DELHI PUBLIC SCHOOL PRAYAGRAJ

ACADEMIC YEAR: 2025-26

PROJECT REPORT ON:
"STUDENT MANAGEMENT SYSTEM"

ROLL NO:
NAME: ABHIRAJ MANDAL
CLASS: XII
SUBJECT: COMPUTER SCIENCE (083)

PROJECT GUIDE: Mrs. Smita Verma PGT (CS)

GitHub Repository:
https://github.com/NyteZoid/student-database-management-system

CERTIFICATE

This is to certify that student **ABHIRAJ MANDAL** of Class XII, CBSE Roll No: _____ has successfully completed the project work entitled **STUDENT MANAGEMENT SYSTEM** in the subject Computer Science (083) as laid down in the regulations of CBSE for the purpose of Practical Examination in Class XII to be held in Delhi Public School Prayagraj.

Date: _____

Signature of External Examiner
Name: _____
Examiner Number: _____

Mrs. Smita Verma (PGT Computer Science)

ACKNOWLEDGEMENT

I would like to express a deep sense of thanks and gratitude to my project guide **Mrs. Smita Verma** (PGT Computer Science), for her valuable guidance, constant encouragement, and constructive criticism throughout the duration of this project. Her mentorship helped me critically review my project and solve every problem encountered during the implementation phase.

My sincere thanks also go to our Principal, **Ms. Sujata Singh**, for providing us with the computer lab facilities and the opportunity to undertake this project.

I am also grateful to my parents for their motivation and support. I must thank all my friends for their timely help and constructive feedback.

Last but not least, I would like to express my gratitude to my previous Computer Science teacher **Mr. Mohammad Shoaib**, who has always recognised my potential and urged me to recognise it as well.

- Abhiraj Mandal

TABLE OF CONTENTS

S. NO.	DESCRIPTION	PAGE NO.
1.	Introduction	4
2.	Objectives of the Project	5
3.	Proposed System	6
4.	System Development Life Cycle (SDLC)	7
5.	Flow Chart	8
6.	Hardware & Software Requirements	9
7.	Installation Procedure	10
8.	Source Code	11
9.	Output	46
10.	Bibliography	52

INTRODUCTION

The **Student Management System** is a comprehensive software application designed to streamline the administrative tasks of a school. It provides a user-friendly Graphical User Interface (GUI) to manage student data, academic records, and performance analysis.

In the current educational landscape, managing vast amounts of student data manually is tedious and error-prone. This project aims to digitize these processes. The system is built using **Python** as the frontend programming language and **MySQL** as the backend database.

Key features of this system include:

- **Student Data Management:** Adding, updating, deleting, and searching student profiles.
- **Examination Management:** Assigning subjects and entering marks for different exams (Half Yearly, Final Exam).
- **Data Visualization:** Generating bar graphs to compare student performance across subjects.
- **Predictive Analysis:** Using Machine Learning (Linear Regression) to predict a student's future performance based on past records.
- **Report Generation:** Automatically generating PDF report cards.

This project demonstrates the effective use of Python libraries such as **`Tkinter`** for GUI, **`Matplotlib`** for graphing, **`Scikit-Learn`** for prediction, and **`MySQL Connector`** for database connectivity.

OBJECTIVES OF THE PROJECT

The primary objective of this project is to apply programming knowledge to solve a real-world problem. The specific objectives are:

1. **Automation:** To replace manual file-keeping with a digital database system, reducing paperwork and increasing data retrieval speed

2. **GUI Development:** To create an intuitive interface using Python's **`Tkinter`** library that allows users to interact with the database easily without needing knowledge of SQL commands.

3. **Data Analysis:** To implement data visualization using **`Matplotlib`**, allowing teachers to visually analyze student performance trends.

4. **Machine Learning Integration:** To demonstrate the application of basic Machine Learning algorithms (Linear Regression) using **`Scikit-learn`** to predict student performance, adding a futuristic dimension to the project.

5. **Security**: To implement password hashing using the **`hashlib`** library to ensure that user credentials are stored securely.

6. **Report Generation:** To generate printable PDF report cards dynamically using the **`reportlab`** library.

PROPOSED SYSTEM

The proposed Student Management System is designed to overcome the limitations of traditional manual record-keeping.

Deficiencies in the Manual System:

- Time-consuming data entry and retrieval.
- High risk of data redundancy and inconsistency.
- Difficulty in generating analytical reports or graphs.
- Physical storage space required for files.

Features of the Proposed System:

1. **Centralized Database:** All data regarding students, subjects, and marks is stored in a relational database (MySQL), ensuring data integrity.

2. **User Authentication:** The system uses a secure loin/registration module. Passwords are encrypted using SHA-256 hashing.

3. **Search Functionality:** Users can search for students by Roll Number, Name, Class, or Section instantly.

4. **Performance Analytics:** The system plots comparative graphs (Half Yearly vs. Final Exam) for individual students.

5. **Future Prediction:** The system analyses past exam trends to predict potential scores in upcoming exams.

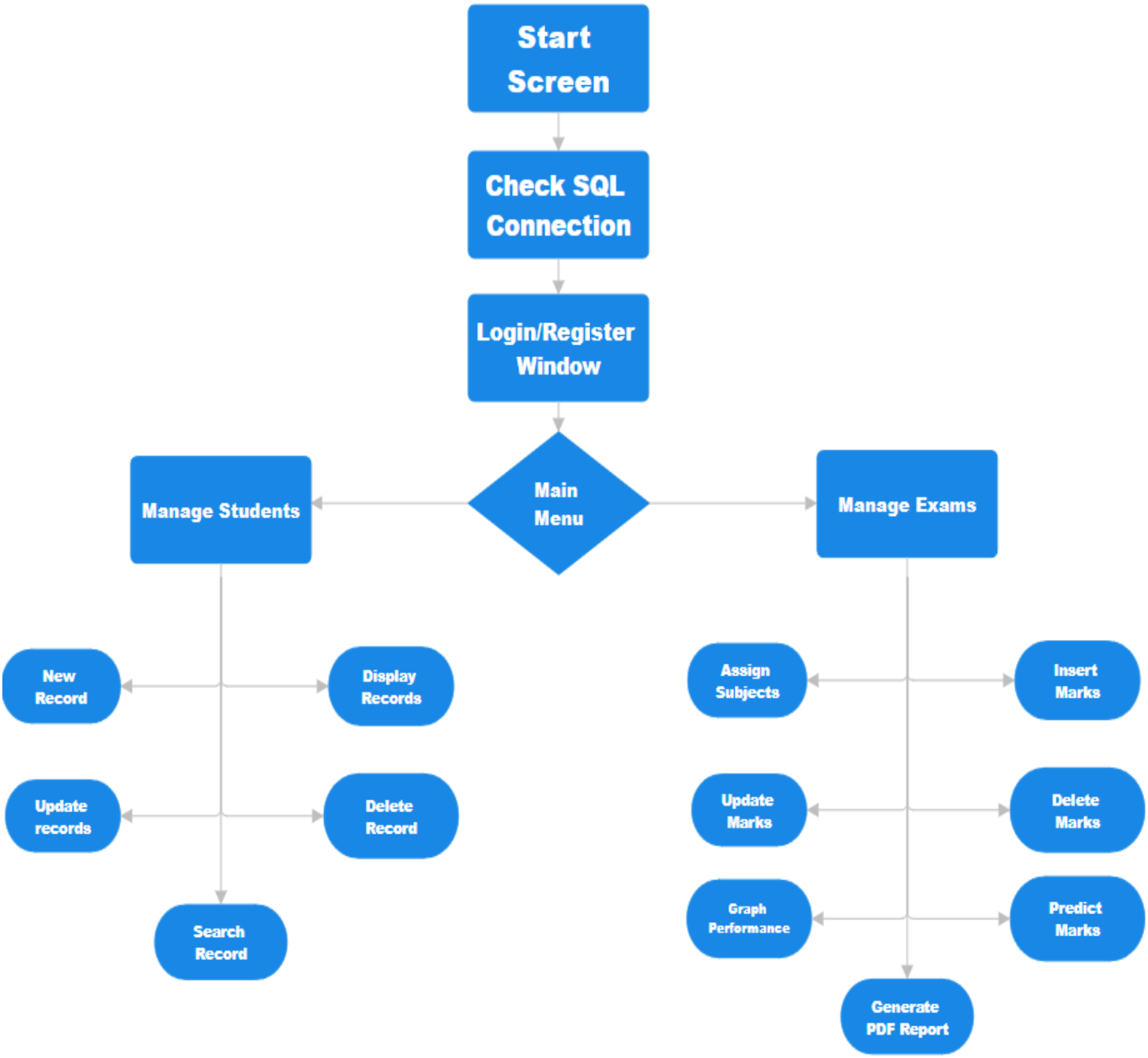
6. **Digital Report Cards:** Instead of handwritten reports, the system generates professional PDF report cards.

SYSTEM DEVELOPMENT LIFE CYCLE

The development of this project followed the standard SDLC phases:

1. **Initiation Phase:** The need for a system to manage student records efficiently was identified. The feasibility of using Python and MySQL was assessed, considering their open-source nature and ease of integration.
2. **Requirement Analysis:** Detailed requirements were gathered. It was determined that the system needed to handle student biodata, subject assignment, and marks entry. Advanced requirements included graphical analysis and marks prediction.
3. **Design Phase:**
    - **Database Design:** Three tables were designed: **`DATA`** (biodata), **`SUBJECTS`** (subject allocation), and **`MARKS`** (exam scores).
    - **UI Design:** A menu-driven interface using Tkinter was planned, separating 'Student Management' and 'Exam Management' into distinct modules.
4. **Development (Coding) Phase:** The code was written in Python. Key modules implemented:
    - **`mysql.connector`** for database linking.
    - **`tkinter`** for forms and buttons.
    - **`matplotlib`** for plotting graphs.
    - **`sklearn`** for linear regression logic.
5. **Testing Phase:** The system underwent testing to ensure:
    - Data is correctly saved to MySQL.
    - Invalid roll numbers are handled (Exception Handling).
    - Graphs render correctly based on the data.
6. **Implementation & Maintenance:** The software is packaged for execution. Maintenance involves updating the code to support new libraries or database structures if requirements change.

FLOW CHART


Start Screen
Check SQL Connection
Login/Register Window
Main Menu
Manage Students
Manage Exams
New Record
Display Records
Update records
Delete Record
Search Record
Assign Subjects
Insert Marks
Update Marks
Delete Marks
Graph Performance
Predict Marks
Generate PDF Report

HARDWARE & SOFTWARE REQUIREMENTS

To effectively run the Student Management System, the following resources are required:

HARDWARE REQUIREMENTS:

1. **Processor:** Intel Core i3 or AMD Ryzen 3 (or higher).
2. **RAM:** 4 GB or higher (8 GB recommended for smooth graph plotting).
3. **Hard Disk:** 500 MB free space.
4. **Monitor:** Standard Colour Monitor (1366x768 resolution or higher).
5. **Input Devices:** Keyboard and Mouse.

SOFTWARE REQUIREMENTS:

1. **Operating System:** Windows 10/11, macOS, or Linux.
2. **Programming Langugae:** Python 3.x.
3. **Database Server:** MySQL Server (8.0 or higher).
4. **Python Libraries Required:**
    - **`mysql-connector-python`** (Database Connectivity)
    - **`matplotlib`** (Data Visualization)
    - **`numpy`** (Numerical Calculations)
    - **`scikit-learn`** (Machine Learning)
    - **`reportlab`** (PDF Generation)
    - **`Pillow`** (Image Handling)

INSTALLATION PROCEDURE

1. **Install Python:** Download and install the latest version of Python from python.org. Ensure "Add Python to PATH" is checked during installation.

2. **Install MySQL:** Download and install MySQL Community Server. During setup, set the root password.

3. **Install Required Libraries:** Open the Command Prompt (CMD) or Terminal and execute the following command to install the dependencies:

   ```
   pip install mysql-connector-python matplotlib numpy scikit-learn reportlab pillow
   ```

4. **Setup Assets:** Ensure a folder named **`assets`** exists in the same directory as the python file. This folder must contain **`logo_dps.png`** and **`dl.png`** for the interface to load images correctly.

