A **Student Management System (SMS)** is a software application that helps educational institutions manage and streamline their academic and administrative tasks. The system is designed to manage all student data, including personal information, academic records, attendance records, health records, and disciplinary records. The SMS helps track, store, maintain, and disseminate all this data safely and securely to students, staff, and parents .

Here is an outline of an assignment on Student Management System where the administrator manages the student list:

I. Introduction

- A Student Management System (SMS) is a comprehensive and integrated software solution designed to streamline and automate the management of student-related information within educational institutions. This sophisticated system serves as a centralized hub for storing, organizing, and processing a wide array of data pertaining to students, offering a holistic approach to student administration. From admission to graduation, an SMS plays a pivotal role in facilitating efficient academic and administrative processes
- At its core, an SMS is designed to handle the entire student lifecycle. The system begins by managing the admission process, capturing essential information during enrollment, including personal details, academic records, and contact information. This initial phase ensures a seamless onboarding experience for new students and provides administrators with a solid foundation for academic planning.
- Once students are admitted, the system excels in its ability to dynamically update and maintain student records. This includes capturing changes in personal details, academic progress, extracurricular activities, and any other relevant data. Regular updates ensure that the information stored in the system remains accurate and up-to-date, allowing educational institutions to make informed decisions regarding student performance and overall well-being.
- The heart of the SMS lies in its data display capabilities, offering a user-friendly interface for authorized personnel to access and retrieve student information effortlessly. Teachers, administrators, and staff can use the system to search for specific student records, track academic performance, and monitor attendance, fostering a collaborative and data-driven approach to education.
- Moreover, the SMS incorporates features for exporting data, enabling users to extract information for reporting, analysis, and compliance purposes. This export functionality supports educational institutions in meeting regulatory requirements

and enhances their ability to generate insightful reports on various aspects of student life and performance

- An essential aspect of the SMS is its secure deletion module, which allows authorized users to remove or update student records when necessary. This feature is implemented with caution, including confirmation mechanisms and audit trails, to prevent accidental or unauthorized data loss.
- In conclusion, a Student Management System is a sophisticated technological solution that revolutionizes the way educational institutions manage student information. By providing a seamless and integrated platform for admissions, updates, data display, exports, and secure deletions, an SMS contributes significantly to the efficient operation of educational institutions and the overall success of students throughout their academic journey.

II. Features of a Student Management System

Admission Module:

- The Admission Module streamlines the enrollment process, capturing crucial details during student admission. It efficiently manages personal information, contact details, and academic history. This module ensures a seamless onboarding experience for new students, facilitating accurate record creation for administrators and setting the foundation for a successful academic journey.

Updation Module:

- The Updation Module empowers authorized users to dynamically update student records. It facilitates modifications to personal details, academic progress, and extracurricular activities. This module ensures that student information remains current and accurate, supporting educators and administrators in making well-informed decisions based on the latest data.

Deletion Module:

- The Deletion Module provides a secure mechanism for removing student records when necessary. It includes confirmation prompts and safeguards against accidental or unauthorized deletions. This module, implemented with caution, ensures data integrity and compliance with privacy regulations, preventing unintended data loss and maintaining the overall security of the student management system.

Show Data Module:

- The Show Data Module is a user-friendly interface that allows authorized personnel to view and access student information effortlessly. It provides a comprehensive display of student records, enabling educators, administrators, and staff to retrieve specific details, fostering efficient communication and decision-making within the educational institution.

Search Student Data Module:

- The Search Student Data Module enhances accessibility by offering robust search functionalities. Users can efficiently locate specific student records based on criteria such as names, IDs, or academic performance. This module expedites the retrieval of targeted information, empowering educators and administrators to address individual student needs promptly and effectively.

Export Data Module:

- The Export Data Module facilitates the extraction of student information for reporting and analysis purposes. It provides a structured format compatible with external tools and software, supporting compliance requirements and enabling educational institutions to generate insightful reports on various aspects of student life, academic performance, and institutional effectiveness.

Exit Module:

- The Exit Module ensures a secure and organized closure of the student management system. It prompts users to save any unsaved changes, contributing to data integrity. This module allows for a graceful shutdown, providing a positive user experience and ensuring that the system concludes its operations securely and efficiently.
- A well-implemented Student Management System is crucial for educational institutions to efficiently manage administrative tasks, ensure data accuracy, and enhance communication among students, faculty, and administrators. It contributes significantly to the overall effectiveness of academic processes and supports informed decision-making.

III. Admitistrator's Role in a Student Management System.

- **System Configuration and Setup-:**

- **Configuration**- Administrators are responsible for configuring the SMS based on the institution's requirements. This includes setting up user roles, access permissions, and system preferences.
- **Setup**-They ensure that the SMS is properly set up and integrated with other systems, such as databases, authentication systems, and communication tools.

- ## User Management

- **Creation and Maintenance-** Administrators create and manage user accounts for faculty, staff, students, and other stakeholders. They oversee account maintenance, including password resets and access management.
- **User Roles**- Assigning appropriate roles and permissions to users is crucial for maintaining security and ensuring that individuals have access to the necessary functionalities.

- ## Data Management

- **Data Entry and Validation**- Administrators may be responsible for entering and validating data in the system, ensuring the accuracy and integrity of student records.
- **Data Security**- They implement and enforce data security measures to protect sensitive information, including the implementation of access controls and encryption.

- ## System Maintenance and Upgrades:

- **Updates and Patches**- Administrators oversee the installation of system updates, patches, and upgrades to ensure that the SMS remains current and secure.
- **Troubleshooting**- In the event of technical issues, administrators troubleshoot and resolve problems promptly to minimize disruptions to the system.

