management system with unique templates by providing the administration a secure database system for storing, evaluating and publishing the test scores and grades of candidates. The database likewise allows the students to observe and gander at the assessment results on the application at whatever point necessary.

The Student Result Management System when implemented manually will have the following shortcomings :

- Using the Student Result system manually will require maintaining a clear and concise report containing the records of the students. This will require more time, calculation and an unnecessary use of resources.
- The evaluation, calculation and reporting of the data becomes a difficult job.
- Keeping track of all student records becomes a tedious job.

Thus, such problems can be solved by developing an automated **STUDENT RESULT MANAGEMENT SYSTEM**.

1.2 Relevance of the project

Computer based information systems are designed to improve existing systems. They have user friendly interfaces having quick authenticated access to data. It provides the facility of maintaining the details of the student records.

A Student Result Management System is an excellent means to keep track of all student records and assessment data in an accurate and timely manner. The hassle of calculation, reporting, storage and collection of records is eliminated with the help of an automated system.

Among the main functionalities of the system:

Admin can add new student details, delete student details, update student attendance and student assessment results.

Students can access their assessment results once updated by the administrator.

This project has a large scope as it has the following features which help in making it easy to use, understand and modify it:

- Automation of Student Result Management
- No Need to do Paper Work
- To save the environment by paper free work
- To increase the accuracy and efficiency of evaluation
- Management of Student Data
- Management of Attendance Data
- Management of Assessment Result Data

1.3 Objective

The main objective of the project on Student Result Management System is to manage the details of Student data, Attendance data, Assessment results. In order to avoid existing student result management problems, we are planning to design a system so that the student result system becomes more automated, accurate and effective.

The objective of this project is to implement and execute a Student Result Management System through a desktop application using MySQL server as the back-end, Python programming language and Tkinter Graphic User Interface as the front-end and provide an automated system for users.

Few basic objectives of the Student Result Management System are:

- Admin is provided with a login feature to ensure security.
- Admin can connect to the database and use features like: Add student details, Delete student details, Update Attendance details and Update Assessment Result details.
- Students can simply connect to the database and view the assessment results for the concerned subjects.

Few main objectives of the project are :

1. Reduce paperwork and create a database for a student result system.
2. Save time and workload.
3. Easy to access.
4. Avoid duplicate Entry.
5. Admin is provided with a login, who has access to update or delete details.
6. Assessment result details are provided to the student user.

Student Result Management System

7. Improve accuracy in result.
8. User friendly interface.

Chapter 2

SYSTEM REQUIREMENTS

2.1 Software requirements:

1. Operating System : Windows
2. Front End : Python 3.8, Tkinter
3. Back End : My SQL
4. File to executable converter : cx_Freeze module
5. Software used for programming:
 - Windows 10
 - Visual Studio Code/Python IDLE(3.8)
 - Xampp server

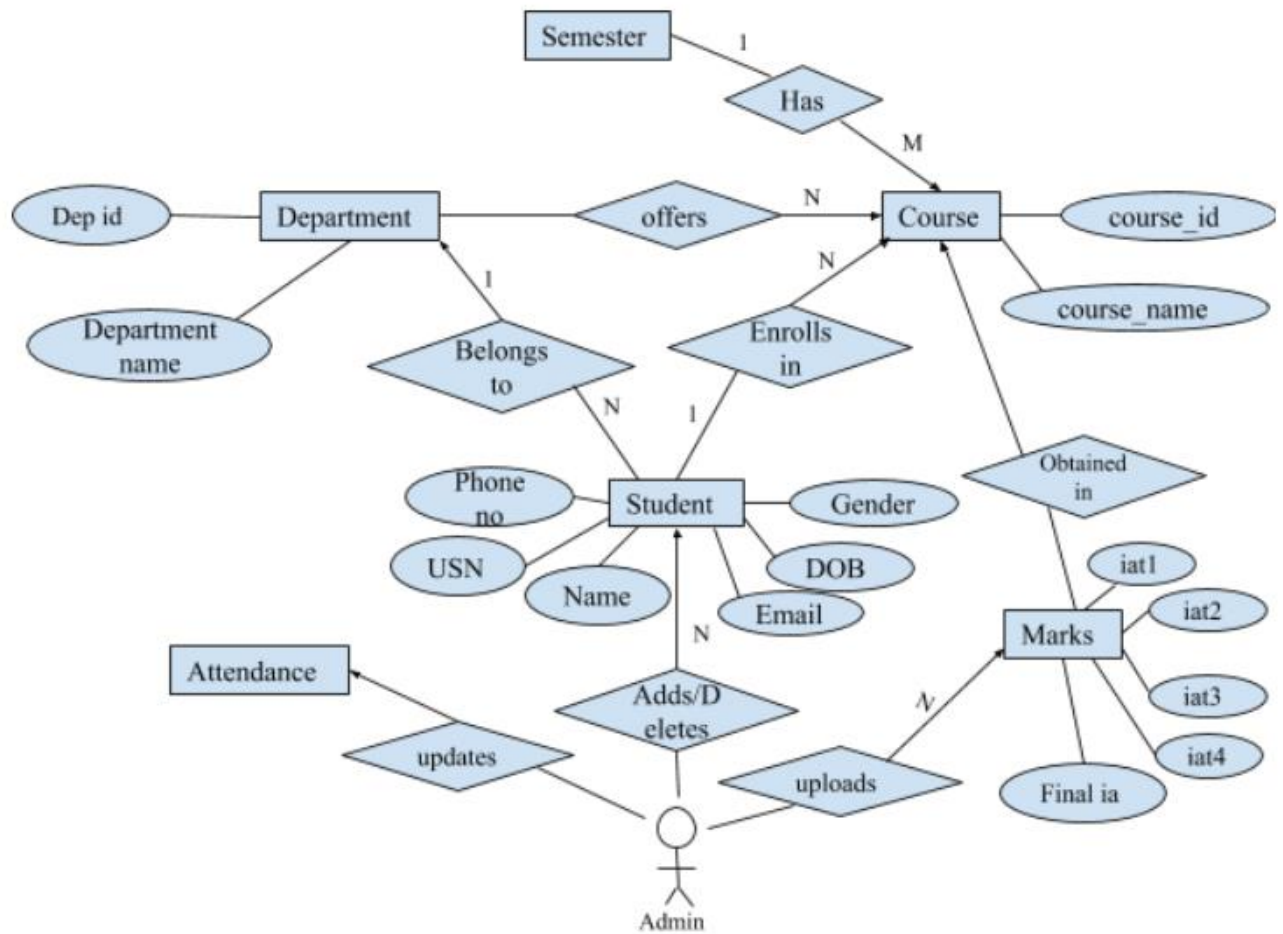
2.2 Hardware requirements:

1. Central Processing Unit : Intel(R) Core(TM) i5-8250U @1.60Ghz 1.80Ghz
2. Clock Speed : 2.5 GHz
3. Main Memory : 8 GB Ram
4. Cache Memory : 1024 KB
5. Hard Disk Capacity : 1024 GB
6. USB : 3.0
7. Keyboard : Standard keyboard
8. Mouse : Optical/Laser Mouse

Chapter 3

DESIGN

3.1 E-R DIAGRAM



3.2 Database Schema

Database Name: studentresultsystem

department

<u>department_id</u>	department_name
----------------------	-----------------

course

<u>course_id</u>	course_name	credits
------------------	-------------	---------

student

<u>usn</u>	name	email	dob	gender	department_id	phonenummer
------------	------	-------	-----	--------	---------------	-------------

attendance

usn	course_id	classes_taken	classes_attended
-----	-----------	---------------	------------------

marks

usn	course_id	iat1	iat2	iat3	iat4	finalia
-----	-----------	------	------	------	------	---------

semester

semester	course_id
----------	-----------

Trigger schema:

studentcount

students

Chapter 4

IMPLEMENTATION

4.1 Connecting to the database :

```
def submitdb():  
    global con, mycursor  
    host = hostval.get()  
    user = userval.get()  
    password = passwordval.get()  
    try:  
        con = mysql.connector.connect(host=host,user=user,password=password)  
        mycursor = con.cursor()  
        dbroot.destroy()  
        messagebox.showinfo('Notification','Successfully connected to database')  
    except:  
        messagebox.showerror('Notification','Data is incorrect! Please try again')  
    return
```

4.2 Admin login function :

```
def checklogin():  
    id = adminval.get()  
    pwd = passwordval.get()  
    try:  
        strr='use studentresultssystem'  
        mycursor.execute(strr)  
        if (id=='admin' and pwd=='123'):  
            messagebox.showinfo('Notification','Successfully logged in')  
            adminroot.destroy()  
            adminmenu()
```

```
else:
    messagebox.showerror('Notification','Incorrect User ID/Password! Please try
again')
except:
    messagebox.showerror('Notification','Please connect to database')
```

4.3 Adding student to the database:

```
def submitadd():
    name= nameval.get()
    usn= usnval.get()
    dob= dobval.get()
    gender= genderval.get()
    email= emailval.get()
    deptid= deptidval.get()
    phoneno= phonenoval.get()
    try:
        usnformat= re.compile(r"^[0-9a-zA-Z]{3}[0-9]{2}[a-zA-Z]{2}[0-9]{3}$")
        usnmatch= usnformat.search(usn)
        dateformat= re.compile(r"^\d{4}-(0?[1-9]|1[012])\-(0?[1-9]|[12][0-9]|3[01])$")
        datematch= dateformat.search(dob)
        emailformat= re.compile(r"^[a-zA-Z0-9_\.]+\@[1][a-zA-Z0-9]+\.$")
        emailmatch= emailformat.search(email)
        genderformat=re.compile(r"^(?:m|M|male|Male|O|o|Other|other|f|F|female|Female)$")
        gendermatch= genderformat.search(gender)
        deptformat= re.compile(r"[1-9]")
        deptmatch= deptformat.search(deptid)
        if ((name=="") or not(name.isalpha)):
            messagebox.showerror('Notification','Please enter valid Name')
        elif (usnmatch == None):
            messagebox.showerror('Notification','Please enter valid USN')
        elif (datematch == None):
            messagebox.showerror('Notification','Please enter valid DOB')
```

```
elif (emailmatch == None):
    messagebox.showerror('Notification','Please enter valid Email ID')
elif (gendermatch == None):
    messagebox.showerror('Notification','Please enter valid gender')
elif (deptmatch == None):
    messagebox.showerror('Notification','Please enter valid Department ID')
elif (len(phoneno)!=10 or phoneno[0] not in '9876'):
    messagebox.showerror('Notification','Please enter a valid 10 digit phone
number')
else:
    trigger1()
    deptid=int(deptid)
    usn=usn.upper()
    name=name.upper()
    email=email.lower()
    insert1 = 'insert into student values(%s,%s,%s,%s,%s,%s,%s)'
    mycursor.execute(insert1,(usn,name,email,dob,gender,deptid,phoneno))
    con.commit()

    res= messagebox.askyesnocancel("Notification","USN { } added
successfully...Clear form?".format(usn),parent=addroot)
    if res == True:
        nameval.set("")
        usnval.set("")
        emailval.set("")
        dobval.set("")
        genderval.set("")
        deptidval.set("")
        phonenoval.set("")
    strr ='select * from student'
    mycursor.execute(strr)
    data = mycursor.fetchall()
```

```
studenttable.delete(*studenttable.get_children())

for i in data:

    list1 = [i[0],i[1]]

    studenttable.insert("",END,values=list1)

count='select * from studentcount'

mycursor.execute(count)

data = mycursor.fetchall()

for i in data:

    list2=['TOTAL:',i[0]]

    studenttable.insert("",END,values=list2)

except:

    messagebox.showerror('Notification','An error occurred...Please try
again',parent=addroot)
```