5. **Run the Application:** Open the file **`StudentManagementSystem.py`** in IDLE or VS Code and run the script.

6. **First Run:** On the first run, the system will ask for the MySQL password. Enter the password you set during installation. The system will automatically create the database **`SDBMS`** and the required tables.

SOURCE CODE

```
#start



#imports
import tkinter as tk
from tkinter import messagebox, ttk
import mysql.connector as sqlconn
from PIL import Image, ImageTk, ImageEnhance
import json
import os
import hashlib
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from reportlab.pdfgen import canvas



#define all file paths
prgrm = os.path.dirname(os.path.abspath(__file__))
assets = os.path.join(prgrm, "assets")
if not os.path.exists(assets):
    os.makedirs(assets)
logopath = os.path.join(assets, "logo_dps.png")
iconpath = os.path.join(assets, "dl.png")
userfilepath = os.path.join(assets, "users.json")



#password hashing and verification
def hash(password):
    return hashlib.sha256(password.encode()).hexdigest()
def verify(hashed, provided):
    return hashed == hash(provided)



#start window
def Main():
    global start
    start = tk.Tk()
    start.geometry('700x600')
    start.configure(bg = 'cornflower blue')
    start.title('STUDENT MANAGEMENT SYSTEM')
    start.resizable(False, False)
```

```
    #create canvas for layering
    canvas = tk.Canvas(start, width = 700, height = 600, bg = 'cornflower
blue', highlightthickness=0)
    canvas.pack(fill = 'both', expand = True)

    #load school logo
    logo = Image.open(logopath).resize((265,331)).convert("RGBA")
    al = logo.split()[3]
    al = ImageEnhance.Brightness(al).enhance(0.15)
    logo.putalpha(al)
    watermark = ImageTk.PhotoImage(logo)

    #place watermark on canvas
    canvas.create_image(350, 300, image=watermark)
    canvas.image = watermark

    #to handle window close event
    start.protocol("WM_DELETE_WINDOW", lambda: (start.destroy()))

    canvas.create_text(350, 50, text = 'DELHI PUBLIC SCHOOL PRAYAGRAJ',
fill = 'black', font = ('Times New Roman', 20))
    canvas.create_text(350, 120, text = 'STUDENT MANAGEMENT SYSTEM', fill =
'black', font = ('Bahnschrift bold', 30))
    canvas.create_text(350, 400, text = 'By :- Abhiraj Mandal', fill =
'black', font = ('Bahnschrift bold', 20))

    def progressbar():
        #to start progress bar
        def startpb(pb, maxval, start):
            val = pb['value']
            if val < maxval:
                pb['value'] = pb['value'] + 1
                #increments in progress bar value
                start.after(30, lambda: startpb(pb, maxval, start))
            else:
                start.withdraw()
                passcheck()

        #to create progress bar window
        def windowpb():
            s = ttk.Style()
            s.theme_use('clam')
            #style for progress bar
            s.configure("Horizontal.TProgressbar", troughcolor = 'white',
background = 'cornflower blue', bordercolor = 'white', lightcolor =
'cornflower blue', darkcolor = 'cornflower blue')

            #create border frame for progress bar
            border_frame = tk.Frame(start, bg='white')
            border_frame.place(x=195, y=495)

            #create progress bar
```

```
            pb = ttk.Progressbar(border_frame, orient = "horizontal",
length = 300, mode = "determinate")
            pb.pack(padx=5, pady=5)

            maxval = 100
            pb["maximum"] = maxval
            #start progress bar updates
            start.after(100, lambda: startpb(pb, maxval, start))

            start.protocol("WM_DELETE_WINDOW", lambda: (start.destroy()))

        windowpb()

    progressbar()
    #mainloop for start window
    start.mainloop()



#password check window
def passcheck():
    pss = tk.Toplevel()
    pss.geometry('400x200')
    pss.configure(bg = 'cornflower blue')
    pss.title('STUDENT MANAGEMENT SYSTEM')
    pss.resizable(False, False)

    pss.protocol("WM_DELETE_WINDOW", lambda: (pss.destroy(),
start.destroy()))

    tk.Label(pss, text = 'ENTER SQL PASSWORD', fg = 'black', bg =
"cornflower blue", font = ('Bahnschrift bold', 20)).place(x=60, y=20)
    n = tk.StringVar()
    T = tk.Entry(pss, fg = "black", bg = "white", textvariable = n, show =
"*", width = 20, font = ('bahnschrift semibold', 10)).place(x=125, y=80)

    #validate sql password
    def VALIDATE():
        try:
            #connect to mysql server
            global myconn
            myconn = sqlconn.connect(
                host = "localhost",
                user = "root",
                password = n.get())

            #create cursor
            global cur
            cur = myconn.cursor()
            cur.execute("CREATE DATABASE IF NOT EXISTS SDBMS;")
            cur.execute("USE SDBMS;")

            #create student data table
```

```
            cur.execute('''CREATE TABLE IF NOT EXISTS DATA(
                           roll INT PRIMARY KEY,
                           name VARCHAR(30),
                           class INT,
                           section CHAR(1),
                           gender VARCHAR(6)
                       );''')

            #create subjects table
            cur.execute('''CREATE TABLE IF NOT EXISTS SUBJECTS(
                           roll INT PRIMARY KEY,
                           subject1 VARCHAR(20),
                           subject2 VARCHAR(20),
                           subject3 VARCHAR(20),
                           subject4 VARCHAR(20),
                           subject5 VARCHAR(20)
                       );''')

            #create marks table
            cur.execute('''CREATE TABLE IF NOT EXISTS MARKS(
                           roll INT,
                           exam VARCHAR(20),
                           sub1 INT,
                           sub2 INT,
                           sub3 INT,
                           sub4 INT,
                           sub5 INT,
                           PRIMARY KEY(roll, exam)
                       );''')
            pss.destroy()
            LRForm()
        except:
            messagebox.showinfo("Access Denied", "Invalid SQL Password")
            n.set('')

    tk.Button(pss, text = "Enter", command = VALIDATE, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=145, y=130)

    pss.bind('<Return>', lambda event: VALIDATE())



#list of available subjects
subjects = ['Accountancy', 'Applied Maths', 'Biology', 'Business Studies',
'Chemistry', 'Computer Science', 'Dance',
            'Economics', 'English', 'Fine Arts', 'French', 'Geography',
'Hindi', 'History', 'Legal Studies', 'Mathematics',
            'Music', 'PE', 'Physics', 'Pol Science', 'Psychology',
'Science', 'Social Science', 'Taxation']
```

```
#load and save registered users
userfile = userfilepath
def Load():
    if os.path.exists(userfile):
        try:
            with open(userfile, 'r') as F:
                return json.load(F)
        except:
            return {}
    else:
        return {}
def Save():
    with open(userfile, 'w') as F:
        json.dump(registered, F)
registered = Load()



#login/register window
def LRForm():
    LRMenu = tk.Toplevel()
    LRMenu.geometry('400x300')
    LRMenu.configure(bg = 'cornflower blue')
    LRMenu.title('STUDENT MANAGEMENT SYSTEM')
    LRMenu.resizable(False, False)

    LRMenu.protocol("WM_DELETE_WINDOW", lambda: (LRMenu.destroy(),
start.destroy(), myconn.close()))

    tk.Label(LRMenu, text = 'LOGIN/REGISTER', fg = 'black', bg =
"cornflower blue", font = ('Bahnschrift bold', 30)).place(x=45, y=20)

    def LOGIN():
        if not registered:
            messagebox.showinfo("No Users", "No registered users found.
Please register first.")
        else:
            LRMenu.destroy()
            LoginForm()
    tk.Button(LRMenu, text = "LOGIN", command = LOGIN, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=150, y=115)

    def REGISTER():
        LRMenu.destroy()
        RegisterForm()
    tk.Button(LRMenu, text = "REGISTER", command = REGISTER, border = 3,
font = ("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
1).place(x=147, y=200)



#login window
```

```
def LoginForm():
    Myform = tk.Toplevel()
    Myform.geometry('400x300')
    Myform.configure(bg = 'cornflower blue')
    Myform.title('STUDENT MANAGEMENT SYSTEM')
    Myform.resizable(False, False)

    Myform.protocol("WM_DELETE_WINDOW", lambda: (Myform.destroy(),
start.destroy(), myconn.close()))

    tk.Label(Myform, text = 'LOGIN', fg = 'black', bg = "cornflower blue",
font = ('Bahnschrift bold', 30)).place(x=145, y=20)
    tk.Label(Myform, text = 'User Name', fg = 'black', bg = "cornflower
blue", font = ('bahnschrift semibold', 20)).place(x=52, y=98)
    tk.Label(Myform, text = 'Password', fg = 'black', bg = "cornflower
blue", font = ('bahnschrift semibold', 20)).place(x=52, y=148)

    v1 = tk.StringVar()
    v2 = tk.StringVar()
    T1 = tk.Entry(Myform, fg = "black", bg = "white", textvariable = v1,
font = ('bahnschrift semibold', 10)).place(x=202, y=110)
    T2 = tk.Entry(Myform, fg = "black", bg = "white", textvariable = v2,
show = "*",font = ('bahnschrift semibold', 10)).place(x=202, y=160)

    def BACK():
        Myform.destroy()
        LRForm()
    tk.Button(Myform, text = "Back", command = BACK, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=42, y=220)

    def CLEAR():
        v1.set('')
        v2.set('')
    tk.Button(Myform, text = "Clear", command = CLEAR, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=152, y=220)

    def VALIDATE():
        #check for valid credentials
        if v1.get() == "" or v2.get() == "":
            messagebox.showinfo("Failed", "Please fill all fields")
        else:
            if v1.get() in registered:
                if verify(registered[v1.get()],v2.get()):
                    Myform.destroy()
                    MenuForm()
                else:
                    messagebox.showinfo("Access Denied", "Invalid Username
or Password")
                    v1.set('')
                    v2.set('')
            else:
```

```
                messagebox.showinfo("Access Denied", "Invalid Username or
Password")
                v1.set('')
                v2.set('')
    tk.Button(Myform, text = "Login", command = VALIDATE, border = 3, font
= ("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=262, y=220)

    #bind enter key to validate login credentials
    Myform.bind('<Return>', lambda event: VALIDATE())



#register window
def RegisterForm():
    Rform = tk.Toplevel()
    Rform.geometry('400x300')
    Rform.configure(bg = 'cornflower blue')
    Rform.title('STUDENT MANAGEMENT SYSTEM')
    Rform.resizable(False, False)

    Rform.protocol("WM_DELETE_WINDOW", lambda: (Rform.destroy(),
start.destroy(), myconn.close()))

    tk.Label(Rform, text = 'REGISTER', fg = 'black', bg = "cornflower
blue", font = ('Bahnschrift bold', 30)).place(x=110, y=20)
    tk.Label(Rform, text = 'User Name', fg = 'black', bg = "cornflower
blue", font = ('bahnschrift semibold', 20)).place(x=52, y=98)
    tk.Label(Rform, text = 'Password', fg = 'black', bg = "cornflower
blue", font = ('bahnschrift semibold', 20)).place(x=52, y=148)

    v1 = tk.StringVar()
    v2 = tk.StringVar()
    T1 = tk.Entry(Rform, fg = "black", bg = "white", textvariable = v1,
font = ('bahnschrift semibold', 10)).place(x=202, y=110)
    T2 = tk.Entry(Rform, fg = "black", bg = "white", textvariable = v2,
show = "*",font = ('bahnschrift semibold', 10)).place(x=202, y=160)

    def BACK():
        Rform.destroy()
        LRForm()
    tk.Button(Rform, text = "Back", command = BACK, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=42, y=220)

    def CLEAR():
        v1.set('')
        v2.set('')
    tk.Button(Rform, text = "Clear", command = CLEAR, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=152, y=220)