- ## Training and Support

- User Training- Administrators provide training sessions to users, including faculty, staff, and students, to ensure they are familiar with the features and functionalities of the SMS.
- User Support- They offer ongoing support to users, addressing inquiries, issues, and providing guidance on system usage.

- ## Report Generation and Analysis

- **Custom Reports**- Administrators often have the responsibility of generating custom reports for stakeholders, including academic performance reports, attendance summaries, and other relevant analytics.
- **Data Analysis**- They use the SMS to analyze data trends, identify areas for improvement, and provide insights to aid decision-making.

- **Communication and Collaboration**

- **Communication**- Administrators facilitate communication within the institution by utilizing communication features within the SMS. This includes announcements, notifications, and messaging functionalities.
- **Collaboration**- They promote collaboration among different departments and stakeholders by leveraging collaborative features of the SMS.

- **Policy Implementation**

- **Policy Enforcement**-Administrators ensure that institutional policies related to data privacy, user conduct, and system usage are enforced within the SMS.
- **Compliance**- They stay informed about relevant regulations and standards to ensure that the SMS remains compliant with educational and data protection standards.
- System Evaluation and Improvement:
- Feedback Collection: Administrators gather feedback from users to identify areas of improvement in the SMS.
- Continuous Improvement: They work towards enhancing the system's functionality and usability, collaborating with vendors or developers as needed.
- In summary, administrators are pivotal in the effective implementation and utilization of a Student Management System. Their responsibilities encompass system configuration, user management, data oversight, system maintenance, training and support, report generation, communication facilitation, policy enforcement, and ongoing system improvement. Through their efforts, administrators contribute to the overall efficiency and success of the educational institution's operations.

IV. Advantages of a Student Management System

A Student Management System (SMS) offers numerous advantages to educational institutions, administrators, teachers, and students. Here are several key benefits .

Efficient Data Management

- SMS streamlines the storage and retrieval of student data, including personal information, academic records, attendance, and more. This efficiency reduces manual paperwork, minimizes errors, and enhances overall data accuracy.

Streamlined Admission Process

- The system facilitates a smooth and organized admission process, allowing institutions to capture essential information during enrollment. This ensures a seamless transition for new students and aids administrators in managing admissions effectively.

Real-time Updates

- The SMS enables real-time updates to student records, ensuring that information is current and accurate. This dynamic updating system provides educators and administrators with the most recent data, supporting informed decision-making.

Enhanced Communication

- By centralizing student information, the SMS promotes efficient communication among educators, administrators, and parents. It provides a platform for sharing academic progress, attendance records, and other relevant information, fostering collaboration and engagement.

Data Analysis and Reporting

- SMS facilitates data analysis, allowing institutions to generate insightful reports on various aspects of student performance, attendance trends, and overall institutional effectiveness. This data-driven approach supports evidence-based decision-making.

Improved Attendance Tracking

- The system automates attendance tracking, making it easier for teachers to monitor and record student attendance. This feature helps identify patterns, address potential issues, and ensures accurate reporting for parents and administrators.

Customized Access Levels

- SMS provides role-based access controls, allowing different users to access only the information relevant to their roles. This ensures data security and privacy compliance while enabling efficient collaboration among various stakeholders.

Time and Resource Efficiency

- Automation of administrative tasks, such as data entry, report generation, and communication, saves time and resources. This efficiency allows educational institutions to focus on providing quality education and support services.

Parental Involvement

- SMS often includes features that facilitate parent-teacher communication. Parents can access their child's academic progress, attendance records, and important announcements, fostering increased parental involvement in the student's education.

Scalability

- A well-designed SMS is scalable, accommodating the evolving needs of educational institutions. Whether an institution grows in size or introduces new programs, the system can adapt to handle increased data and user requirements.

Compliance and Accountability

- SMS helps institutions maintain compliance with regulatory requirements by ensuring accurate record-keeping and reporting. The system also promotes accountability by logging user activities, providing an audit trail for data-related actions.

Implementing a robust Student Management System not only addresses current administrative challenges but also positions educational institutions to adapt and thrive in an increasingly digital and data-driven educational landscape.

- Explain how a Student Management System can help educational institutions save time and money.

Implementing a Student Management System (SMS) in educational institutions can lead to significant time and cost savings. Here are several ways in which an SMS contributes to efficiency and resource optimization:

Automation of Administrative Processes

Time Savings- Manual administrative tasks, such as student registration, data entry, and record-keeping, can be time-consuming. An SMS automates these processes, reducing the need for manual intervention and saving time.

Cost Savings- Automation reduces the staffing requirements for routine administrative tasks, leading to potential cost savings in terms of personnel expenses.

Streamlined Registration and Enrollment

Time Savings- With online registration and enrollment features, students can register for courses and programs more efficiently. This eliminates the need for manual paperwork and reduces the time spent on managing enrollment processes.

Cost Savings-The reduction in paperwork and manual processing translates to savings in printing, storage, and administrative overhead costs.

Improved Data Accuracy

Time Savings-Automated data entry and validation processes in an SMS reduce the likelihood of errors. This minimizes the time spent on rectifying mistakes and ensures accurate and up-to-date student records.

Cost Savings-Avoiding errors in data entry prevents the need for corrections and associated costs. It also contributes to better decision-making based on reliable data.

4. Enhanced Communication and Collaboration-

Time Savings Communication between students, faculty, and administrators is streamlined through the SMS. Quick and effective communication reduces the time spent on back-and-forth interactions.

Cost Savings- Improved communication minimizes misunderstandings and reduces the need for additional administrative efforts to address communication-related issues.

5. Resource Optimization-

Time Savings- Access to real-time data on student enrollment, course popularity, and other metrics allows institutions to optimize resource allocation, including faculty, classrooms, and materials.

Cost Savings- Efficient resource allocation prevents unnecessary expenditures and ensures that resources are utilized effectively.