4.4 Deleting a student from the database :

```
def delete():

    name = nameval.get()

    usn= usnval.get()

    try:

        usnformat= re.compile(r"^[0-9a-zA-Z]{3}[0-9]{2}[a-zA-Z]{2}[0-9]{3}$")

        usnmatch= usnformat.search(usn)

        if (usnmatch == None):

            messagebox.showerror('Notification','Please enter valid USN')

        else:

            trigger2()

            usn=usn.upper()

            strr = 'select count(usn) from student where usn=%s'

            mycursor.execute(strr,(usn,))

            for i in mycursor.fetchall():

                data=i[0]

                if (data != 0):

                    strr = 'delete from student where usn=%s'
```



```
mycursor.execute(strr,(usn,))

con.commit()

res = messagebox.askyesnocancel('Notification','USN { } Deleted
successfully..Clear form?').format(usn),parent=delroot)

if(res==True):

    nameval.set("")

    usnval.set("")

    studenttable.delete(*studenttable.get_children())

    strr='select * from student'

    mycursor.execute(strr)

    data = mycursor.fetchall()

    for i in data:

        list1 = [i[0],i[1]]

        studenttable.insert("",END,values=list1)

    count='select * from studentcount'

    mycursor.execute(count)

    data = mycursor.fetchall()

    for i in data:

        list2=["TOTAL:",i[0]]

        studenttable.insert("",END,values=list2)

    else:

        messagebox.showerror('Notification','USN { } does not
exist..'.format(usn),parent=delroot)

    except:

        messagebox.showerror('Notification','An error occurred...Please try
again',parent=delroot)
```

4.5 Updating student attendance in a particular course :

```
def attend():

    usn=usnval.get()

    course_id=courseval.get()

    taken=takenval.get()
```

```
attended=attendedval.get()

if taken=="":
    taken = 0

if attended=="":
    attended=0

try:
    usnformat= re.compile(r"^[0-9a-zA-Z]{3}[0-9]{2}[a-zA-Z]{2}[0-9]{3}$")
    usnmatch= usnformat.search(usn)
    courseformat=re.compile(r"^[1][0-9][a-zA-Z]+[1-8][0-9]$")
    coursematch= courseformat.search(course_id)
    if (usnmatch == None):
        messagebox.showerror('Notification','Please enter valid USN')
    elif (coursematch == None):
        messagebox.showerror('Notification','Please enter valid Course ID')
    else:
        usn=usn.upper()
        course_id=course_id.upper()
        strr = 'select count(usn) from student where usn=%s'
        mycursor.execute(strr,(usn,))
        for i in mycursor.fetchall():
            students=i[0]
        if students!=0:
            strr = 'select count(course_id) from course where course_id=%s'
            mycursor.execute(strr,(course_id,))
            for i in mycursor.fetchall():
                courses=i[0]
            if courses != 0:
                strr='delete from attendance where usn=%s and course_id=%s'
                mycursor.execute(strr,(usn,course_id))
                strr='insert into attendance values(%s,%s,%s,%s)'
                mycursor.execute(strr,(usn,course_id,taken,attended))
```

```
con.commit()

res = messagebox.askyesnocancel('Notification','Attendance for USN { }
Added successfully..Clear form?').format(usn),parent=attroot)

if(res==True):

    usnval.set("")
    courseval.set("")
    takenval.set("")
    attendedval.set("")

    attended=int(attended)
    taken=int(taken)

    attendance=(attended/taken)*100

    studenttable.delete(*studenttable.get_children())

    name='select s.name,c.course_name from student s, course c where
usn=%s and c.course_id=%s'

    mycursor.execute(name,(usn,course_id))

    for i in mycursor.fetchall():

        data=[usn,i[0],course_id,i[1]]

        studenttable.insert("",END,values=data)

        list2=['ATTENDANCE:',attendance]

        studenttable.insert("",END,values=list2)

    else:

        messagebox.showerror('Notification',"Course { } does not
exist..".format(course_id),parent=attroot)

    else:

        messagebox.showerror('Notification',"USN { } does not
exist..".format(usn),parent=attroot)

    except:

        messagebox.showerror('Notification',"An error occurred...Please try
again",parent=attroot)
```

4.6 Updating student marks in a particular course :

```
def marks():

    usn=usnval.get()
```

```

course_id=courseval.get()
iat1=iat1val.get()
iat2=iat2val.get()
iat3=iat3val.get()
iat4=iat4val.get()
if iat1=="":
    iat1=0
if iat2=="":
    iat2=0
if iat3=="":
    iat3=0
if iat4=="":
    iat4=0
iat1,iat2,iat3,iat4=int(iat1),int(iat2),int(iat3),int(iat4)
iat1=(iat1*4)/5
iat2=(iat2*4)/5
iat3=(iat3*4)/5
iat4=(iat4*4)/5
finalia=0
try:
    usnformat= re.compile(r"^[0-9a-zA-Z]{3}[0-9]{2}[a-zA-Z]{2}[0-9]{3}$")
    usnmatch= usnformat.search(usn)
    courseformat=re.compile(r"^[1][0-9][a-zA-Z]+[1-8][0-9]$")
    coursematch= courseformat.search(course_id)
    if (usnmatch == None):
        messagebox.showerror('Notification','Please enter valid USN')
    elif (coursematch == None):
        messagebox.showerror('Notification','Please enter valid Course ID')
    else:
        usn=usn.upper()
        course_id=course_id.upper()

```

```
str='select exists(select * from marks where usn=%s and course_id=%s)'
mycursor.execute(str,(usn,course_id))
for i in mycursor.fetchall():
    data=i[0]
if data==0:
    str = 'select count(usn) from student where usn=%s'
    mycursor.execute(str,(usn,))
    for i in mycursor.fetchall():
        students=i[0]
    if students!=0:
        str = 'select count(course_id) from course where course_id=%s'
        mycursor.execute(str,(course_id,))
        for i in mycursor.fetchall():
            courses=i[0]
        if courses != 0:
            str='insert into marks values(%s,%s,%s,%s,%s,%s,%s,%s)'
            mycursor.execute(str,(usn,course_id,iat1,iat2,iat3,iat4,finalia))
            str='update marks set finalia = (((iat1+iat2+iat3+iat4)-
least(iat1,iat2,iat3,iat4))/3)'
            mycursor.execute(str)
            con.commit()
            res = messagebox.askyesnocancel('Notification','Marks for USN {}
Added successfully..Clear form?').format(usn),parent=markroot)
            if(res==True):
                usnval.set("")
                courseval.set("")
                iat1val.set("")
                iat2val.set("")
                iat3val.set("")
                iat4val.set("")
            str='select
m.usn,s.name,m.course_id,c.course_name,m.iat1,m.iat2,m.iat3,m.iat4,m.finalia from
```