    def VALIDATE():
```

```
        if v1.get() == "" or v2.get() == "":
            messagebox.showinfo("Failed", "Please fill all fields")
        else:
            if v1.get() in registered:
                messagebox.showinfo("Failed", "Username already exists")
                v1.set('')
                v2.set('')
            else:
                registered[v1.get()] = hash(v2.get())
                Save()
                messagebox.showinfo("Success", "Registration Successful.
Please login now.")
                Rform.destroy()
                LRForm()
    tk.Button(Rform, text = "Register", command = VALIDATE, border = 3,
font = ("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
5).place(x=262, y=220)

    #bind enter key to validate login credentials
    Rform.bind('<Return>', lambda event: VALIDATE())



#menu window
def MenuForm():
    Menu = tk.Toplevel()
    Menu.geometry('400x300')
    Menu.configure(bg = 'cornflower blue')
    Menu.title('STUDENT MANAGEMENT SYSTEM')
    Menu.resizable(False, False)

    Menu.protocol("WM_DELETE_WINDOW", lambda: (Menu.destroy(),
start.destroy(), myconn.close()))

    tk.Label(Menu, text = 'MENU', fg = 'black', bg = "cornflower blue",
font = ('Bahnschrift bold', 30)).place(x=145, y=20)

    def SMENU():
        Menu.destroy()
        StudentMenuForm()
    tk.Button(Menu, text = "Manage Students", command = SMENU, border = 3,
font = ("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=100, y=115)

    def EMENU():
        Menu.destroy()
        ExamMenuForm()
    tk.Button(Menu, text = "Manage Marks", command = EMENU, border = 3,
font = ("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
26).place(x=100, y=200)
```

```
#student menu window
def StudentMenuForm():
    SMenu = tk.Toplevel()
    SMenu.geometry('500x500')
    SMenu.configure(bg = 'seagreen3')
    SMenu.title('STUDENT MANAGEMENT SYSTEM')
    SMenu.resizable(False,False)

    SMenu.protocol("WM_DELETE_WINDOW", lambda: (SMenu.destroy(),
start.destroy(), myconn.close()))

    tk.Label(SMenu, text = 'STUDENT MENU', fg = 'black', bg = "seagreen3",
font = ('bahnschrift bold', 30)).place(x=115, y=50)

    def New():
        SMenu.destroy()
        NewForm()
    def Display():
        SMenu.destroy()
        DisplayForm()
    def Update():
        SMenu.destroy()
        UpdateForm()
    def Delete():
        SMenu.destroy()
        DeleteForm()
    def Search():
        SMenu.destroy()
        SearchForm()
    def Back():
            SMenu.destroy()
            MenuForm()

    tk.Button(SMenu, text = "NEW", command = New, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
30).place(x=90, y=180)
    tk.Button(SMenu, text = "DISPLAY", command = Display, border = 3, font
= ("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=290, y=180)
    tk.Button(SMenu, text = "UPDATE", command = Update, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=90, y=280)
    tk.Button(SMenu, text = "DELETE", command = Delete, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
18).place(x=290, y=280)
    tk.Button(SMenu, text = "SEARCH", command = Search, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=90, y=380)
    tk.Button(SMenu, text = "BACK", command = Back, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
28).place(x=290, y=380)
```

```
#new student entry form
def NewForm():
    New = tk.Toplevel()
    New.geometry('500x500')
    New.configure(bg = 'seagreen3')
    New.title('STUDENT MANAGEMENT SYSTEM')
    New.resizable(False,False)

    New.protocol("WM_DELETE_WINDOW", lambda: (New.destroy(),
start.destroy(), myconn.close()))

    tk.Label(New, text = 'NEW RECORD', fg = 'black', bg = 'seagreen3', font
= ('bahnschrift bold', 30)).place(x=120, y=20)
    try:
        #get the max roll number to assign next roll number
        cur.execute("SELECT max(roll) FROM DATA")
        result = cur.fetchone()
        nextroll = result[0] + 1
    except:
        nextroll = 101


    #set roll number variable to next roll number
    rn = tk.StringVar(value = str(nextroll))
    nm = tk.StringVar()
    cl = tk.StringVar()
    sc = tk.StringVar()
    gn = tk.StringVar()

    tk.Label(New, text='Roll Number', fg = 'black', bg = "seagreen3", font
= ('bahnschrift semibold', 20)).place(x=60,y=120)
    T1 = tk.Entry(New, fg = "black", bg = "white", textvariable = rn, state
= "readonly", font = ('bahnschrift semibold', 9)).place(x=300, y=130)
    tk.Label(New, text='Name', fg = 'black', bg = "seagreen3", font =
('bahnschrift semibold', 20)).place(x=60,y=170)
    T2 = tk.Entry(New, fg = "black", bg = "white", textvariable = nm, font
= ('bahnschrift semibold', 9)).place(x=300, y=180)
    tk.Label(New, text = 'Class', fg = 'black', bg = "seagreen3", font =
('bahnschrift semibold', 20)).place(x=60,y=220)
    T3 = ttk.Combobox(New, state = "readonly", textvariable = cl, values =
[1,2,3,4,5,6,7,8,9,10,11,12], width = 17, font = ('bahnschrift semibold',
9)).place(x=300,y=230)
    tk.Label(New, text = 'Section', fg = 'black', bg = "seagreen3", font =
('bahnschrift semibold', 20)).place(x=60,y=270)
    T4 = ttk.Combobox(New, state = "readonly", textvariable = sc, values =
["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S
","T","U","V","W","X","Y","Z"], width = 17, font = ('bahnschrift semibold',
9)).place(x=300,y=280)
    tk.Label(New, text = 'Gender', fg = 'black', bg = "seagreen3", font =
('bahnschrift semibold', 20)).place(x=60,y=320)
```

```
    T5 = ttk.Combobox(New, state = "readonly", textvariable = gn, values =
["MALE", "FEMALE", "OTHER"], width = 17, font = ('bahnschrift semibold',
9)).place(x=300,y=330)

    def BACK():
        New.destroy()
        StudentMenuForm()
    tk.Button(New, text = "Back", command = BACK, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=60, y=420)

    def CLEAR():
        nm.set('')
        cl.set('')
        sc.set('')
        gn.set('')
    tk.Button(New, text = "Clear", command = CLEAR, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=192.5, y=420)

    def VALIDATE():
        #check for empty fields
        if rn.get() == "" or nm.get() == "" or cl.get() == "" or sc.get()
== "" or gn.get() == "":
            messagebox.showinfo("Failed", "Please fill all fields")
        else:
            roll = int(rn.get())
            name =  nm.get()
            clas = int(cl.get())
            sect = sc.get()
            gend = gn.get()
            sql = "INSERT INTO DATA VALUES (%s,%s,%s,%s,%s);"
            data = (roll,name,clas,sect,gend)
            #insert new record into database
            cur.execute(sql,data)
            myconn.commit()
            messagebox.showinfo("Success","Record added")
            New.destroy()
            StudentMenuForm()
    tk.Button(New, text = "Enter", command = VALIDATE, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=325, y=420)

    New.bind('<Return>', lambda event: VALIDATE())



#delete student record form
def DeleteForm():
    Del = tk.Toplevel()
    Del.geometry('500x300')
    Del.configure(bg = 'seagreen3')
    Del.title('STUDENT MANAGEMENT SYSTEM')
```

```
    Del.resizable(False,False)

    Del.protocol("WM_DELETE_WINDOW", lambda: (Del.destroy(),
start.destroy(), myconn.close()))

    tk.Label(Del, text = 'DELETE RECORD', fg = 'black', bg = 'seagreen3',
font = ('bahnschrift bold', 30)).place(x=100, y=20)
    tk.Label(Del, text = 'Roll Number', fg = 'black',bg = "seagreen3", font
= ('bahnschrift semibold', 20)).place(x=80,y=120)
    n = tk.StringVar()
    T = tk.Entry(Del, fg = "black", bg = "white", textvariable = n, width =
10, font = ('bahnschrift semibold', 9)).place(x=320, y=133)

    def BACK():
        Del.destroy()
        StudentMenuForm()
    tk.Button(Del, text = "Back", command = BACK, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=60, y=220)

    def CLEAR():
        n.set('')
    tk.Button(Del, text = "Clear", command = CLEAR, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=192.5, y=220)

    def VALIDATE():
        #check for empty fields
        if n.get() == "":
            messagebox.showinfo("Failed", "Please fill all fields")
        else:
            cur.execute("SELECT roll FROM DATA;")
            L = cur.fetchall()
            H = []
            for x in L:
                #create list of existing roll numbers
                H.append(str(x[0]))
            #check if roll number exists
            if n.get() in H:
                #confirm deletion
                confirm = messagebox.askyesno("Confirm Delete", f"Delete
record with Roll No {n.get()}?")
                if confirm:
                    cur.execute(f"DELETE FROM DATA WHERE roll =
{n.get()};")
                    myconn.commit()
                    messagebox.showinfo("Success", "Record Deleted")
                    Del.destroy()
                    StudentMenuForm()
            else:
                #invalid roll number message
                messagebox.showinfo("Failed", "Invalid Roll Number")
```

```
    tk.Button(Del, text = "Delete", command = VALIDATE, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
13).place(x=325, y=220)

    Del.bind('<Return>', lambda event: VALIDATE())



#display student records form
def DisplayForm():
    Dis = tk.Toplevel()
    Dis.geometry('700x500')
    Dis.configure(bg = 'seagreen3')
    Dis.title('STUDENT MANAGEMENT SYSTEM')
    Dis.resizable(False, False)

    Dis.protocol("WM_DELETE_WINDOW", lambda: (Dis.destroy(),
start.destroy(), myconn.close()))

    tk.Label(Dis, text = 'DISPLAY RECORDS', fg = 'black', bg = 'seagreen3',
font = ('bahnschrift bold', 30)).place(x=180, y=20)

    style = ttk.Style()
    #set theme for treeview
    style.theme_use('clam')
    style.configure("Treeview", background = "white", foreground = "black",
rowheight = 25, fieldbackground = "white")
    #style for selected row
    style.map('Treeview', background = [('selected', 'seagreen3')])

    #create treeview for displaying records
    tree = ttk.Treeview(Dis, columns = ("roll", "name", "class", "section",
"gender"), show = 'headings', height = 12)

    #set headings for columns
    tree.heading("roll", text = "Roll No")
    tree.heading("name", text = "Name")
    tree.heading("class", text = "Class")
    tree.heading("section", text = "Section")
    tree.heading("gender", text = "Gender")

    #set column properties
    tree.column("roll", anchor = tk.CENTER, width = 80)
    tree.column("name", anchor = tk.W, width = 180)
    tree.column("class", anchor = tk.CENTER, width = 80)
    tree.column("section", anchor = tk.CENTER, width = 80)
    tree.column("gender", anchor = tk.CENTER, width = 120)

    cur.execute("SELECT * FROM DATA;")
    data = cur.fetchall()
    for row in data:
        #fetch all records and insert into treeview
        tree.insert('', 'end', values = row)
```

```
    #add scrollbar to treeview
    scroll = ttk.Scrollbar(Dis, orient = "vertical", command = tree.yview)
    tree.configure(yscrollcommand = scroll.set)
    scroll.place(x=660, y=80, height = 331)

    #place treeview on form
    tree.place(x=75, y=80)

    def BACK():
        Dis.destroy()
        StudentMenuForm()
    tk.Button(Dis, text = "Back", command = BACK, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=300, y=430)



#update student record form
def UpdateForm():
    Upd = tk.Toplevel()
    Upd.geometry('500x400')
    Upd.configure(bg = 'seagreen3')
    Upd.title('STUDENT MANAGEMENT SYSTEM')
    Upd.resizable(False, False)

    Upd.protocol("WM_DELETE_WINDOW", lambda: (Upd.destroy(),
start.destroy(), myconn.close()))

    tk.Label(Upd, text = 'UPDATE RECORD', fg = 'black', bg = 'seagreen3',
font = ('bahnschrift bold', 30)).place(x=95, y=20)

    tk.Label(Upd, text = 'Roll Number', fg = 'black',bg = "seagreen3", font
= ('bahnschrift semibold', 20)).place(x=80, y=120)