Centralized Data Management

Time Savings- Having a centralized system for student data management eliminates the need to search through multiple files and databases. Information is easily accessible, saving time in locating and managing data.

Cost Savings- Centralized data management reduces the risk of data loss or duplication, saving resources that would be spent on data recovery or cleanup efforts.

Enhanced Reporting and Decision-Making

Time Savings- Automated reporting features in an SMS provide quick access to key performance indicators and analytics, saving time on manual data compilation and analysis.

Cost Savings- Informed decision-making based on accurate and timely data can lead to cost-effective strategies and resource allocation.

A well-implemented Student Management System contributes to time and cost savings by automating processes, improving data accuracy, optimizing resource allocation, and enhancing communication and collaboration within educational institutions. These efficiencies not only improve the overall functioning of the institution but also contribute to financial savings over the long term.

V. Conclusion

- In summary, a Student Management System with Oracle database integration provides a robust platform for managing student information throughout their academic journey. These features collectively contribute to the system's efficiency, accuracy, and usability, facilitating the smooth operation of educational institutions.
- Reiterate the importance of a Student Management System in educational institutions.

I hope this helps! Let me know if you have any other questions or if there's anything else I can help you with.

<u>**Student Management System Source Code**</u>

```
from tkinter import*
from PIL import ImageTk
from tkinter import messagebox
from tkinter import ttk,filedialog
import mysql.connector
import pandas

root=Tk()
root.title("Adminstrator Login of Student Management System")
root.geometry("1360x750+0+0")
root.resizable(0,0)

# Adminstrator Variable
Name=StringVar()
Email=StringVar()
Gender=StringVar()
Phone=StringVar()
Username=StringVar()
Password=StringVar()
# Login Panel
UsernameN=StringVar()
PasswordN=StringVar()

# Student Variable
SregisterId=StringVar()
Sname=StringVar()
Sroll=StringVar()
Ssubject=StringVar()
Semail=StringVar()
Sdob=StringVar()
Sstate=StringVar()
Sdistrict=StringVar()
```

```
Sgender=StringVar()

Scaste=StringVar()

Snationality=StringVar()

Snumber=StringVar()


# UserWindow Function

def UserWindow():

    def exportData():

        url=filedialog.asksaveasfilename(defaultextension=".csv")

        lst=[]

        index=StudentTable.get_children()

        for data in index:

            ct=StudentTable.item(data)

            datalist=ct["values"]

            lst.append(datalist)

         Table=pandas.DataFrame(lst,columns=["Registration_ID","Name","Roll","Subject","Email","DOB","State","Distric
t","Gender","Caste","Nationality","Mobile_No"])

        Table.to_csv(url,index=False)

        messagebox.showinfo("Exported","Data Change into Spredsheet and Saved Successfully")


    def updatePanel():

        UpdatePanel=Toplevel(userWindow)

        UpdatePanel.title("Update Data")

        UpdatePanel.geometry("600x540+120+10")

        UpdatePanel.resizable(0,0)


        def updateData():

            try:

                conn=mysql.connector.connect(user="root",password="Mysql@123",host="localhost",port=3306,database=
"StudentManagement")

            except:

                messagebox.showwarning("Student Database","Database Not Found")

            else:
```

```
        cur=conn.cursor()
                                                                                    sql="UPDATE STUDENT SET
SName=%s,SRoll=%s,SSubject=%s,SEmail=%s,SDOB=%s,SState=%s,SDistrict=%s,SGender=%s,SCaste=%s,SNationality=
%s,SNumber=%s WHERE Registation_ID=%s"
         cur.execute(sql,(SUName.get(),SURoll.get(),SUSubject.get(),SUEmail.get(),SUDob.get(),SUState.get(),SUDistri
ct.get(),SUGenderBox.get(),SUCasteBox.get(),SUNationalityBox.get(),SUNumber.get(),SURegisterId.get()))
        conn.commit()
        showData()
        conn.close()
        messagebox.showinfo("Updated",f"Data has been Updated\n Registation_ID={SURegisterId.get()}")

    UpdateFrame=Frame(UpdatePanel)
    UpdateFrame.pack()
    SRegisterId=Label(UpdateFrame,text="Registration Id:",font=("Arial",15,"bold"))
    SRegisterId.grid(row=0,column=0,padx=5,pady=5,sticky=W)
    SURegisterId=Entry(UpdateFrame,font=("Arial",15,"bold"),width=25)
    SURegisterId.grid(row=0,column=1,padx=5,pady=5,sticky=W)
    SName=Label(UpdateFrame,text="Student Name:",font=("Arial",15,"bold"))
    SName.grid(row=1,column=0,padx=5,pady=5,sticky=W)
    SUName=Entry(UpdateFrame,font=("Arial",15,"bold"),width=25)
    SUName.grid(row=1,column=1,padx=5,pady=5,sticky=W)
    SRoll=Label(UpdateFrame,text="Roll:",font=("Arial",15,"bold"))