marks m,course c,student s where m.usn=%s and s.usn=%s and m.course_id=%s and m.course_id=c.course_id'

```
mycursor.execute(strr,(usn,usn,course_id))

studenttable.delete(*studenttable.get_children())

for i in mycursor.fetchall():

    data=[i[0],i[1],i[2],i[3],i[4],i[5],i[6],i[7],i[8]]

    studenttable.insert(",END,values=data)

else:

    messagebox.showerror('Notification',"Course { } does not
exit".format(course_id),parent=markroot)

else:

    messagebox.showerror('Notification',"USN { } does not
exist".format(usn),parent=markroot)

elif data==1:

    strr='select iat1,iat2,iat3,iat4 from marks where usn=%s and course_id=%s'

    mycursor.execute(strr,(usn,course_id))

    for i in mycursor.fetchall():

        ia1,ia2,ia3,ia4=i[0],i[1],i[2],i[3]

        if(iat1==0 and iat2==0 and iat3==0 and iat4==0):

            iat1,iat2,iat3,iat4=ia1,ia2,ia3,ia4

        elif(iat1==0 and iat2==0 and iat3==0):

            iat1,iat2,iat3=ia1,ia2,ia3

        elif(iat1==0 and iat2==0 and iat4==0):

            iat1,iat2,iat4=ia1,ia2,ia4

        elif(iat1==0 and iat3==0 and iat4==0):

            iat1,iat3,iat4=ia1,ia3,ia4

        elif(iat2==0 and iat3==0 and iat4==0):

            iat2,iat3,iat4=ia2,ia3,ia4

        elif(iat1==0 and iat2==0):

            iat1,iat2=ia1,ia2

        elif(iat1==0 and iat3==0):

            iat1,iat3=ia1,ia3
```

```
elif(iat1==0 and iat4==0):
    iat1,iat4=ia1,ia4
elif(iat2==0 and iat3==0):
    iat2,iat3=ia2,ia3
elif(iat2==0 and iat4==0):
    iat2,iat4=ia2,ia4
elif(iat3==0 and iat4==0):
    iat3,iat4=ia3,ia4
elif(iat1==0):
    iat1=ia1
elif(iat2==0):
    iat2=ia2
elif(iat3==0 ):
    iat3=ia3
elif(iat4==0):
    iat4=ia4

strr='update marks set iat1=%s,iat2=%s,iat3=%s,iat4=%s,finalia=%s where
usn=%s and course_id=%s'

mycursor.execute(strr,(iat1,iat2,iat3,iat4,finalia,usn,course_id))

strr='update marks set finalia =(((iat1+iat2+iat3+iat4)-
least(iat1,iat2,iat3,iat4))/3)'

mycursor.execute(strr)

con.commit()

res = messagebox.askyesnocancel('Notification','Marks for USN { } Added
successfully..Clear form?').format(usn),parent=markroot)

if(res==True):
    usnval.set("")
    courseval.set("")
    iat1val.set("")
    iat2val.set("")
    iat3val.set("")
    iat4val.set("")
```

```

        strr='select
m.usn,s.name,m.course_id,c.course_name,m.iat1,m.iat2,m.iat3,m.iat4,m.finalia from
marks m,course c,student s where m.usn=%s and s.usn=%s and m.course_id=%s and
m.course_id=c.course_id'

        mycursor.execute(strr,(usn,usn,course_id))

        studenttable.delete(*studenttable.get_children())

        for i in mycursor.fetchall():

            data=[i[0],i[1],i[2],i[3],i[4],i[5],i[6],i[7],i[8]]

            studenttable.insert(",END",values=data)

    except:

        messagebox.showerror('Notification',"An error occurred...Please try
again",parent=markroot)

```

4.7 Checking student result :

```

def showdata():

    usn=usnval.get()

    sem=semval.get()

    try:

        strr='use studentresultsystem'

        mycursor.execute(strr)

        usnformat= re.compile(r"^[0-9a-zA-Z]{3}[0-9]{2}[a-zA-Z]{2}[0-9]{3}$")

        usnmatch= usnformat.search(usn)

        semformat=re.compile(r"[1-8]{1}")

        semmatch = semformat.search(sem)

        if (usnmatch == None):

            messagebox.showerror('Notification','Please enter valid USN')

        elif (semmatch == None):

            messagebox.showerror('Notification','Please enter valid semester')

        else:

            result = 'select

m.usn,s.name,c.course_id,c.course_name,m.iat1,m.iat2,m.iat3,m.iat4,m.finalia from
marks m, student s, course c,semester se\

            where se.semester=%s and m.usn=%s and s.usn=%s and
se.course_id=c.course_id and se.course_id=m.course_id order by course_id'

```



```
mycursor.execute(result,(sem,usn,usn))

data = mycursor.fetchall()

if data==[]:

    messagebox.showerror('Notification','Data unavailabale...')

else:

    studenttable.delete(*studenttable.get_children())

    for i in data:

        list1 = [i[0],i[1],i[2],i[3],i[4],i[5],i[6],i[7],i[8]]

        studenttable.insert("",END,values=list1)

    except:

        messagebox.showerror('Notification','Please connect to
        database..",parent=studentroot)
```

4.8 Trigger function to be implemented when a new student is added to the database:

```
def trigger1():

    try:

        trigger ='CREATE TRIGGER COUNTSTUDENTS AFTER INSERT ON
        STUDENT FOR EACH ROW UPDATE studentcount SET studentcount.Students =
        (studentcount.Students+1)'

        mycursor.execute(trigger)

    except:

        pass
```