    n = tk.StringVar()
    T1 = tk.Entry(Upd, fg = "black", bg = "white", textvariable = n, width
= 14, font = ('bahnschrift semibold', 9)).place(x=320, y=133)

    tk.Label(Upd, text = 'Column', fg = 'black',bg = "seagreen3", font =
('bahnschrift semibold', 20)).place(x=80, y=180)
    col = tk.StringVar()
    T2 = ttk.Combobox(Upd, state = "readonly", textvariable = col, values =
['Name', 'Class', 'Section', 'Gender'], width = 11, font = ('bahnschrift
semibold', 9)).place(x=320, y=193)

    tk.Label(Upd, text = 'New Value', fg = 'black',bg = "seagreen3", font =
('bahnschrift semibold', 20)).place(x=80, y=240)
    uv = tk.StringVar()
    T3 = tk.Entry(Upd, fg = "black", bg = "white", textvariable = uv, width
= 14, font = ('bahnschrift semibold', 9)).place(x=320, y=253)

    def BACK():
        Upd.destroy()
```

```
        StudentMenuForm()
    tk.Button(Upd, text = "Back", command = BACK, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=60, y=320)

    def CLEAR():
        n.set('')
        col.set('')
        uv.set('')
    tk.Button(Upd, text = "Clear", command = CLEAR, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=192.5, y=320)

    def VALIDATE():
        #check for empty fields
        if n.get() == "" or col.get() == "" or uv.get() == "":
            messagebox.showinfo("Failed", "Please fill all fields")
        else:
            cur.execute("SELECT roll FROM DATA;")
            L = cur.fetchall()
            H = []
            for x in L:
                H.append(str(x[0]))
            #update record based on column type
            if n.get() in H:
                if col.get().lower() in ['name','section','gender']:
                    cur.execute(f"UPDATE DATA SET {col.get()} =
'{(uv.get())}' WHERE roll = {n.get()};")
                else:
                    cur.execute(f"UPDATE DATA SET {col.get()} =
{(uv.get())} WHERE roll = {n.get()};")
                myconn.commit()
                messagebox.showinfo("Success", "Record Updated")
                Upd.destroy()
                StudentMenuForm()
            else:
                messagebox.showinfo("Failed", "Invalid Roll Number")
    tk.Button(Upd, text = "Enter", command = VALIDATE, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=325, y=320)

    Upd.bind('<Return>', lambda event: VALIDATE())



#search student record(s) form
def SearchForm():
    Ser = tk.Toplevel()
    Ser.geometry('700x500')
    Ser.configure(bg = 'seagreen3')
    Ser.title('STUDENT MANAGEMENT SYSTEM')
    Ser.resizable(False, False)
```

```
    Ser.protocol("WM_DELETE_WINDOW", lambda: (Ser.destroy(),
start.destroy(), myconn.close()))

    tk.Label(Ser, text = 'SEARCH RECORDS', fg = 'black', bg = 'seagreen3',
font = ('bahnschrift bold', 30)).place(x=180, y=20)

    c = tk.StringVar()
    n = tk.StringVar()
    T1 = ttk.Combobox(Ser, state = "readonly", textvariable = c, values =
['Roll No', 'Name', 'Class', 'Section', 'Gender'], width = 11, font =
('bahnschrift semibold', 15)).place(x=165, y=100)
    T2 = tk.Entry(Ser, fg = "black", bg = "white", textvariable = n, width
= 10, font = ('bahnschrift semibold', 15)).place(x=425, y=100)

    def VALIDATE():
        if not c.get():
            messagebox.showinfo("Error", "Please select a search field.")
            return
        elif not n.get():
            messagebox.showinfo("Error", "Please enter a value to search.")
            return

        if c.get() == "Roll No":
            z = "roll"
        elif c.get() == "Name":
            z = "name"
        elif c.get() == "Class":
            z = "class"
        elif c.get() == "Section":
            z = "section"
        elif c.get() == "Gender":
            z = "gender"

        style = ttk.Style()
        style.theme_use('clam')
        style.configure("Treeview", background = "white", foreground =
"black", rowheight = 25, fieldbackground = "white")
        style.map('Treeview', background = [('selected', 'cornflower
blue')])

        global tree
        tree = ttk.Treeview(Ser, columns = ("roll", "name", "class",
"section", "gender"), show = 'headings', height = 8)

        tree.heading("roll", text = "Roll No")
        tree.heading("name", text = "Name")
        tree.heading("class", text = "Class")
        tree.heading("section", text = "Section")
        tree.heading("gender", text = "Gender")

        tree.column("roll", anchor = tk.CENTER, width = 80)
        tree.column("name", anchor = tk.W, width = 180)
        tree.column("class", anchor = tk.CENTER, width = 80)
```

```
        tree.column("section", anchor = tk.CENTER, width = 80)
        tree.column("gender", anchor = tk.CENTER, width = 120)

        if z in ['section','gender']:
            if n.get().isalpha() == False:
                messagebox.showinfo("Error", f"{c.get()} must be a word.")
                return
            #case insensitive search for section and gender
            cur.execute(f"SELECT * FROM DATA WHERE {z} =
'{n.get().upper()}';")
        elif z in ['name']:
            if n.get().replace(" ","").isalpha() == False:
                messagebox.showinfo("Error", f"{c.get()} must be a word.")
                return
            #partial match search for name
            cur.execute(f"SELECT * FROM DATA WHERE {z} LIKE
'%{n.get()}%';")
        else:
            if n.get().isdigit() == False:
                messagebox.showinfo("Error", f"{c.get()} must be a
number.")
                return
            #exact match search for roll no and class
            cur.execute(f"SELECT * FROM DATA WHERE {z} = {n.get()};")

        values = cur.fetchall()
        for row in values:
            tree.insert('', 'end', values = row)

        scroll = ttk.Scrollbar(Ser, orient = "vertical", command =
tree.yview)
        tree.configure(yscrollcommand = scroll.set)
        scroll.place(x=660, y=160, height = 231)

        tree.place(x=75, y=160)
    tk.Button(Ser, text = "Enter", command = VALIDATE, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=300, y=430)

    Ser.bind('<Return>', lambda event: VALIDATE())

    def BACK():
        Ser.destroy()
        StudentMenuForm()
    tk.Button(Ser, text = "Back", command = BACK, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=190, y=430)

    def CLEAR():
        n.set('')
        c.set('')
        tree.destroy()
```

```
    tk.Button(Ser, text = "Clear", command = CLEAR, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=410, y=430)



#exam menu window
def ExamMenuForm():
    EMenu = tk.Toplevel()
    EMenu.geometry('500x600')
    EMenu.configure(bg = 'salmon')
    EMenu.title('STUDENT MANAGEMENT SYSTEM')
    EMenu.resizable(False,False)

    EMenu.protocol("WM_DELETE_WINDOW", lambda: (EMenu.destroy(),
start.destroy(), myconn.close()))

    tk.Label(EMenu, text = 'EXAM MENU', fg = 'black', bg = "salmon", font =
('bahnschrift bold', 30)).place(x=135, y=50)

    def Subjects():
        EMenu.destroy()
        ExamSubjectsForm()
    def Marks():
        EMenu.destroy()
        ExamMarksForm()
    def Update():
        EMenu.destroy()
        ExamUpdateForm()
    def Delete():
        EMenu.destroy()
        ExamDeleteForm()
    def Graph():
        EMenu.destroy()
        ExamGraphForm()
    def Predict():
        EMenu.destroy()
        ExamPredictForm()
    def Report():
        EMenu.destroy()
        ExamReportForm()
    def Back():
        EMenu.destroy()
        MenuForm()

    tk.Button(EMenu, text = "SUBJECTS", command = Subjects, border = 3,
font = ("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
10).place(x=90, y=180)
    tk.Button(EMenu, text = "MARKS", command = Marks, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
21).place(x=290, y=180)
```

```
    tk.Button(EMenu, text = "UPDATE", command = Update, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
19).place(x=90, y=280)
    tk.Button(EMenu, text = "DELETE", command = Delete, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
20).place(x=290, y=280)
    tk.Button(EMenu, text = "GRAPH", command = Graph, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
22).place(x=90, y=380)
    tk.Button(EMenu, text = "PREDICT", command = Predict, border = 3, font
= ("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
16).place(x=290, y=380)
    tk.Button(EMenu, text = "REPORT", command = Report, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
19).place(x=90, y=480)
    tk.Button(EMenu, text = "BACK", command = Back, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
28).place(x=290, y=480)



#exam subjects assignment form
def ExamSubjectsForm():
    ESub = tk.Toplevel()
    ESub.geometry('500x500')
    ESub.configure(bg = 'salmon')
    ESub.title('STUDENT MANAGEMENT SYSTEM')
    ESub.resizable(False,False)

    ESub.protocol("WM_DELETE_WINDOW", lambda: (ESub.destroy(),
start.destroy(), myconn.close()))

    tk.Label(ESub, text = 'ASSIGN SUBJECTS', fg = 'black', bg = 'salmon',
font = ('bahnschrift bold', 30)).place(x=90, y=20)

    rn = tk.StringVar()
    s1 = tk.StringVar()
    s2 = tk.StringVar()
    s3 = tk.StringVar()
    s4 = tk.StringVar()
    s5 = tk.StringVar()

    tk.Label(ESub, text = 'Roll Number', fg = 'black', bg = "salmon", font
= ('bahnschrift semibold', 20)).place(x=60,y=110)
    T1 = tk.Entry(ESub, fg = "black", bg = "white", textvariable = rn, font
= ('bahnschrift semibold', 9)).place(x=300, y=120)

    tk.Label(ESub, text = 'Subject 1', fg = 'black', bg = "salmon", font =
('bahnschrift semibold', 20)).place(x=60,y=153)
    T2 = ttk.Combobox(ESub, state = "readonly", textvariable = s1, values =
subjects, width = 17, font = ('bahnschrift semibold',
9)).place(x=300,y=163)
```

```
    tk.Label(ESub, text = 'Subject 2', fg = 'black', bg = "salmon", font =
('bahnschrift semibold', 20)).place(x=60,y=196)
    T3 = ttk.Combobox(ESub, state = "readonly", textvariable = s2, values =
subjects, width = 17, font = ('bahnschrift semibold',
9)).place(x=300,y=206)

    tk.Label(ESub, text = 'Subject 3', fg = 'black', bg = "salmon", font =
('bahnschrift semibold', 20)).place(x=60,y=239)
    T4 = ttk.Combobox(ESub, state = "readonly", textvariable = s3, values =
subjects, width = 17, font = ('bahnschrift semibold',
9)).place(x=300,y=249)

    tk.Label(ESub, text = 'Subject 4', fg = 'black', bg = "salmon", font =
('bahnschrift semibold', 20)).place(x=60,y=282)
    T5 = ttk.Combobox(ESub, state = "readonly", textvariable = s4, values =
subjects, width = 17, font = ('bahnschrift semibold',
9)).place(x=300,y=292)

    tk.Label(ESub, text = 'Subject 5', fg = 'black', bg = "salmon", font =
('bahnschrift semibold', 20)).place(x=60,y=325)
    T6 = ttk.Combobox(ESub, state = "readonly", textvariable = s5, values =
subjects, width = 17, font = ('bahnschrift semibold',
9)).place(x=300,y=335)

    def BACK():
        ESub.destroy()
        ExamMenuForm()
    tk.Button(ESub, text = "Back", command = BACK, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=60, y=420)

    def CLEAR():
        rn.set('')
        s1.set('')
        s2.set('')
        s3.set('')
        s4.set('')
        s5.set('')
    tk.Button(ESub, text = "Clear", command = CLEAR, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=192.5, y=420)

    def VALIDATE():
        #check for empty fields
        if rn.get() == "" or s1.get() == "" or s2.get() == "" or s3.get()
== "" or s4.get() == "" or s5.get() == "":
            messagebox.showinfo("Failed", "Please fill all fields")
        else:
            #check for duplicate subjects
            selected = [s1.get(), s2.get(), s3.get(), s4.get(), s5.get()]
            if len(selected) != len(set(selected)):
                messagebox.showinfo("Failed", "Each subject must be unique.
Please select different subjects.")
```

```
            else:
                cur.execute("SELECT roll FROM DATA;")