    SRoll.grid(row=2,column=0,padx=5,pady=5,sticky=W)
    SURoll=Entry(UpdateFrame,font=("Arial",15,"bold"),width=25)
    SURoll.grid(row=2,column=1,padx=5,pady=5,sticky=W)
    SSubject=Label(UpdateFrame,text="Subject:",font=("Arial",15,"bold"))
    SSubject.grid(row=3,column=0,padx=5,pady=5,sticky=W)
    SUSubject=Entry(UpdateFrame,font=("Arial",15,"bold"),width=25)
    SUSubject.grid(row=3,column=1,padx=5,pady=5,sticky=W)
    SEmail=Label(UpdateFrame,text="Student Email:",font=("Arial",15,"bold"))
    SEmail.grid(row=4,column=0,padx=5,pady=5,sticky=W)
    SUEmail=Entry(UpdateFrame,font=("Arial",15,"bold"),width=25)
    SUEmail.grid(row=4,column=1,padx=5,pady=5,sticky=W)
    SDob=Label(UpdateFrame,text="Dob:",font=("Arial",15,"bold"))
```

```
SDob.grid(row=5,column=0,padx=5,pady=5,sticky=W)

SUDob=Entry(UpdateFrame,font=("Arial",15,"bold"),width=25)

SUDob.grid(row=5,column=1,padx=5,pady=5,sticky=W)

SState=Label(UpdateFrame,text="State:",font=("Arial",15,"bold"))

SState.grid(row=6,column=0,padx=5,pady=5,sticky=W)

SUState=Entry(UpdateFrame,font=("Arial",15,"bold"),width=25)

SUState.grid(row=6,column=1,padx=5,pady=5,sticky=W)

SDistrict=Label(UpdateFrame,text="District:",font=("Arial",15,"bold"))

SDistrict.grid(row=7,column=0,padx=5,pady=5,sticky=W)

SUDistrict=Entry(UpdateFrame,font=("Arial",15,"bold"),width=25)

SUDistrict.grid(row=7,column=1,padx=5,pady=5,sticky=W)

SGender=Label(UpdateFrame,text="Gender:",font=("Arial",15,"bold"))

SGender.grid(row=8,column=0,sticky=W)

SUGenderBox=ttk.Combobox(UpdateFrame,font=("Arial",15,"bold"),state="readonly",width=8)

SUGenderBox["value"]=("Male","Female","Transgender")

SUGenderBox.current(0)

SUGenderBox.grid(row=8,column=1,padx=5,pady=5,sticky=W)

SCaste=Label(UpdateFrame,text="Caste:",font=("Arial",15,"bold"))

SCaste.grid(row=9,column=0,padx=5,pady=5,sticky=W)

SUCasteBox=ttk.Combobox(UpdateFrame,font=("Arial",12,"bold"),state="readonly",width=8)

SUCasteBox["value"]=("General","BC-1","BC-2","EBC","SC","ST")

SUCasteBox.current(0)

SUCasteBox.grid(row=9,column=1,padx=5,pady=5,sticky=W)

SNationality=Label(UpdateFrame,text="Nationality:",font=("Arial",15,"bold"))

SNationality.grid(row=10,column=0,padx=5,pady=5,sticky=W)

SUNationalityBox=ttk.Combobox(UpdateFrame,font=("Arial",12,"bold"),state="readonly",width=8)

SUNationalityBox["value"]=("India","Nepal","China","USA","Brazil","Russia","Israel")

SUNationalityBox.current(0)

SUNationalityBox.grid(row=10,column=1,padx=5,pady=5,sticky=W)

SNumber=Label(UpdateFrame,text="Mobile No:",font=("Arial",15,"bold"))

SNumber.grid(row=11,column=0,padx=5,pady=5,sticky=W)

SUNumber=Entry(UpdateFrame,font=("Arial",15,"bold"),width=25)
```

```
    SUNumber.grid(row=11,column=1,padx=5,pady=5,sticky=W)


    SUpdate=Button(UpdateFrame,text="Update Now",font=("Arial",18,"bold"),bd=5,command=updateData)
    SUpdate.grid(row=12,column=1,columnspan=2,padx=5,pady=5,sticky=W)


    index=StudentTable.focus()
    Ct=StudentTable.item(index)
    data=Ct["values"]
    SURegisterId.insert(0,data[0])
    SUName.insert(0,data[1])
    SURoll.insert(0,data[2])
    SUSubject.insert(0,data[3])
    SUEmail.insert(0,data[4])
    SUDob.insert(0,data[5])
    SUState.insert(0,data[6])
    SUDistrict.insert(0,data[7])
    SUGenderBox.insert(0,data[8])
    SUCasteBox.insert(0,data[9])
    SUNationalityBox.insert(0,data[10])
    SUNumber.insert(0,data[11])


    UpdatePanel.mainloop()
  def deleteData():
    try:
      conn=mysql.connector.connect(user="root",password="Mysql@123",host="localhost",port=3306,database="S
tudentManagement")
    except:
      messagebox.showwarning("Student Database","Database Not Found")
    else:
      cur=conn.cursor()
      index=StudentTable.focus()
      Ct=StudentTable.item(index)
      regID=Ct["values"][0]
```

```
            sql="DELETE FROM STUDENT WHERE Registation_ID='{}' ".format(regID)
            cur.execute(sql)
            conn.commit()
            showData()
            conn.close()
            messagebox.showinfo("Deleted",f"Data has been deleted\n Registation_ID={regID}")


    def searchPanel():
        SearchPanel=Toplevel(userWindow)
        SearchPanel.title("Search Student")
        SearchPanel.geometry("400x300+300+10")


        # Searching Data and fetch to Display
        def searchData():
            try:
                conn=mysql.connector.connect(user="root",password="Mysql@123",host="localhost",port=3306,database=
"StudentManagement")
            except:
                messagebox.showwarning("Student Database","Database Not Found")
            else:
                cur=conn.cursor()
                    sql="SELECT *FROM STUDENT WHERE SRoll=%s or Registation_ID=%s or SNumber=%s or SEmail=%s or
SName=%s"
                cur.execute(sql,(Rollen.get(),Registration_Iden.get(),Mobile_noen.get(),Email_Iden.get(),Nameen.get()))
                fetchData=cur.fetchall()
                StudentTable.delete(*StudentTable.get_children())