4.9 Trigger function to be implemented when a student is deleted from the database:

```
def trigger2():

    try:

        trigger ='CREATE TRIGGER STUDENTCOUNT AFTER DELETE ON
        STUDENT FOR EACH ROW UPDATE studentcount SET studentcount.Students =
        (studentcount.Students-1)'

        mycursor.execute(trigger)

    except:
```

pass

4.10 Root window configurations(front end):

```
root = Tk()
```

```
root.title("STUDENT RESULT MANAGEMENT SYSTEM")
```

```
root.config(background='gray28')
```

```
root.geometry('1174x650+75+60')
```

```
root.iconbitmap('icon.ico')
```

```
root.resizable(False,False)
```

```
#Frames
```

```
DataEntryFrame = Frame(root,bg = 'gray75',relief = GROOVE,borderwidth = 5)
```

```
DataEntryFrame.place(x=10,y=80,width=500,height=400)
```

```
#-----dataentry intro frame
```

```
frontlabel = Label(DataEntryFrame,text='-----Welcome-----',width=30,font=('arial',20,'italic bold'),bg='lightsteelblue2')
```

```
frontlabel.pack(side=TOP,expand=True)
```

```
adminbtn=Button(DataEntryFrame,text='Admin Login',width=25,font=('helvetica',20,'bold'),bd=6,bg='alice blue',activebackground='slategray4',relief=RIDGE,activeforeground='white',command=adminlogin)
```

```
adminbtn.pack(side=TOP,expand=True)
```

```
studentbtn=Button(DataEntryFrame,text='Student Result',width=25,font=('helvetica',20,'bold'),bd=6,bg='alice blue',activebackground='slategray4',relief=RIDGE,activeforeground='white',command=studentlogin)
```

```
studentbtn.pack(side=TOP,expand=True)
```

```
exitbtn=Button(DataEntryFrame,text='Exit',width=25,font=('helvetica',20,'bold'),bd=6,bg='alice blue',activebackground='slategray4',relief=RIDGE,activeforeground='white',command=exitwindow)
```

```
exitbtn.pack(side=TOP,expand=True)
```

```
#-----
```

```
ShowDataFrame = Frame(root,bg = 'gray90',relief = GROOVE,borderwidth = 5)
```

```
ShowDataFrame.place(x=550,y=80,width=620,height=500)
```

```

style = ttk.Style()

style.configure('Treeview.Heading',font=('helvetica',15,'bold'),foreground='sky blue')

style.configure('Treeview',font=('times',10,'bold'),foreground='black')

scroll_x= Scrollbar(ShowDataFrame,orient=HORIZONTAL)

scroll_y= Scrollbar(ShowDataFrame,orient=VERTICAL)

studenttable =
Treeview(ShowDataFrame,columns=('USN','NAME','COURSE_ID','COURSE_NAME','
IAT-1','IAT-2','IAT-3','IAT-
4','FINAL'),yscrollcommand=scroll_y.set,xscrollcommand=scroll_x.set)

scroll_x.pack(side=BOTTOM,fill=X)

scroll_y.pack(side=RIGHT,fill=Y)

scroll_x.config(command=studenttable.xview)

scroll_y.config(command=studenttable.yview)

studenttable.heading('USN',text='USN')

studenttable.heading('NAME',text='NAME')

studenttable.heading('COURSE_ID',text='COURSE')

studenttable.heading('COURSE_NAME',text='COURSE NAME')

studenttable.heading('IAT-1',text='IAT-1')

studenttable.heading('IAT-2',text='IAT-2')

studenttable.heading('IAT-3',text='IAT-3')

studenttable.heading('IAT-4',text='IAT-4')

studenttable.heading('FINAL',text='FINAL IAT')

studenttable['show']='headings'

studenttable.column('USN',width=150)

studenttable.column('NAME',width=200)

studenttable.column('COURSE_ID',width=110)

studenttable.column('COURSE_NAME',width=275)

studenttable.column('IAT-1',width=110)

studenttable.column('IAT-2',width=110)

studenttable.column('IAT-3',width=110)

studenttable.column('IAT-4',width=110)

studenttable.column('FINAL',width=120)

```

```

studenttable.pack(fill=BOTH,expand=1)

heading = "WELCOME TO STUDENT RESULT SYSTEM"

count = 0

text= "

Sliderlabel = Label(root,text=heading,font=('helvetica',20,'italic
bold'),relief=RIDGE,borderwidth=4,width=35,bg='light grey')

Sliderlabel.place(x=260,y=0)

IntroLabelTick()

clock =
Label(root,font=('times',14,'bold'),relief=RIDGE,borderwidth=4,bg='lightskyblue2')

clock.place(x=0,y=0)

tick()

connectbutton = Button(root,text='Connect to
Database',width=23,font=('helvetica',12,'italic
bold'),borderwidth=4,bg='palegreen2',activebackground='blue2',activeforeground='white',
command=connectdb)

connectbutton.place(x=930,y=0)

root.mainloop()

```

4.11 Stored procedure:

```

def storedprocedure():

    try:

        strr = 'create procedure display() select * from student'

        mycursor.execute(strr)

    except:

        pass

```

4.12 Display current date and time:

```

def tick():

    time_string=time.strftime("%H:%M:%S")

    date_string=time.strftime("%d-%m-%Y")

    clock.config(text='Date : '+date_string+'\nTime : '+time_string)

    clock.after(125,tick)

```

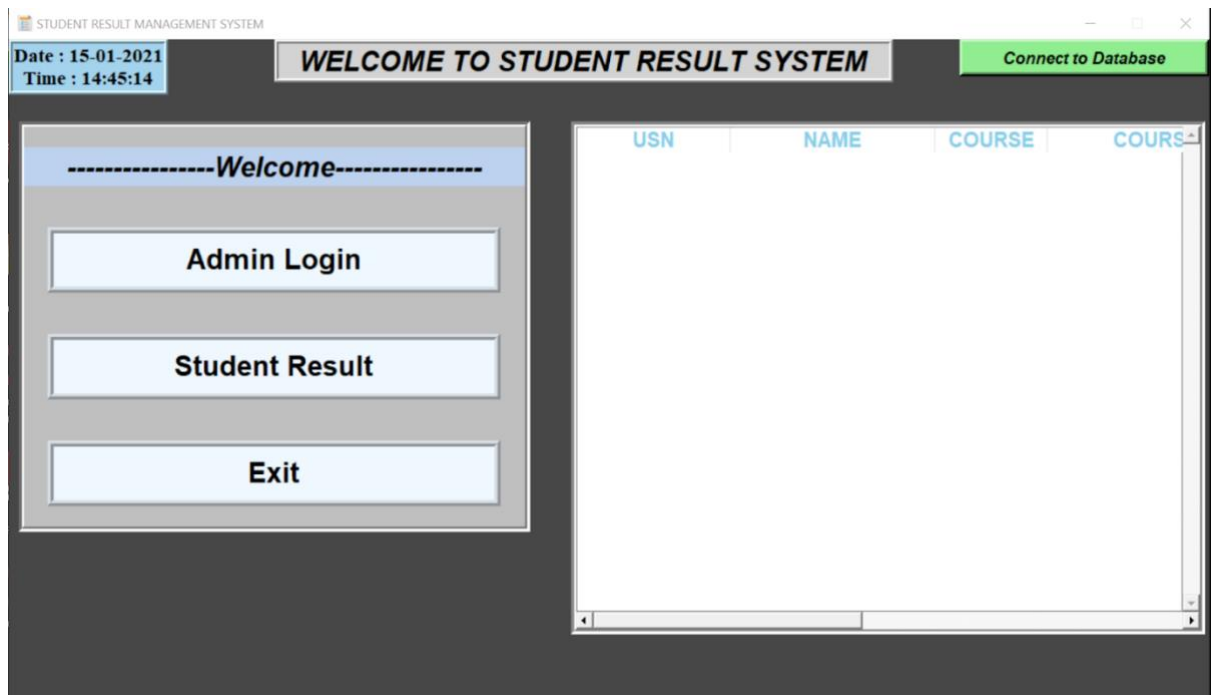
4.13 Slider label title display:

```
def IntroLabelTick():  
    global count,text  
    if (count>=len(heading)):  
        Sliderlabel.config(text=heading)  
    else:  
        text += heading[count]  
        Sliderlabel.config(text=text)  
        count += 1  
Sliderlabel.after(125,IntroLabelTick)
```