                L = cur.fetchall()
                H = []
                for x in L:
                    #create list of existing roll numbers
                    H.append(str(x[0]))
                #check if roll number exists
                if rn.get() in H:
                    cur.execute("SELECT * FROM SUBJECTS WHERE roll = %s;",
(int(rn.get()),))
                    exist = cur.fetchone()
                    if exist:
                        messagebox.showinfo("Failed", "Subjects already
assigned for this student")
                        rn.set('')
                        s1.set('')
                        s2.set('')
                        s3.set('')
                        s4.set('')
                        s5.set('')
                    else:
                        roll = int(rn.get())
                        subject1 = s1.get()
                        subject2 = s2.get()
                        subject3 = s3.get()
                        subject4 = s4.get()
                        subject5 = s5.get()
                        sql = "INSERT INTO SUBJECTS
VALUES(%s,%s,%s,%s,%s,%s);"
                        data =
(roll,subject1,subject2,subject3,subject4,subject5)
                        #insert new record into database
                        cur.execute(sql,data)
                        myconn.commit()
                        messagebox.showinfo("Success","Subjects assigned")
                        ESub.destroy()
                        ExamMenuForm()
                else:
                    messagebox.showinfo("Failed", "Invalid Roll Number")
    tk.Button(ESub, text = "Enter", command = VALIDATE, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=325, y=420)

    ESub.bind('<Return>', lambda event: VALIDATE())



#exam marks entry form
def ExamMarksForm():
    EMarks = tk.Toplevel()
    EMarks.geometry('500x500')
    EMarks.configure(bg = 'salmon')
```

```
    EMarks.title('STUDENT MANAGEMENT SYSTEM')
    EMarks.resizable(False,False)

    EMarks.protocol("WM_DELETE_WINDOW", lambda: (EMarks.destroy(),
start.destroy(), myconn.close()))

    tk.Label(EMarks, text = 'INSERT MARKS', fg = 'black', bg = 'salmon',
font = ('bahnschrift bold', 30)).place(x=110, y=20)

    rn = tk.StringVar()
    ex = tk.StringVar()
    m1 = tk.StringVar()
    m2 = tk.StringVar()
    m3 = tk.StringVar()
    m4 = tk.StringVar()
    m5 = tk.StringVar()

    tk.Label(EMarks, text = 'Roll Number', fg = 'black', bg = "salmon",
font = ('bahnschrift semibold', 20)).place(x=60,y=105)
    T1 = tk.Entry(EMarks, fg = "black", bg = 'white', textvariable = rn,
font = ('bahnschrift semibold', 9), width = 10).place(x=300, y=115)

    icon = Image.open(iconpath).resize((20, 20))
    dl = ImageTk.PhotoImage(icon)

    #load subjects based on roll number
    def LOAD():
        if rn.get() == "":
            messagebox.showinfo("Failed", "Please enter Roll Number")
        else:
            cur.execute("SELECT * FROM SUBJECTS WHERE roll = %s;",
(int(rn.get()),))
            record = cur.fetchone()
            if record:
                sub1 = record[1]
                sub2 = record[2]
                sub3 = record[3]
                sub4 = record[4]
                sub5 = record[5]
                tk.Label(EMarks, text = 'Exam Name', fg = 'black', bg =
"salmon", font = ('bahnschrift semibold', 20)).place(x=60,y=148)
                T2 = ttk.Combobox(EMarks, state = "readonly", textvariable
= ex, values = ["Half Yearly", "Final Exam"], width = 17, font =
('bahnschrift semibold', 9)).place(x=300,y=158)

                tk.Label(EMarks, text = sub1, fg = 'black', bg = "salmon",
font = ('bahnschrift semibold', 20)).place(x=60,y=191)
                T3 = tk.Entry(EMarks, fg = "black", bg = "white",
textvariable = m1, font = ('bahnschrift semibold', 9)).place(x=300,y=201)

                tk.Label(EMarks, text = sub2, fg = 'black', bg = "salmon",
font = ('bahnschrift semibold', 20)).place(x=60,y=234)
```

```
                T4 = tk.Entry(EMarks, fg = "black", bg = "white",
textvariable = m2, font = ('bahnschrift semibold', 9)).place(x=300,y=244)

                tk.Label(EMarks, text = sub3, fg = 'black', bg = "salmon",
font = ('bahnschrift semibold', 20)).place(x=60,y=277)
                T5 = tk.Entry(EMarks, fg = "black", bg = "white",
textvariable = m3, font = ('bahnschrift semibold', 9)).place(x=300,y=287)

                tk.Label(EMarks, text = sub4, fg = 'black', bg = "salmon",
font = ('bahnschrift semibold', 20)).place(x=60,y=320)
                T6 = tk.Entry(EMarks, fg = "black", bg = "white",
textvariable = m4, font = ('bahnschrift semibold', 9)).place(x=300,y=330)

                tk.Label(EMarks, text = sub5, fg = 'black', bg = "salmon",
font = ('bahnschrift semibold', 20)).place(x=60,y=363)
                T7 = tk.Entry(EMarks, fg = "black", bg = "white",
textvariable = m5, font = ('bahnschrift semibold', 9)).place(x=300,y=373)
            else:
                messagebox.showinfo("Failed", "No subjects assigned for
this student")

    dlbtn = tk.Button(EMarks, text = "Load", image = dl, command = LOAD,
border = 3, font = ("bahnschrift semibold", 7), bg = "gray67", fg =
"black", padx = 15)
    dlbtn.image = dl
    dlbtn.place(x=390, y=113)

    def BACK():
        EMarks.destroy()
        ExamMenuForm()
    tk.Button(EMarks, text = "Back", command = BACK, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=60, y=420)

    def CLEAR():
        rn.set('')
        ex.set('')
        m1.set('')
        m2.set('')
        m3.set('')
        m4.set('')
        m5.set('')
    tk.Button(EMarks, text = "Clear", command = CLEAR, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=192.5, y=420)

    def VALIDATE():
        #check for empty fields
        if rn.get() == "" or ex.get() == "" or m1.get() == "" or m2.get()
== "" or m3.get() == "" or m4.get() == "" or m5.get() == "":
            messagebox.showinfo("Failed", "Please fill all fields")
        else:
            cur.execute("SELECT roll FROM DATA;")
```

```
            L = cur.fetchall()
            H = []
            for x in L:
                #create list of existing roll numbers
                H.append(str(x[0]))
            #check if roll number exists
            if rn.get() in H:
                cur.execute("SELECT * FROM MARKS WHERE roll = %s and exam =
%s;", (int(rn.get()),ex.get()))
                exist = cur.fetchone()
                if exist:
                    messagebox.showinfo("Failed", "Marks already inserted
for this student")
                else:
                    roll = int(rn.get())
                    exam = ex.get()
                    marks1 = int(m1.get())
                    marks2 = int(m2.get())
                    marks3 = int(m3.get())
                    marks4 = int(m4.get())
                    marks5 = int(m5.get())

                    #validate marks range
                    if marks1<0 or marks1>100 or marks2<0 or marks2>100 or
marks3<0 or marks3>100 or marks4<0 or marks4>100 or marks5<0 or marks5>100:
                        messagebox.showinfo("Failed", "Marks should be
between 0 and 100")
                    else:
                        sql = "INSERT INTO MARKS
VALUES(%s,%s,%s,%s,%s,%s,%s);"
                        data =
(roll,exam,marks1,marks2,marks3,marks4,marks5)
                        #insert new record into database
                        cur.execute(sql,data)
                        myconn.commit()
                        messagebox.showinfo("Success","Marks inserted")
                        EMarks.destroy()
                        ExamMenuForm()
            else:
                messagebox.showinfo("Failed", "Invalid Roll Number")
    tk.Button(EMarks, text = "Enter", command = VALIDATE, border = 3, font
= ("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=325, y=420)

    EMarks.bind('<Return>', lambda event: VALIDATE())



#exam marks update form
def ExamUpdateForm():
    EUpd = tk.Toplevel()
    EUpd.geometry('500x400')
    EUpd.configure(bg = 'salmon')
```

```
    EUpd.title('STUDENT MANAGEMENT SYSTEM')
    EUpd.resizable(False, False)

    EUpd.protocol("WM_DELETE_WINDOW", lambda: (EUpd.destroy(),
start.destroy(), myconn.close()))

    tk.Label(EUpd, text = 'UPDATE MARKS', fg = 'black', bg = 'salmon', font
= ('bahnschrift bold', 30)).place(x=95, y=20)

    tk.Label(EUpd, text = 'Roll Number', fg = 'black',bg = "salmon", font =
('bahnschrift semibold', 20)).place(x=80, y=120)
    rn = tk.StringVar()
    T1 = tk.Entry(EUpd, fg = "black", bg = "white", textvariable = rn,
width = 14, font = ('bahnschrift semibold', 9)).place(x=320, y=133)

    tk.Label(EUpd, text = 'Exam Name', fg = 'black',bg = "salmon", font =
('bahnschrift semibold', 20)).place(x=80, y=160)
    ex = tk.StringVar()
    T2 = ttk.Combobox(EUpd, state = "readonly", textvariable = ex, values =
['Half Yearly', 'Final Exam'], width = 11, font = ('bahnschrift semibold',
9)).place(x=320, y=173)

    tk.Label(EUpd, text = 'Subject', fg = 'black',bg = "salmon", font =
('bahnschrift semibold', 20)).place(x=80, y=200)
    sub = tk.StringVar()
    T2 = ttk.Combobox(EUpd, state = "readonly", textvariable = sub, values
= subjects, width = 11, font = ('bahnschrift semibold', 9)).place(x=320,
y=213)

    tk.Label(EUpd, text = 'New Marks', fg = 'black',bg = "salmon", font =
('bahnschrift semibold', 20)).place(x=80, y=240)
    mk = tk.StringVar()
    T3 = tk.Entry(EUpd, fg = "black", bg = "white", textvariable = mk,
width = 14, font = ('bahnschrift semibold', 9)).place(x=320, y=253)

    def BACK():
        EUpd.destroy()
        ExamMenuForm()
    tk.Button(EUpd, text = "Back", command = BACK, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=60, y=320)

    def CLEAR():
        rn.set('')
        ex.set('')
        sub.set('')
        mk.set('')
    tk.Button(EUpd, text = "Clear", command = CLEAR, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=192.5, y=320)

    def VALIDATE():
        #check for empty fields
```

```
        if rn.get() == "" or ex.get() == "" or sub.get() == "" or mk.get()
== "":
            messagebox.showinfo("Failed", "Please fill all fields")
        else:
            cur.execute("SELECT roll FROM MARKS;")
            L = cur.fetchall()
            H = []
            for x in L:
                H.append(str(x[0]))
            if rn.get() in H:
                #check if subject is assigned to student
                cur.execute("SELECT subject1, subject2, subject3, subject4,
subject5 FROM SUBJECTS WHERE roll = %s", (rn.get(),))
                subs = cur.fetchone()
                collist = ["sub1", "sub2", "sub3", "sub4", "sub5"]
                sublist = list(subs)
                if sub.get() not in sublist:
                    messagebox.showinfo("Failed", f"{sub.get()} not
assigned for this student")
                else:
                    #update marks in the corresponding subject column
                    index = sublist.index(sub.get())
                    col = collist[index]
                    cur.execute(f"UPDATE MARKS SET {col} =
{(int(mk.get()))} WHERE roll = {rn.get()} AND exam = '{ex.get()}';")
                    myconn.commit()
                    messagebox.showinfo("Success", "Marks Updated")
                    EUpd.destroy()
                    ExamMenuForm()
            else:
                messagebox.showinfo("Failed", "Invalid Roll Number")
    tk.Button(EUpd, text = "Enter", command = VALIDATE, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=325, y=320)

    EUpd.bind('<Return>', lambda event: VALIDATE())



#exam marks deletion form
def ExamDeleteForm():
    EDel = tk.Toplevel()
    EDel.geometry('500x300')
    EDel.configure(bg = 'salmon')
    EDel.title('STUDENT MANAGEMENT SYSTEM')
    EDel.resizable(False,False)

    EDel.protocol("WM_DELETE_WINDOW", lambda: (EDel.destroy(),