                for Data in fetchData:
                    StudentTable.insert("",END,values=Data)


        Roll=Label(SearchPanel,text="Roll",font=("Arial",12,"bold"))
        Roll.grid(row=0,column=0,padx=5,pady=5,sticky=W)
        Rollen=Entry(SearchPanel,font=("Arial",15),bd=5)
        Rollen.grid(row=0,column=1,padx=5,pady=5,sticky=W)
        Registration_Id=Label(SearchPanel,text="Registration ID",font=("Arial",12,"bold"))
        Registration_Id.grid(row=1,column=0,padx=5,pady=5,sticky=W)
        Registration_Iden=Entry(SearchPanel,font=("Arial",15),bd=5)
        Registration_Iden.grid(row=1,column=1,padx=5,pady=5,sticky=W)
        Mobile_no=Label(SearchPanel,text="Mobile No",font=("Arial",12,"bold"))
        Mobile_no.grid(row=2,column=0,padx=5,pady=5,sticky=W)
        Mobile_noen=Entry(SearchPanel,font=("Arial",15),bd=5)
        Mobile_noen.grid(row=2,column=1,padx=5,pady=5,sticky=W)
        Email_Id=Label(SearchPanel,text="Email ID",font=("Arial",12,"bold"))
        Email_Id.grid(row=3,column=0,padx=5,pady=5,sticky=W)
        Email_Iden=Entry(SearchPanel,font=("Arial",15),bd=5)
        Email_Iden.grid(row=3,column=1,padx=5,pady=5,sticky=W)
        Name=Label(SearchPanel,text="Name",font=("Arial",12,"bold"))
        Name.grid(row=4,column=0,padx=5,pady=5,sticky=W)
        Nameen=Entry(SearchPanel,font=("Arial",15),bd=5)
        Nameen.grid(row=4,column=1,padx=5,pady=5,sticky=W)

        SearchBtn=Button(SearchPanel,text="Search",font=("Times new Roman",18,"bold"),bd=5,bg="yellow",fg="gray20",command=searchData)
        SearchBtn.grid(row=5,column=1,columnspan=2,pady=5)

        SearchPanel.mainloop()

    def showData():
        try:
            conn=mysql.connector.connect(user="root",password="Mysql@123",host="localhost",port=3306,database="StudentManagement")
        except:
            messagebox.showwarning("Student Database","Database Not Found")
        else:
            cur=conn.cursor()
            sql="SELECT *FROM STUDENT"
            cur.execute(sql)
```

```
        fetchData=cur.fetchall()
        StudentTable.delete(*StudentTable.get_children())
        for TableData in fetchData:
            Data=list(TableData)
            StudentTable.insert("",END,values=Data)


def Ssubmit():
    if(SregisterId.get()=="" or Sname.get()=="" or Sroll.get()=="" or Ssubject.get()=="" or
       Semail.get()=="" or Sdob.get()=="" or Sstate.get()=="" or Sdistrict.get()=="" or
       Sgender.get()=="" or Scaste.get()=="" or Snationality.get()=="" or Snumber.get()==""):
        messagebox.showerror("Student records Blank","All fields are mendatory")
    else:
        try:
            conn=mysql.connector.connect(user="root",password="Mysql@123",host="localhost",port=3306,database=
"StudentManagement")
            cur=conn.cursor()
            cur.execute("insert into student values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",
                    (SregisterId.get(), Sname.get(), Sroll.get(), Ssubject.get(),
                    Semail.get(), Sdob.get(), Sstate.get(), Sdistrict.get(),
                    Sgender.get(), Scaste.get(), Snationality.get(), Snumber.get()))
        except:
            messagebox.showwarning("Student Database","Duplicate Entry")
        else:
            conn.commit()
            showData()
            conn.close()
            messagebox.showinfo("Successful","Data Inserted")
        finally:
            SregisterId.set("")
            Sname.set("")
            Sroll.set("")
            Ssubject.set("")
            Semail.set("")
        Sdob.set("")
        Sstate.set("")
        Sdistrict.set("")
        Sgender.set("")
        Scaste.set("")
        Snationality.set("")
        Snumber.set("")


def AddmissionPanel():
    AddmissionPanel=Toplevel(userWindow)
    AddmissionPanel.title("Adminstrator Login of Student Management System")
    AddmissionPanel.geometry("600x700+120+10")
    AddmissionPanel.resizable(0,0)

    addmissionFrame=Frame(AddmissionPanel)
    addmissionFrame.pack()

    SRegisterId=Label(addmissionFrame,text="Registration Id:",font=("Arial",15,"bold"))
    SRegisterId.grid(row=0,column=0,padx=5,pady=5,sticky=W)
    SRegisterId=Entry(addmissionFrame,font=("Arial",15,"bold"),width=25,textvariable=SregisterId)
    SRegisterId.grid(row=0,column=1,padx=5,pady=5,sticky=W)
    SName=Label(addmissionFrame,text="Student Name:",font=("Arial",15,"bold"))
    SName.grid(row=1,column=0,padx=5,pady=5,sticky=W)
    SName=Entry(addmissionFrame,font=("Arial",15,"bold"),width=25,textvariable=Sname)
    SName.grid(row=1,column=1,padx=5,pady=5,sticky=W)
    SRoll=Label(addmissionFrame,text="Roll:",font=("Arial",15,"bold"))
    SRoll.grid(row=2,column=0,padx=5,pady=5,sticky=W)
    SRoll=Entry(addmissionFrame,font=("Arial",15,"bold"),width=25,textvariable=Sroll)
    SRoll.grid(row=2,column=1,padx=5,pady=5,sticky=W)
    SSubject=Label(addmissionFrame,text="Subject:",font=("Arial",15,"bold"))