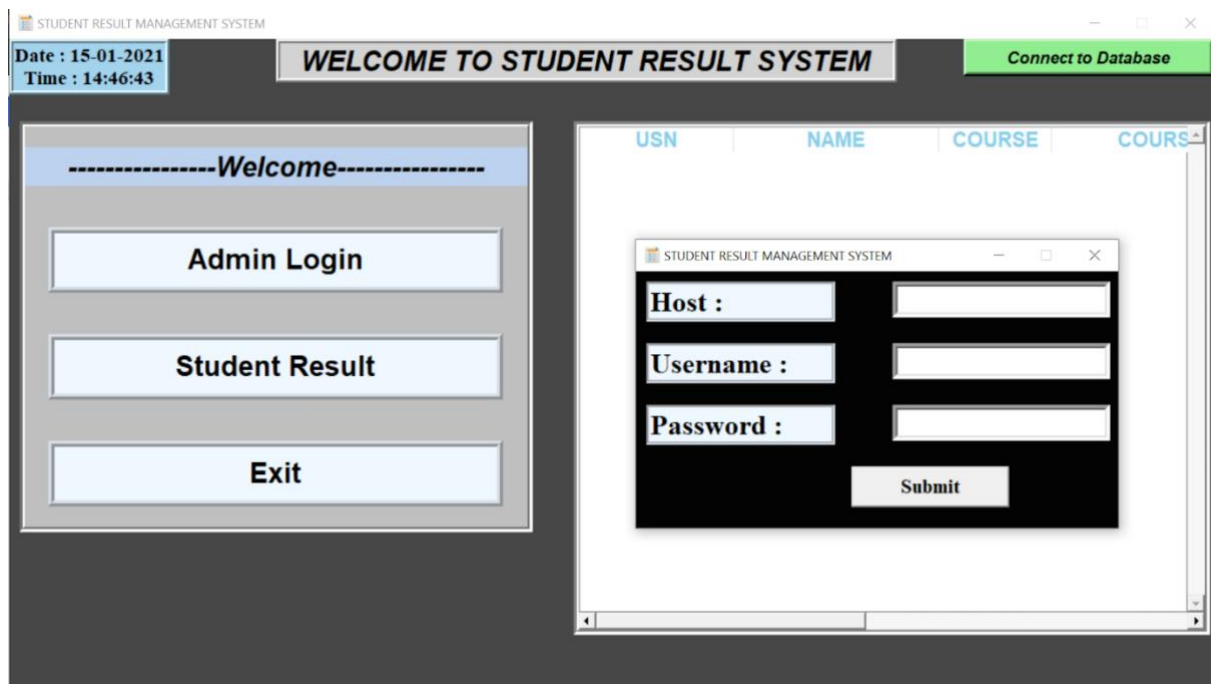
Chapter 5

SCREENSHOTS

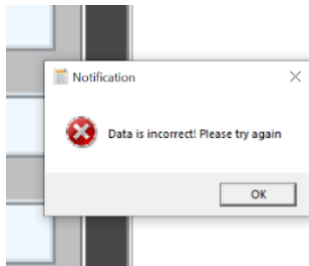
5.1 Home window:



5.2 On clicking Connect to Database button → Login window to connect to the database:



5.2.1 If login data is incorrect, error message:



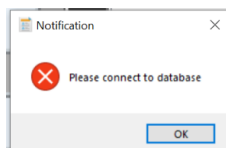
5.2.2 On successful login, pop-up message:



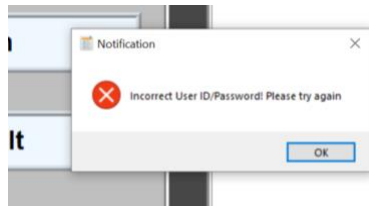
5.3 On clicking Admin Login button → Admin login window:



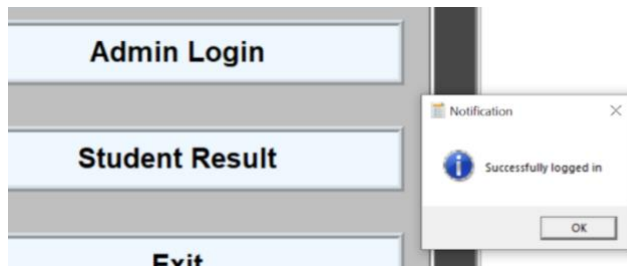
5.3.1 If not connected to database, error message:



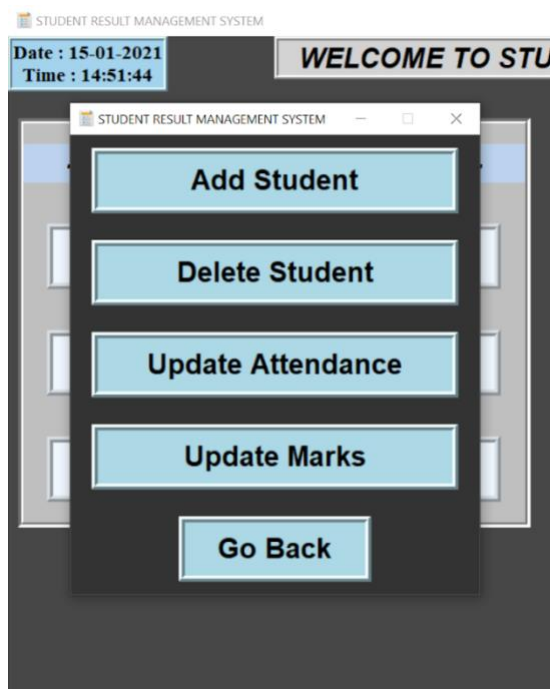
5.3.2 If Admin login data is incorrect, error message:



5.3.3 On successful Admin login, message:



5.4 Admin menu :



5.5 Add student window :

STUDENT RESULT MANAGEMENT SYSTEM

TO STUDENT RESULT SYSTEM

USN NAME

Name : Abhishek J

USN : 1CR18CS001

DOB(yyyy-mm-dd): 2000-01-01

Email : abhj18cs@cmrit.ac.in

Gender : M

Department ID : 2

Phone Number : 9912234304

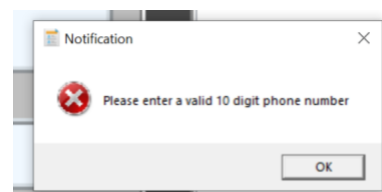
Submit

Notification

USN 1CR18CS001 added successfully...Clear form?

Yes No Cancel

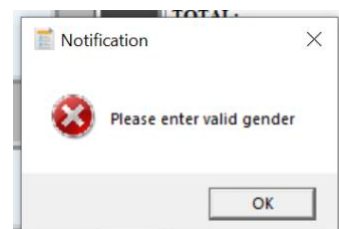
5.5.1 Error message for wrong phone number format :



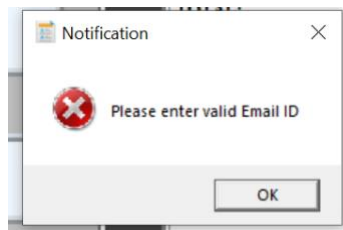
5.5.2 Error message for wrong department id format :



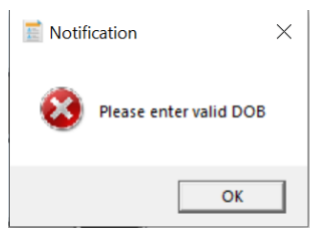
5.5.3 Error message for invalid gender format:



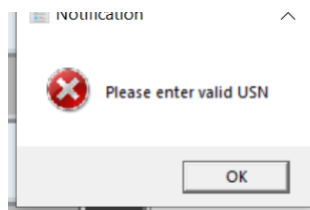
5.5.4 Error message for wrong email id format :



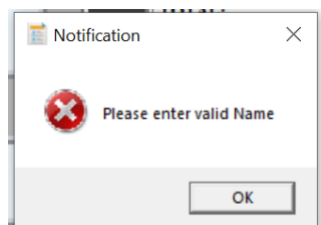
5.5.5 Error message for wrong DOB format :



5.5.6 Error message for wrong USN format :



5.5.7 Error message for wrong name format(blank) :

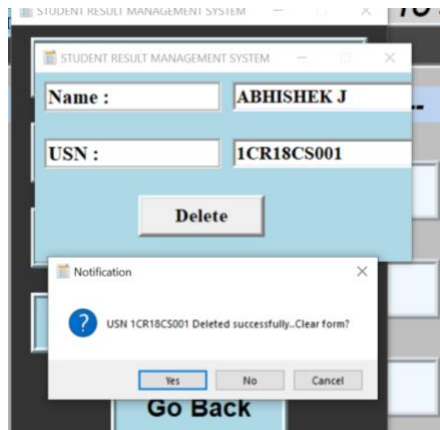


5.5.8 Total students present in the database:

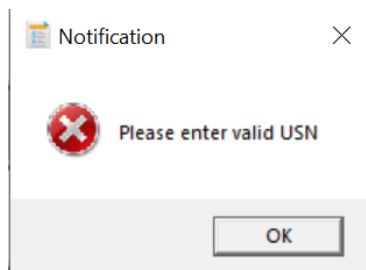
< TO STUDENT RESULT SYSTEM

USN	NAME
ICR18CS001	ABHISHEK J
ICR18CS162	SOURITA PODDAR
ICR18CS164	SRADDHA C
TOTAL:	3

5.6 Delete student window:



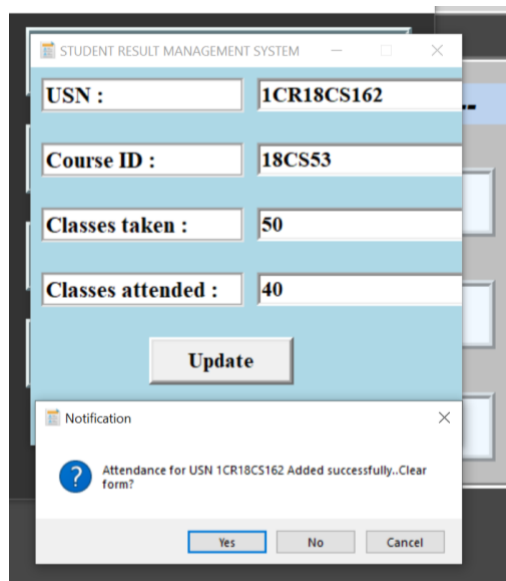
5.6.1 Error message for wrong USN format :



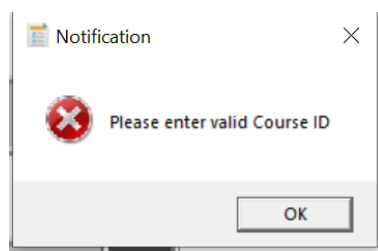
5.6.2 Total students present in the database:

USN	NAME
1CR18CS162	SOURITA PODDAR
1CR18CS164	SRADDHA C
TOTAL:	2

5.7 Attendance menu:



5.7.1 Error message for wrong Course id format :



5.7.2 Attendance display :

USN	NAME	COURSE	COURSE NAME
1CR18CS162	SOURITA PODDAR	18CS53	DATABASE MANAGEMENT SYSTEM
ATTENDANCE:	80.0		

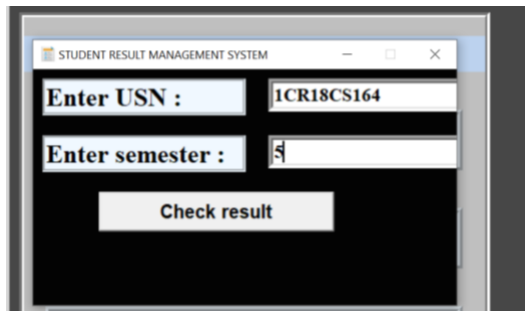
5.8 Marks menu :

5.8.1 Error message for wrong Course id format :

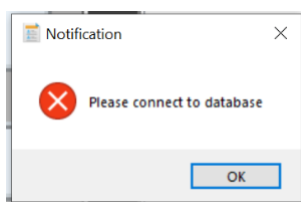
5.8.2 Marks Display:

USN	NAME	COURSE	COURSE NAME	IAT-1	IAT-2	IAT-3	IAT-4	FINAL IAT
1CR18CS164	SRADDHA C	18CS53	DATABASE MANAGEMENT SYSTEM	36	28	32	0	32

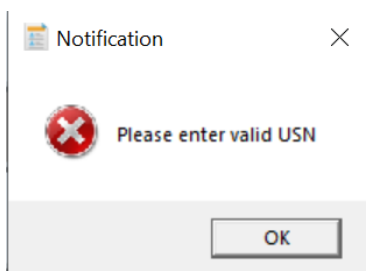
5.9 Student Result Menu:



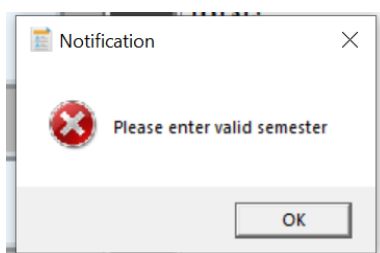
5.9.1 If not connected to database, error message:



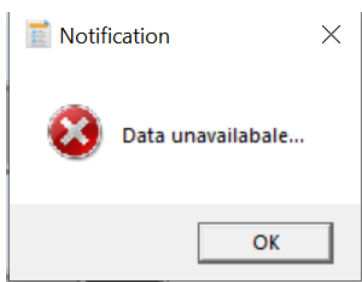
5.9.2 Error message for wrong USN format :



5.9.3 Error message for wrong semester format:



5.9.4 Error message when result data is unavailable :



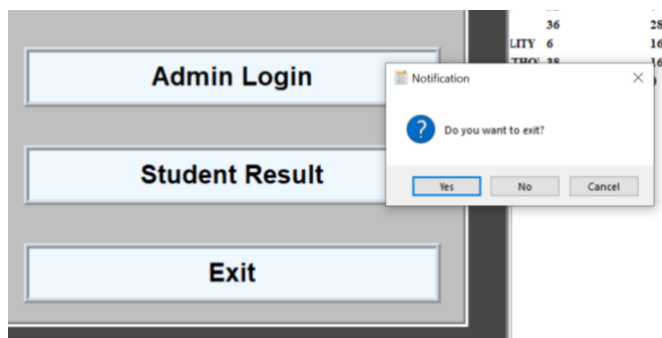
5.9.5 Result Display:

USN	NAME	COURSE	COURS
1CR18CS164	SRADDHA C	18CS51	MANAGEMENT AND E
1CR18CS164	SRADDHA C	18CS52	COMPUTER NETWORK
1CR18CS164	SRADDHA C	18CS53	DATABASE MANAGE
1CR18CS164	SRADDHA C	18CS54	AUTOMATA THEORY
1CR18CS164	SRADDHA C	18CS55	APPLICATION DEVEL
1CR18CS164	SRADDHA C	18CS56	UNIX PROGRAMMING

COURSE	COURSE NAME	IAT-1	IAT-2
18CS51	MANAGEMENT AND ENTREPRENEURSHIP FO	32	24
18CS52	COMPUTER NETWORKS AND SECURITY	32	0
18CS53	DATABASE MANAGEMENT SYSTEM	36	28
18CS54	AUTOMATA THEORY AND COMPUTABILITY	6	16
18CS55	APPLICATION DEVELOPMENT WITH PYTHON	38	16
18CS56	UNIX PROGRAMMING	38	0

	IAT-1	IAT-2	IAT-3	IAT-4	FINAL IAT
IP FO	32	24	36	0	31
	32	0	28	0	20
	36	28	32	0	32
LITY	6	16	0	24	15
THO	38	16	36	2	30
	38	0	36	24	33

5.10 Exit Window:



Chapter 6

FUTURE SCOPE & CONCLUSION

6.1 Future Scope:

- Automate the process of this Student Result Management in a much more advanced manner.
- The application must automatically verify eligibility of student to attend assessments based on their attendance.
- The faculty must be given unique login credentials rather than common administration login details.
- The students can access their assessment results without connecting to the database.
- The updation of student details without deleting pre-existing details should be allowed.
- Also, a common website/portal to be made with this same Functionality so it would be more informative and holistic for the users.
- These things would also be added for better functionality:
 - Faculty remarks
 - Common assignment submission platform
 - Online chat service between students and faculty

6.2 Conclusion:

- Presently we designed our Student Result Management System to be very User Friendly.
- Many features are enhanced to the present Student Result System.
- With this Student Result Management System most of the Faculty's and Students' time is saved.
- All the results are easy to maintain without discrepancies.
- Data security is provided by allowing alteration/updates on student data only on successful authenticated process to make user data secure which is our main concern in this era.
- The features of the system can be further enhanced in many ways.
- The documentation can enable even a person with minimum knowledge to understand it well.

BIBLIOGRAPHY

- [1] https://www.tutorialspoint.com/python/python_gui_programming.htm
- [2] <https://www.edusys.co/blog/student-result-management-system>
- [3] https://www.geeksforgeeks.org/using-cx_freeze-python/
- [4] https://www.tutorialspoint.com/plsql/plsql_triggers.htm
- [5] <https://www.sqlshack.com/sql-server-stored-procedures-for-beginners/>
- [6] **Fundamentals of DBMS - Ramez Elmasri, Shamkant B. Navathe**