start.destroy(), myconn.close()))

    tk.Label(EDel, text = 'DELETE MARKS', fg = 'black', bg = 'salmon', font
= ('bahnschrift bold', 30)).place(x=100, y=20)
```

```
    tk.Label(EDel, text = 'Roll Number', fg = 'black',bg = "salmon", font =
('bahnschrift semibold', 20)).place(x=80,y=95)
    rn = tk.StringVar()
    T1 = tk.Entry(EDel, fg = "black", bg = "white", textvariable = rn,
width = 13, font = ('bahnschrift semibold', 9)).place(x=320, y=108)
    tk.Label(EDel, text = 'Exam Name', fg = 'black',bg = "salmon", font =
('bahnschrift semibold', 20)).place(x=80,y=145)
    ex = tk.StringVar()
    T2 = ttk.Combobox(EDel, state = "readonly", textvariable = ex, values =
['Half Yearly', 'Final Exam'], width = 11, font = ('bahnschrift semibold',
9)).place(x=320, y=158)

    def BACK():
        EDel.destroy()
        ExamMenuForm()
    tk.Button(EDel, text = "Back", command = BACK, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=60, y=220)

    def CLEAR():
        rn.set('')
        ex.set('')
    tk.Button(EDel, text = "Clear", command = CLEAR, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=192.5, y=220)

    def VALIDATE():
        #check for empty fields
        if rn.get() == "" or ex.get() == "":
            messagebox.showinfo("Failed", "Please fill all fields")
        else:
            cur.execute("SELECT roll FROM MARKS;")
            L = cur.fetchall()
            H = []
            for x in L:
                #create list of existing roll numbers
                H.append(str(x[0]))
            #check if roll number exists
            if rn.get() in H:
                #confirm deletion
                confirm = messagebox.askyesno("Confirm Delete", "Delete
marks?")
                if confirm:
                    cur.execute(f"DELETE FROM MARKS WHERE roll = {rn.get()}
AND exam = '{ex.get()}';")
                    myconn.commit()
                    messagebox.showinfo("Success", "Marks Deleted")
                    EDel.destroy()
                    ExamMenuForm()
            else:
                #invalid roll number message
                messagebox.showinfo("Failed", "Invalid Roll Number")
```

```
    tk.Button(EDel, text = "Delete", command = VALIDATE, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
13).place(x=325, y=220)

    EDel.bind('<Return>', lambda event: VALIDATE())



def ExamGraphForm():
    Grph = tk.Toplevel()
    Grph.geometry('500x300')
    Grph.configure(bg = 'salmon')
    Grph.title('STUDENT MANAGEMENT SYSTEM')
    Grph.resizable(False,False)

    Grph.protocol("WM_DELETE_WINDOW", lambda: (Grph.destroy(),
start.destroy(), myconn.close()))

    tk.Label(Grph, text = 'PLOT GRAPH', fg = 'black', bg = 'salmon', font =
('bahnschrift bold', 30)).place(x=100, y=20)
    tk.Label(Grph, text = 'Roll Number', fg = 'black',bg = "salmon", font =
('bahnschrift semibold', 20)).place(x=80,y=120)
    n = tk.StringVar()
    T = tk.Entry(Grph, fg = "black", bg = "white", textvariable = n, width
= 10, font = ('bahnschrift semibold', 9)).place(x=320, y=133)

    def BACK():
        Grph.destroy()
        ExamMenuForm()
    tk.Button(Grph, text = "Back", command = BACK, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=60, y=220)

    def CLEAR():
        n.set('')
    tk.Button(Grph, text = "Clear", command = CLEAR, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=192.5, y=220)

    def VALIDATE():
        #check for empty fields
        if n.get() == "":
            messagebox.showinfo("Failed", "Please fill all fields")
        else:
            cur.execute("SELECT roll FROM MARKS;")
            L = cur.fetchall()
            H = []
            for x in L:
                #create list of existing roll numbers
                H.append(str(x[0]))
            #check if roll number exists
            if n.get() in H:
```

```
                cur.execute("SELECT subject1, subject2, subject3, subject4,
subject5 FROM SUBJECTS WHERE roll = %s;", (int(n.get()),))
                subs = cur.fetchone()
                if not subs:
                    messagebox.showinfo("Failed", "No subjects assigned for
this student")
                else:
                    subslist = list(subs)
                    cur.execute("SELECT exam, sub1, sub2, sub3, sub4, sub5
FROM MARKS WHERE roll = %s;", (int(n.get()),))
                    recs = cur.fetchall()
                    if not recs:
                        messagebox.showinfo("Failed", "No marks found for
this student")
                    else:
                        #prepare data for plotting
                        hy = []
                        fe = []
                        for rec in recs:
                            ex = rec[0]
                            m = list(rec[1:])

                            #separate marks based on exam type
                            if ex == "Half Yearly":
                                hy = m
                            elif ex == "Final Exam":
                                fe = m

                        if hy and fe:
                            #plot graph using matplotlib
                            plt.figure(figsize=(10,6))
                            index = np.arange(5)
                            width = 0.35
                            plt.bar(index - width/2, hy, width, label='Half
Yearly', color='blue')
                            plt.bar(index + width/2, fe, width,
label='Final Exam', color='orange')
                            plt.xlabel('Subjects')
                            plt.ylabel('Marks')
                            plt.xticks(index, subslist, ha='center')
                            cur.execute("SELECT name FROM DATA WHERE roll =
%s;", (int(n.get()),))
                            nm = cur.fetchone()
                            plt.title(f'Subject wise marks comparison for
{nm[0]} ({n.get()})')
                            plt.ylim(0, 105)
                            plt.legend()

                            #add data labels on top of bars
                            for i in range(5):
                                plt.text(index[i] - width/2, hy[i] + 2,
str(hy[i]), ha = 'center')
```

```
                                plt.text(index[i] + width/2, fe[i] + 2,
str(fe[i]), ha = 'center')

                            plt.tight_layout()
                            plt.show()
                        else:
                            messagebox.showinfo("Failed", "Not enough data
to plot graph")
            else:
                #invalid roll number message
                messagebox.showinfo("Failed", "Not enough data to plot
graph")

    tk.Button(Grph, text = "Plot", command = VALIDATE, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=325, y=220)

    Grph.bind('<Return>', lambda event: VALIDATE())



def ExamPredictForm():
    Pred = tk.Toplevel()
    Pred.geometry('500x300')
    Pred.configure(bg = 'salmon')
    Pred.title('STUDENT MANAGEMENT SYSTEM')
    Pred.resizable(False,False)

    Pred.protocol("WM_DELETE_WINDOW", lambda: (Pred.destroy(),
start.destroy(), myconn.close()))

    tk.Label(Pred, text = 'PREDICT MARKS', fg = 'black', bg = 'salmon',
font = ('bahnschrift bold', 30)).place(x=95, y=20)
    tk.Label(Pred, text = 'Roll Number', fg = 'black',bg = "salmon", font =
('bahnschrift semibold', 20)).place(x=80,y=120)
    n = tk.StringVar()
    T = tk.Entry(Pred, fg = "black", bg = "white", textvariable = n, width
= 10, font = ('bahnschrift semibold', 9)).place(x=320, y=133)

    def BACK():
        Pred.destroy()
        ExamMenuForm()
    tk.Button(Pred, text = "Back", command = BACK, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=60, y=220)

    def CLEAR():
        n.set('')
    tk.Button(Pred, text = "Clear", command = CLEAR, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=192.5, y=220)

    def VALIDATE():
```

```
        #check for empty fields
        if n.get() == "":
            messagebox.showinfo("Failed", "Please fill all fields")
        else:
            cur.execute("SELECT roll FROM MARKS;")
            L = cur.fetchall()
            H = []
            for x in L:
                #create list of existing roll numbers
                H.append(str(x[0]))
            #check if roll number exists
            if n.get() in H:
                cur.execute("SELECT subject1, subject2, subject3, subject4,
subject5 FROM SUBJECTS WHERE roll = %s;", (int(n.get()),))
                subs = cur.fetchone()
                if not subs:
                    messagebox.showinfo("Failed", "No subjects assigned for
this student")
                else:
                    subslist = list(subs)
                    cur.execute("SELECT exam, sub1, sub2, sub3, sub4, sub5
FROM MARKS WHERE roll = %s;", (int(n.get()),))
                    recs = cur.fetchall()
                    if not recs:
                        messagebox.showinfo("Failed", "Not enough data to
predict marks")
                    else:
                        hy = []
                        fe = []
                        for rec in recs:
                            ex = rec[0]
                            m = list(rec[1:])

                            if ex == "Half Yearly":
                                hy = m
                            elif ex == "Final Exam":
                                fe = m

                        if hy and fe:
                            cur.execute("SELECT name FROM DATA WHERE roll =
%s;", (int(n.get()),))
                            nm = cur.fetchone()

                            predicted = []

                            #predict marks using linear regression
                            for i in range(5):
                                x = np.array([[1], [2]])
                                y = np.array([hy[i], fe[i]])

                                model = LinearRegression()
                                model.fit(x,y)
```

```
                                next = model.predict([[3]])[0]
                                next = max(0, min(100, next))
                                predicted.append(round(next,2))

                            plt.figure(figsize=(12,7))

                            exams = ['Half Yearly', 'Final Exam',
'Predicted']
                            xpos = np.arange(3)

                            colours = ['blue', 'green', 'red', 'purple',
'orange']

                            #plot marks for each subject
                            for i in range(5):
                                marksp = [hy[i], fe[i], predicted[i]]
                                plt.plot(xpos, marksp, marker = 'o', label
= subslist[i], color = colours[i])

                                for k, v in enumerate(marksp):
                                    plt.text(xpos[k], v + 2, f'{v:.1f}', ha
= 'center')

                            plt.xlabel('Exams')
                            plt.ylabel('Marks')
                            plt.title(f'Marks Prediction for {nm[0]}
({n.get()})')
                            plt.xticks(xpos, exams)
                            plt.ylim(0, 110)
                            plt.legend()
                            plt.grid()

                            plt.tight_layout()
                            plt.show()
                        else:
                            messagebox.showinfo("Failed", "Not enough data
to predict marks")
            else:
                #invalid roll number message
                messagebox.showinfo("Failed", "Invalid Roll Number")

    tk.Button(Pred, text = "Predict", command = VALIDATE, border = 3, font
= ("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
14).place(x=325, y=220)

    Pred.bind('<Return>', lambda event: VALIDATE())



def ExamReportForm():
    Rep = tk.Toplevel()
    Rep.geometry('400x250')
    Rep.configure(bg = 'salmon')
```