    SSubject.grid(row=3,column=0,padx=5,pady=5,sticky=W)
    SSubject=Entry(addmissionFrame,font=("Arial",15,"bold"),width=25,textvariable=Ssubject)

    SSubject.grid(row=3,column=1,padx=5,pady=5,sticky=W)

    SEmail=Label(addmissionFrame,text="Student Email:",font=("Arial",15,"bold"))

    SEmail.grid(row=4,column=0,padx=5,pady=5,sticky=W)

    SEmailEn=Entry(addmissionFrame,font=("Arial",15,"bold"),width=25,textvariable=Semail)

    SEmailEn.grid(row=4,column=1,padx=5,pady=5,sticky=W)

    SDob=Label(addmissionFrame,text="Dob:",font=("Arial",15,"bold"))

    SDob.grid(row=5,column=0,padx=5,pady=5,sticky=W)

    SDobEn=Entry(addmissionFrame,font=("Arial",15,"bold"),width=25,textvariable=Sdob)

    SDobEn.grid(row=5,column=1,padx=5,pady=5,sticky=W)

    SState=Label(addmissionFrame,text="State:",font=("Arial",15,"bold"))

    SState.grid(row=6,column=0,padx=5,pady=5,sticky=W)

    SStateEn=Entry(addmissionFrame,font=("Arial",15,"bold"),width=25,textvariable=Sstate)

    SStateEn.grid(row=6,column=1,padx=5,pady=5,sticky=W)

    SDistrict=Label(addmissionFrame,text="District:",font=("Arial",15,"bold"))

    SDistrict.grid(row=7,column=0,padx=5,pady=5,sticky=W)

    SDistrictEn=Entry(addmissionFrame,font=("Arial",15,"bold"),width=25,textvariable=Sdistrict)

    SDistrictEn.grid(row=7,column=1,padx=5,pady=5,sticky=W)

    SGender=Label(addmissionFrame,text="Gender:",font=("Arial",15,"bold"))

    SGender.grid(row=8,column=0,sticky=W)

    SGenderBox=ttk.Combobox(addmissionFrame,font=("Arial",15,"bold"),state="readonly",width=8,textvariable=Sg
ender)

    SGenderBox["value"]=("Male","Female","Transgender")

    SGenderBox.current(0)

    SGenderBox.grid(row=8,column=1,padx=5,pady=5,sticky=W)

    SCaste=Label(addmissionFrame,text="Caste:",font=("Arial",15,"bold"))

    SCaste.grid(row=9,column=0,padx=5,pady=5,sticky=W)

    SCasteBox=ttk.Combobox(addmissionFrame,font=("Arial",12,"bold"),state="readonly",width=8,textvariable=Scas
te)

    SCasteBox["value"]=("General","BC-1","BC-2","EBC","SC","ST")

    SCasteBox.current(0)

    SCasteBox.grid(row=9,column=1,padx=5,pady=5,sticky=W)

    SNationality=Label(addmissionFrame,text="Nationality:",font=("Arial",15,"bold"))
```

```
        SNationality.grid(row=10,column=0,padx=5,pady=5,sticky=W)

        SNationalityBox=ttk.Combobox(addmissionFrame,font=("Arial",12,"bold"),state="readonly",width=8,textvariable
=Snationality)

        SNationalityBox["value"]=("India","Nepal","China","USA","Brazil","Russia","Israel")

        SNationalityBox.current(0)

        SNationalityBox.grid(row=10,column=1,padx=5,pady=5,sticky=W)

        SNumber=Label(addmissionFrame,text="Mobile No:",font=("Arial",15,"bold"))

        SNumber.grid(row=11,column=0,padx=5,pady=5,sticky=W)

        SNumberEn=Entry(addmissionFrame,font=("Arial",15,"bold"),width=25,textvariable=Snumber)

        SNumberEn.grid(row=11,column=1,padx=5,pady=5,sticky=W)


        SSubmit=Button(addmissionFrame,text="Submit Now!",font=("Arial",20,"bold"),command=Ssubmit)

        SSubmit.grid(row=12,column=1,columnspan=2,padx=5,pady=5,sticky=W)


        AddmissionPanel.mainloop()


    def ConnectDB():

        AddmissionST.config(state=NORMAL)

        UpdationST.config(state=NORMAL)

        SearchST.config(state=NORMAL)

        Export.config(state=NORMAL)

        DeleteST.config(state=NORMAL)

        ShowST.config(state=NORMAL)


    def ExitAdmin():

        op=messagebox.askyesno("Close Window","Are you sure to Exit")

        if (op>0):

            messagebox.showinfo("Bye!","Thank You for Visit")

            root.destroy()

        else:

            messagebox.showinfo("Close","Your current Window Close ")

            userWindow.destroy()
    userWindow=Toplevel(root)
    userWindow.title("Adminstrator Login of Student Management System")
    userWindow.geometry("1360x750+0+0")
    userWindow.resizable(0,0)
    # Background Image
    Background=ImageTk.PhotoImage(file="project\\StudentManagement\\b1.jpg")
    bg=Label(userWindow,image=Background)
    bg.place(x=0,y=0)
    HeadingIm=ImageTk.PhotoImage(file="project\\StudentManagement\\bg1.jpg")
    Heading=Label(userWindow,image=HeadingIm,font=("Times New Roman",30,"bold"),width=800,height=80)
    Heading.pack(fill=X)
    HeadingT=Label(userWindow,text="Tarkeshwar Prasad Varma College ",font=("Times New Roman",30,"bold"),fg="gold",bg="blue")
    HeadingT.place(x=360,y=4)


    DBC=Button(userWindow,text="Connect DataBase",font=("Arial",15,"bold"),bd=5,bg="gray20",fg="white",activebackground="Yellow",command=ConnectDB)
    DBC.place(x=1120,y=25)


    Option=Frame(userWindow,bd=5,bg="gold")
    Option.place(x=30,y=100)


    AddmissionST=Button(Option,text="Addmission",font=("Arial",15,"bold"),bg="blue",fg="white",bd=5,width=18,state=DISABLED,cursor="hand2",command=AddmissionPanel)
    AddmissionST.grid(row=0,column=0,padx=20,pady=20,sticky=W)
    UpdationST=Button(Option,text="Updation",font=("Arial",15,"bold"),bg="blue",fg="white",bd=5,width=18,state=DISABLED,cursor="hand2",command=updatePanel)
    UpdationST.grid(row=1,column=0,padx=20,pady=20,sticky=W)
    SearchST=Button(Option,text="Search",font=("Arial",15,"bold"),bg="blue",fg="white",bd=5,width=18,state=DISABLED,cursor="hand2",command=searchPanel)
    SearchST.grid(row=2,column=0,padx=20,pady=20,sticky=W)
    ShowST=Button(Option,text="Show Data",font=("Arial",15,"bold"),bg="blue",fg="white",bd=5,width=18,state=DISABLED,cursor="hand2",command=showData)
    ShowST.grid(row=3,column=0,padx=20,pady=20,sticky=W)
    DeleteST=Button(Option,text="Delete",font=("Arial",15,"bold"),bg="blue",fg="white",bd=5,width=18,state=DISABLED,cursor="hand2",command=deleteData)

    DeleteST.grid(row=4,column=0,padx=20,pady=20,sticky=W)

    Exit=Button(Option,text="Exit",font=("Arial",15,"bold"),bg="blue",fg="white",bd=5,width=18,cursor="hand2",command=ExitAdmin)

    Exit.grid(row=5,column=0,padx=20,pady=20,sticky=W)

    Export=Button(Option,text="Export",font=("Arial",15,"bold"),bg="blue",fg="white",bd=5,width=18,state=DISABLED,cursor="hand2",command=exportData)

    Export.grid(row=6,column=0,padx=20,pady=20,sticky=W)


    # Display Frame

    OutputFrame=Frame(userWindow,bd=5,bg="gold")

    OutputFrame.place(x=300,y=100,width=1000,height=630)


    OutputScrollX=Scrollbar(OutputFrame,orient=VERTICAL)

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

    OutputScrollY=Scrollbar(OutputFrame,orient=HORIZONTAL)

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


    StudentTable=ttk.Treeview(OutputFrame,yscrollcommand=OutputScrollX.set,xscrollcommand=OutputScrollY.set)

    StudentTable["columns"]=("Reg","Name","Rol","Sub","Gmail","Dob","State","Dist","Gender","Cast","Nation","Num")

    style=ttk.Style()

    style.configure("Treeview",font=("Arial",10),rowheight=30)

    style.configure("Treeview.Heading",font=("Arial",12))


    # Table width

    StudentTable.column("Reg",width=120,anchor=CENTER)

    StudentTable.column("Name",width=180,anchor=CENTER)

    StudentTable.column("Rol",width=100,anchor=CENTER)

    StudentTable.column("Sub",width=180,anchor=CENTER)

    StudentTable.column("Gmail",width=180,anchor=CENTER)

    StudentTable.column("Dob",width=120,anchor=CENTER)

    StudentTable.column("State",width=150,anchor=CENTER)
    StudentTable.column("Dist",width=180,anchor=CENTER)

    StudentTable.column("Gender",width=80,anchor=CENTER)

    StudentTable.column("Cast",width=80,anchor=CENTER)

    StudentTable.column("Nation",width=180,anchor=CENTER)

    StudentTable.column("Num",width=100,anchor=CENTER)

    # Table Heading

    StudentTable.heading("Reg",text="Registration_ID")

    StudentTable.heading("Name",text="Name")

    StudentTable.heading("Rol",text="Roll")

    StudentTable.heading("Sub",text="Subject")

    StudentTable.heading("Gmail",text="Email_Id")

    StudentTable.heading("Dob",text="Date of Birth")

    StudentTable.heading("State",text="State")

    StudentTable.heading("Dist",text="District")

    StudentTable.heading("Gender",text="Gender")

    StudentTable.heading("Cast",text="Caste")

    StudentTable.heading("Nation",text="Nationality")

    StudentTable.heading("Num",text="Mobile_No")


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

    StudentTable["show"]="headings" #It removes unnecessary column which comes on 0th index

    OutputScrollX.config(command=StudentTable.yview)

    OutputScrollY.config(command=StudentTable.xview)


    userWindow.mainloop()

# Login Function

def AdminLogin():

    if(usernameEn.get()=="" or passwordEn.get()==""):

        messagebox.showwarning("Warning","Username or Password can't Empty")

    else:

        try:

            conn=mysql.connector.connect(
                user="root",
                password="Mysql@123",
                host="localhost",
                port=3306,
                database="studentmanagement")
        except:
            messagebox.showerror("Database not Connected")
        else:
            cur=conn.cursor()
            sql="SELECT *FROM registration"
            cur.execute(sql)
            lst=cur.fetchall()
            for i in range(0,len(lst)):
                for j in range(0,6):
                    if(UsernameN.get()==lst[i][j]):
                        for k in range(0,len(lst)):
                            for l in range(0,6):
                                if(PasswordN.get()==lst[k][l]):
                                    messagebox.showinfo("Successful","Congrats Welcome to TPVC")
                                    UserWindow()
                                    # print(lst[i][j])
                                    # print(lst[k][l])


def AdminSubmit():
    if(Name.get()=="" or Email.get()=="" or Phone.get()=="" or Username.get()=="" or Password.get()==""):
        messagebox.showwarning("Empty","All fields are mendatory")
    else:
        try:
            conn=mysql.connector.connect(