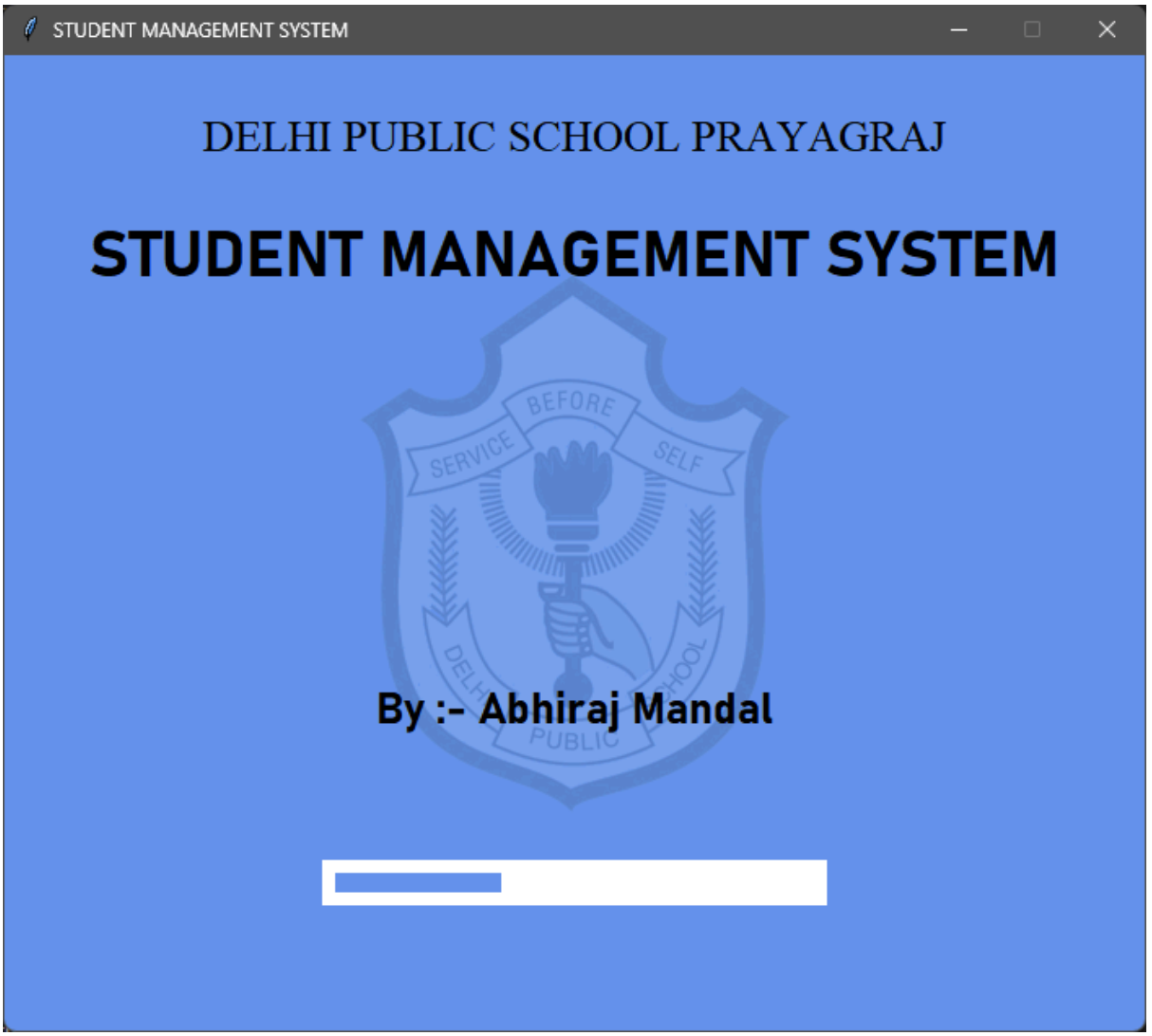
```
    Rep.title('STUDENT MANAGEMENT SYSTEM')
    Rep.resizable(False,False)

    Rep.protocol("WM_DELETE_WINDOW", lambda: (Rep.destroy(),
start.destroy(), myconn.close()))

    tk.Label(Rep, text = 'REPORT CARD', fg = 'black', bg = 'salmon', font =
('bahnschrift bold', 25)).place(x=100, y=20)
    tk.Label(Rep, text = 'Roll Number', fg = 'black', bg = "salmon", font =
('bahnschrift semibold', 20)).place(x=50,y=100)
    n = tk.StringVar()
    T = tk.Entry(Rep, fg = "black", bg = "white", textvariable = n, width =
10, font = ('bahnschrift semibold', 9)).place(x=270, y=113)

    def BACK():
        Rep.destroy()
        ExamMenuForm()
    tk.Button(Rep, text = "Back", command = BACK, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=45, y=180)

    def CLEAR():
        n.set('')
    tk.Button(Rep, text = "Clear", command = CLEAR, border = 3, font =
("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
15).place(x=145, y=180)

    #function to determine grade based on marks
    def grade(m):
        if m >= 90:
            return 'A+'
        elif m >= 80:
            return 'A'
        elif m >= 70:
            return 'B+'
        elif m >= 60:
            return 'B'
        elif m >= 50:
            return 'C+'
        elif m >= 40:
            return 'C'
        elif m >= 33:
            return 'D'
        else:
            return 'F'

    def GENERATE():
        if n.get() == "":
            messagebox.showinfo("Failed", "Please fill all fields")
        else:
            roll = int(n.get())
            cur.execute("SELECT name, class, section FROM DATA WHERE roll =
%s;", (roll,))
```

```
            data = cur.fetchone()
            if not data:
                messagebox.showinfo("Failed", "Invalid Roll Number")
            else:
                name = data[0]
                cls = data[1]
                sec = data[2]
                cur.execute("SELECT subject1, subject2, subject3, subject4,
subject5 FROM SUBJECTS WHERE roll = %s;", (roll,))
                subs = cur.fetchone()
                if not subs:
                    messagebox.showinfo("Failed", "No subjects assigned for
this student")
                else:
                    subslist = list(subs)
                    cur.execute("SELECT exam, sub1, sub2, sub3, sub4, sub5
FROM MARKS WHERE roll = %s;", (roll,))
                    recs = cur.fetchall()
                    if not recs:
                        messagebox.showinfo("Failed", "No marks found for
this student")
                    else:
                        marks = {'Half Yearly': None, 'Final Exam': None}
                        for rec in recs:
                            marks[rec[0]] = list(rec[1:])

                        hyp = None
                        fep = None

                        if marks['Half Yearly']:
                            hyp = sum(marks['Half Yearly'])/5
                        if marks['Final Exam']:
                            fep = sum(marks['Final Exam'])/5

                        #generate PDF report card using reportlab
                        filenm = f'Report_Card_Roll_{roll}.pdf'
                        c = canvas.Canvas(filenm)

                        y = 800
                        c.setFont("Times-Roman", 16)
                        c.drawString(50, y, "DPS PRAYAGRAJ")
                        y = y - 40
                        c.drawString(50, y, "REPORT CARD")
                        y = y - 40
                        c.drawString(50, y, f"Name: {name}")
                        y = y - 20
                        c.drawString(50, y, f"Class: {cls}-{sec}")
                        y = y - 20
                        c.drawString(50, y, f"Roll Number: {roll}")
                        y = y - 30
                        c.drawString(50, y , "Subject wise performance:")
                        y = y - 25
```

```
                        #display marks and grades for each subject
                        for i in range(5):
                            hy = marks['Half Yearly'][i] if marks['Half
Yearly'] else '-'
                            fe = marks['Final Exam'][i] if marks['Final
Exam'] else '-'
                            hyg = grade(hy) if hy != '-' else '-'
                            feg = grade(fe) if fe != '-' else '-'
                            c.drawString(50, y, f"{subslist[i]}: HY: {hy}
({hyg}), FE: {fe} ({feg})")
                            y = y - 20

                        #display overall percentages
                        y = y - 20
                        if hyp is not None:
                            c.drawString(50, y, f"Half Yearly percentage:
{hyp:.2f}%")
                            y = y - 20
                        if fep is not None:
                            c.drawString(50, y, f"Final Exam percentage:
{fep:.2f}%")
                            y = y - 20

                        c.save()
                        messagebox.showinfo("Success", f"Report card
generated as {filenm}")

    tk.Button(Rep, text = "Generate", command = GENERATE, border = 3, font
= ("bahnschrift semibold", 15), bg = "gray67", fg = "black", padx =
1).place(x=250, y=180)

    Rep.bind("<Return>", lambda event: GENERATE())



#start the application
Main()



#end
```