                    user="root",
                    password="Mysql@123",
                    host="localhost",
                port=3306,
                database="studentmanagement")
        except:
            messagebox.showerror("Database not Connected")
        else:
            cur=conn.cursor()
            cur.execute("insert into registration values(%s,%s,%s,%s,%s,%s)",(Name.get(),Email.get(),Gender.get(),Phone.get(),Username.get(),Password.get()))
            conn.commit()
            conn.close()
            messagebox.showinfo("Successful","Data Inserted")
        finally:
            Name.set("")
            Email.set("")
            Gender.set("")
            Phone.set("")
            Username.set("")
            Password.set("")

def AdminRegister():
    register=Toplevel(root)
    register.title("Registration")
    register.geometry("400x380+100+20")
    register.resizable(0,0)

    registerFrame=LabelFrame(register,text="User Registration",font=("Arial",15,"bold"))
    registerFrame.place(x=50,y=10)

    NameLabel=Label(registerFrame,text="Name",font=("Arial",12,"bold"))
    NameLabel.grid(row=0,column=0,sticky=W)
    NameEn=Entry(registerFrame,font=("Arial",12,"bold"),textvariable=Name)
    NameEn.grid(row=0,column=1,padx=10,pady=10)
    EmailLabel=Label(registerFrame,text="Email",font=("Arial",12,"bold"))
    EmailLabel.grid(row=1,column=0,sticky=W)
    EmailEn=Entry(registerFrame,font=("Arial",12,"bold"),textvariable=Email)
    EmailEn.grid(row=1,column=1,padx=10,pady=10)

    GenderLabel=Label(registerFrame,text="Gender",font=("Arial",12,"bold"))
    GenderLabel.grid(row=2,column=0,sticky=W)
    GenderBox=ttk.Combobox(registerFrame,text="Gender",font=("Arial",12,"bold"),state="readonly",width=8,textvari
able=Gender)
    GenderBox["value"]=("Male","Female","Transgender")
    GenderBox.current(0)
    GenderBox.grid(row=2,column=1,padx=10,pady=10,sticky=W)

    PhoneLabel=Label(registerFrame,text="Phone No",font=("Arial",12,"bold"))
    PhoneLabel.grid(row=3,column=0,sticky=W)
    PhoneEn=Entry(registerFrame,font=("Arial",12,"bold"),textvariable=Phone)
    PhoneEn.grid(row=3,column=1,padx=10,pady=10)

    UserLabel=Label(registerFrame,text="User Name",font=("Arial",12,"bold"))
    UserLabel.grid(row=4,column=0,sticky=W)
    UserEn=Entry(registerFrame,font=("Arial",12,"bold"),textvariable=Username)
    UserEn.grid(row=4,column=1,padx=10,pady=10)

    PasswordLabel=Label(registerFrame,text="Password",font=("Arial",12,"bold"))
    PasswordLabel.grid(row=5,column=0,sticky=W)
    PasswordEn=Entry(registerFrame,font=("Arial",12,"bold"),textvariable=Password)
    PasswordEn.grid(row=5,column=1,padx=10,pady=10)

    SubmitBtn=Button(registerFrame,text="Submit",font=("Arial",15,"bold"),bg="blue",fg="white",activebackground="
yellow",cursor="hand2",command=AdminSubmit)
    SubmitBtn.grid(row=6,column=0,columnspan=2,padx=10,pady=10)
    register.mainloop()
```

```
# Background Image

Background=ImageTk.PhotoImage(file="project\\StudentManagement\\bg1.jpg")

bg=Label(root,image=Background)

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


loginFrame=Frame(root,bg="gray50")

loginFrame.place(x=550,y=150)


logoImage=PhotoImage(file="project\\StudentManagement\\logo.png")

logoLabel=Label(loginFrame,image=logoImage)

logoLabel.grid(row=0,column=0,columnspan=2,padx=10,pady=10)


usernameImage=PhotoImage(file="project\\StudentManagement\\user.png")

usernameLabel=Label(loginFrame,text="Username",font=("Arial",12,"bold"),image=usernameImage,compound=LEFT
,bg="gray50")

usernameLabel.grid(row=1,column=0,padx=10,pady=10)

usernameEn=Entry(loginFrame,font=("Arial",15,"bold"),width=15,bd=5,textvariable=UsernameN)

usernameEn.grid(row=1,column=1,padx=10,pady=10)


passwordImage=PhotoImage(file="project\StudentManagement\password.png")

passwordLabel=Label(loginFrame,text="Password",font=("Arial",12,"bold"),image=passwordImage,compound=LEFT,b
g="gray50")

passwordLabel.grid(row=2,column=0,padx=10,pady=10)

passwordEn=Entry(loginFrame,font=("Arial",15,"bold"),width=15,bd=5,textvariable=PasswordN)

passwordEn.grid(row=2,column=1,padx=10,pady=10)


LoginButton=Button(loginFrame,font=("Arial",15,"bold"),text="Login",bd=5,activebackground="blue",activeforegrou
nd="white",width=8,cursor="hand2",command=AdminLogin)

LoginButton.grid(row=4,column=0,columnspan=2,padx=10,pady=10)


RegisterButton=Button(root,font=("Arial",15,"bold"),text="Register",width=10,cursor="hand2",activebackground="bl
ue",activeforeground="white",command=AdminRegister)

RegisterButton.place(x=1220,y=10)

root.mainloop()
```

POWERED BY TARKESHWAR PRASAD VARMA COLLEGE STUDENTS

TP VARMA COLLEGE

Narkatiyaganj

Assignment (Synopsis)

BCA sem- 6th

Group Project on

Student Management System

By Roll No:-

214072 - Amrita Kumari Suman

214073 - Abhinav Prakash

214074 - Abhishek Kumar

214077 - Nitish Ranjan

214078 - Rahul Kumar Gupta

214079 - Rohini Kumari