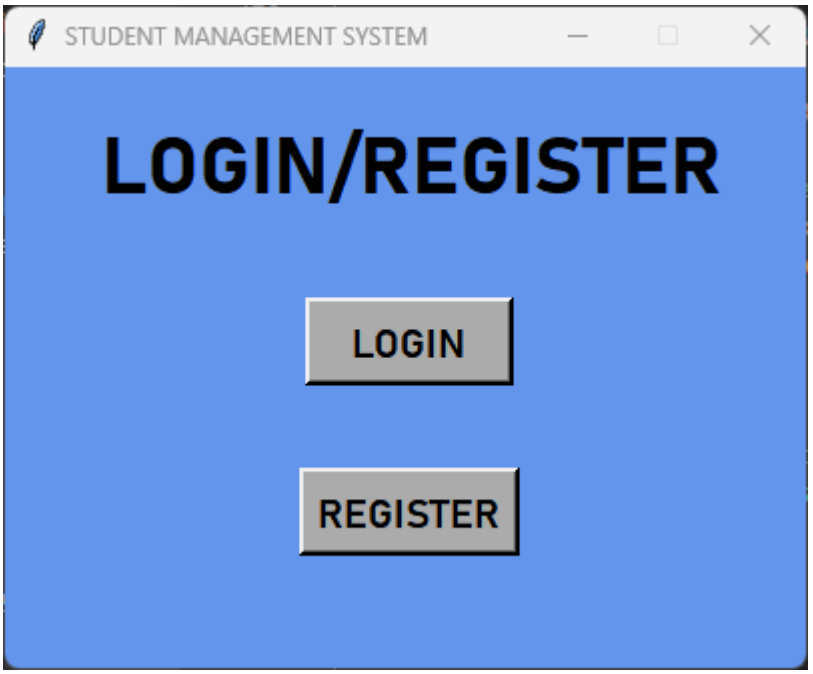
OUTPUT

1. Loading Screen

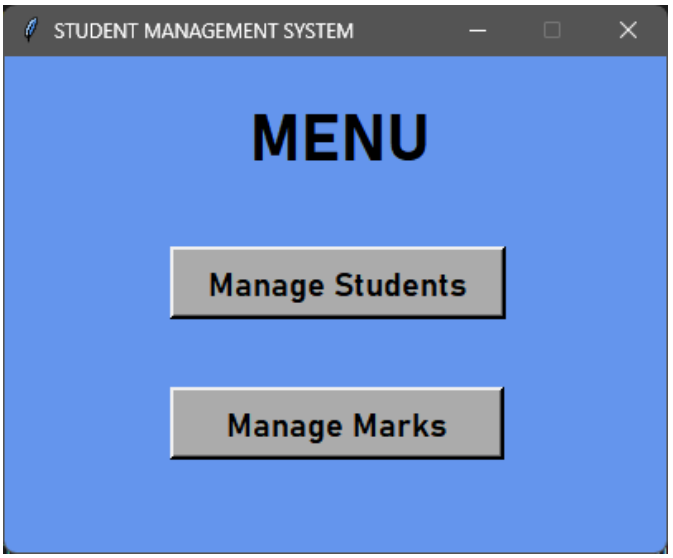


2. SQL Password Entry

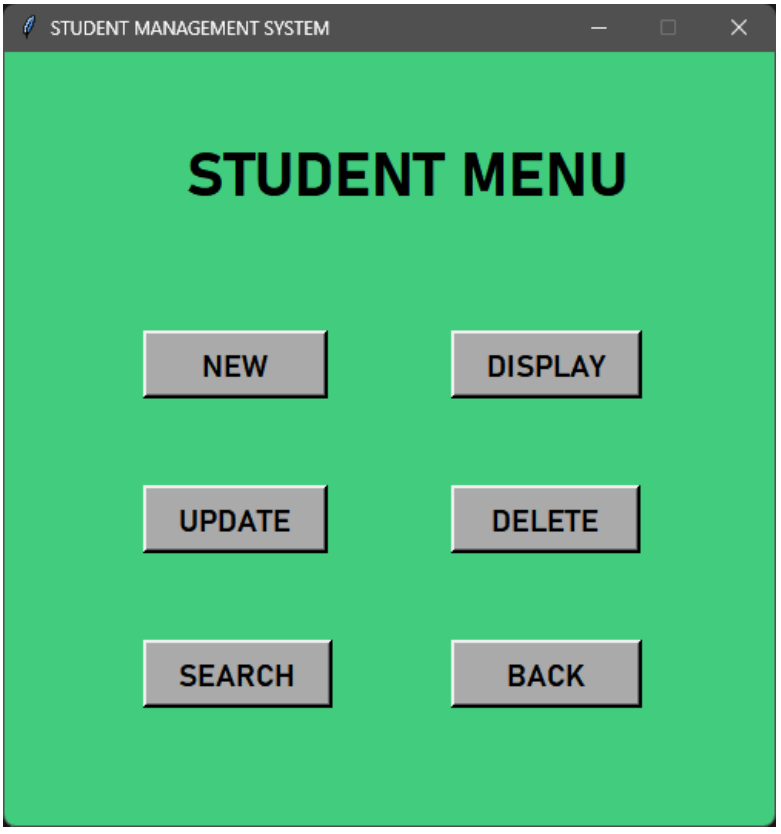
3. Login/Register Screen



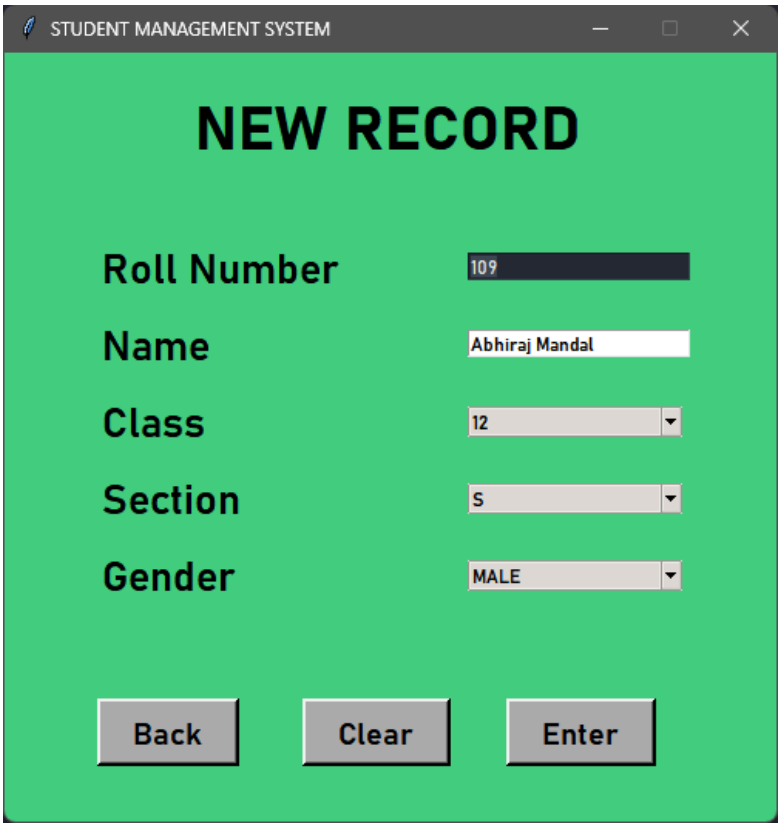
4. Main Menu



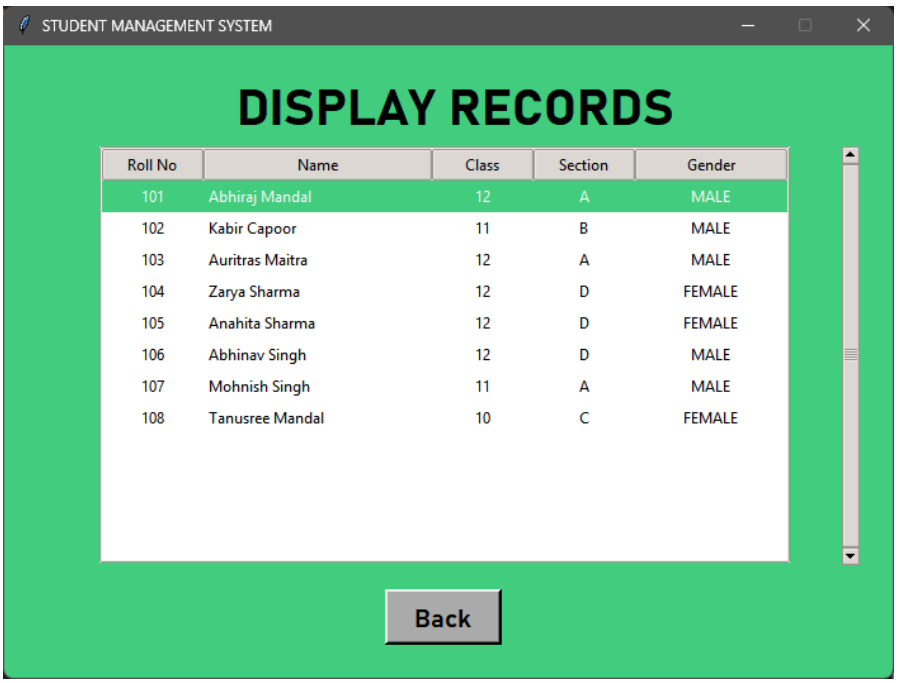
5. Student Menu



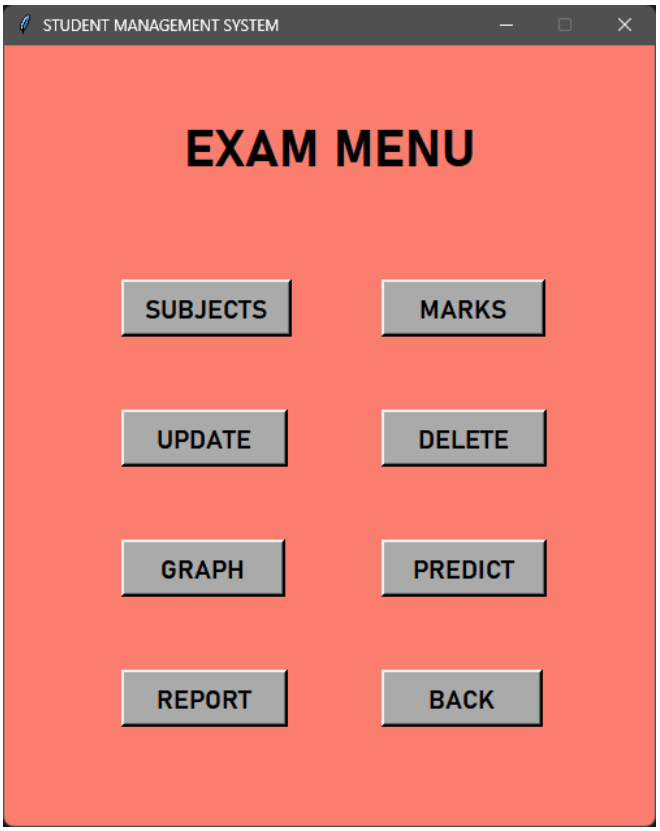
6. New Student Form



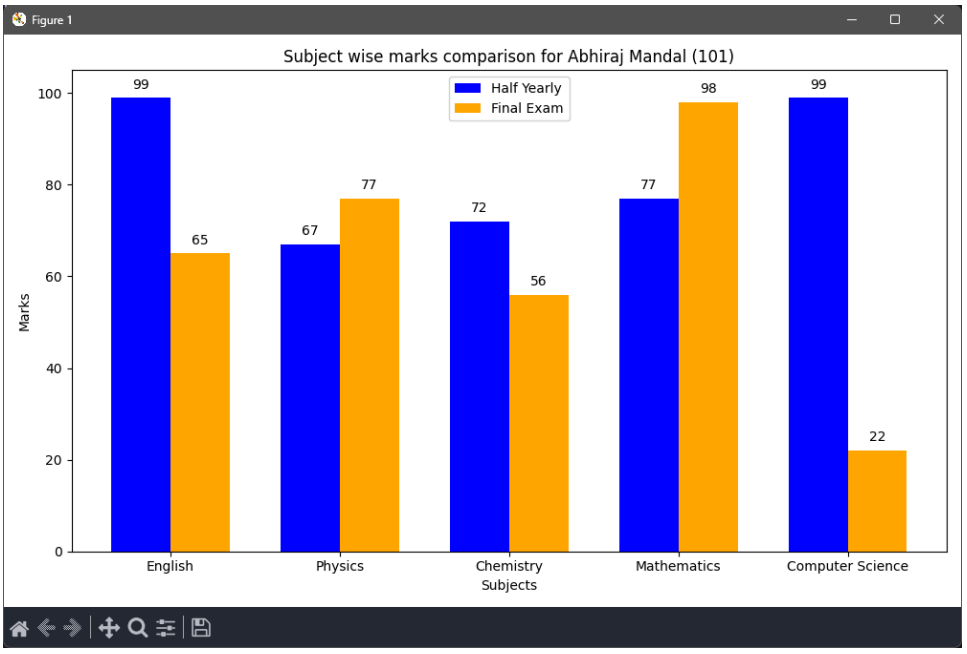
7. Display Records



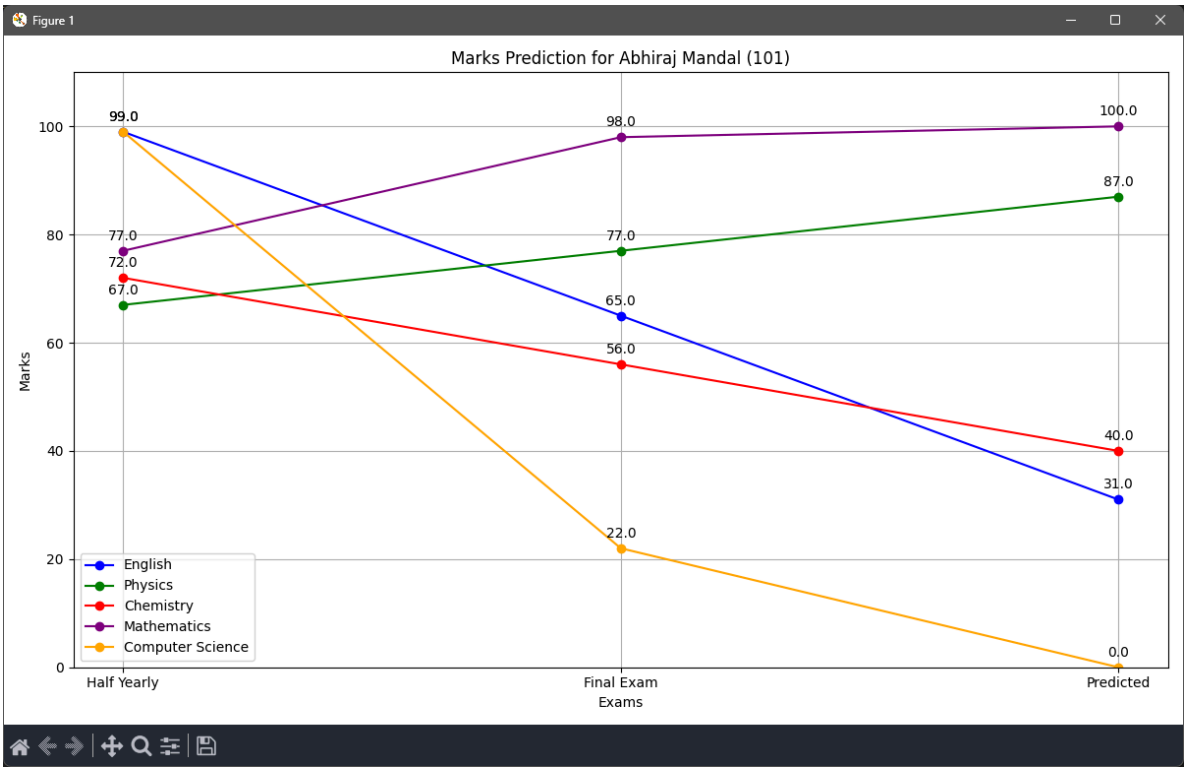
8. Exam Menu



9. Graph Output



10. Prediction Output



11. Generated PDF Report

DPS PRAYAGRAJ

REPORT CARD

Name: Abhiraj Mandal
Class: 12-A
Roll Number: 101

Subject wise performance:

English: HY: 99 (A+), FE: 65 (B)
Physics: HY: 67 (B), FE: 77 (B+)
Chemistry: HY: 72 (B+), FE: 56 (C+)
Mathematics: HY: 77 (B+), FE: 98 (A+)
Computer Science: HY: 99 (A+), FE: 22 (F)

Half Yearly percentage: 82.80%
Final Exam percentage: 63.60%

BIBLIOGRAPHY

The following resources were consulted for the successful completion of the project:

Books:

1. *Computer Science with Python* - Class XII by Sumita Arora.

Websites:

1. Python Documentation: docs.python.org
2. MySQL Documentation: dev.mysql.com/doc
3. Matplotlib Tutorials: matplotlib.org
4. Stack Overflow: stackoverflow.com (For debugging and error resolution)
5. GeeksforGeeks: geeksforgeeks.org (For algorithm logic)

Software Tools:

1. VS Code (IDE)
2. MySql Workbench 8.0