EduData

A COMPREHENSIVE STUDENT MANAGEMENT SYSTEM

Ashmit Srivastava
AMITY INTERNATIONAL SCHOOL | MAYUR VIHAR

Certificate

This is to certify that Ashmit Srivastava of class XII N, Amity International School, Mayur Vihar has successfully completed their project for Computer Science for the AISSCE as prescribed by the CBSE for academic year 2023-24.

Teacher in-charge

Acknowledgement

I would like to express my deep gratitude and sincere thanks to my Computer Science teacher for guiding and supporting me throughout this project.

Sincere thanks to my Computer Science lab in-charge who made the necessary arrangements and supported for the completion of this project.

I also would thank my parents and friends who have helped me with their valuable suggestions in this project.

Ashmit Srivastava

Contents

Introduction

Welcome to EduData: A Comprehensive Student Management System

As a student in the 12th standard at Amity International School, pursuing Computer Science as a subject under the Central Board of Secondary Education (CBSE), I am excited to present my project, EduData. This endeavour stems from the amalgamation of theoretical concepts learned in the classroom with hands-on practical application. The project aligns with the CBSE curriculum, providing an opportunity to explore the realms of software development, databases, and user interface design.

Project Overview

EduData is a student management system designed to streamline and enhance the management of student information within an educational institution. This software aims to address the challenges faced in organizing, retrieving, and manipulating student data efficiently. By leveraging the power of Python, SQL, and Tkinter, EduData offers a user-friendly Graphical User Interface (GUI) that simplifies the process of managing student records.

Objectives

The primary objectives of EduData include:

1. **User-Friendly Interface:**
   Develop a visually appealing and intuitive user interface to facilitate easy interaction with the software.
2. **Database Integration:**
   Implement a robust database system using MySQL to store and manage student information securely.
3. **Data Manipulation:**
   Provide functionalities for adding, searching, updating, and deleting student records, ensuring comprehensive data manipulation capabilities.
4. **Export Functionality:**
   Enable users to export student data to external files, enhancing the accessibility and portability of information.

Significance of EduData

EduData addresses the vital need for an organized and efficient system to manage student records within educational institutions. The project not only aligns with the CBSE curriculum but also serves as a practical application of the skills acquired during the course, fostering a deeper understanding of software development concepts.

This documentation provides insights into the development process, the technologies used, and the functionalities offered by EduData. I hope this project showcases the practical application of computer science concepts and serves as an asset for educational institutions seeking an effective solution for student data management.

Modules

1. **tkinter**

   tkinter is the standard GUI toolkit for Python. It provides the necessary tools to create graphical user interfaces.

2. **PIL (Pillow)**

   The PIL (Python Imaging Library) module is used for opening, manipulating, and saving image files. In the provided code, it is used for handling image files, such as logos and background images.

3. **time**

   The time module provides various time-related functions. In the code, it is used for displaying the current date and time.

4. **ttkthemes**

   The ttkthemes module provides themed Tkinter widgets. It is used for setting the theme of the Tkinter application in the provided code.

5. **ttk**

   The ttk module provides access to the Tk themed widget set. It includes advanced widgets like Button, Treeview, and others with a more modern look compared to the standard Tkinter widgets.

6. **messagebox**

   The messagebox module provides a simple way to create message boxes for displaying information, warnings, and errors.

7. **filedialog**

   The filedialog module is used to display the file dialog, allowing users to select files or directories.

8. **pymysql**

   The pymysql module is a MySQL database adapter for Python. It allows Python programs to communicate with a MySQL database.

9. **pandas**

   The pandas module is a powerful data manipulation and analysis library. In the provided code, it is used for creating and handling data frames, especially for exporting data to CSV.

10. **ImageTk**

    The ImageTk module is part of the PIL library and provides support for using images in Tkinter.

These modules collectively enable the creation of a GUI-based student management system with functionalities like connecting to a database, adding, searching, updating, and deleting student records, as well as exporting data.

Source Code

main.py

```
from tkinter import *
from tkinter import messagebox
from PIL import ImageTk

def login():
    if usernameEntry.get()=='' or passwordEntry.get()=='':
        messagebox.showerror('Error', 'Field cannot be empty.')
    elif usernameEntry.get()=='administrator' and
passwordEntry.get()=='admin':
        window.destroy()
        import sms
    else:
        messagebox.showerror('Error', 'Please enter valid credentials.')

# Window
window = Tk()
window.geometry('1054x720+243+50')
window.title('Login | Amity International School || EduData')
window.resizable(False, False)

# Window Logo
logo=PhotoImage(file='ais.png')
window.iconphoto(False, logo)

# Background Image
backgroundImage = ImageTk.PhotoImage(file='bg.jpg')
bgLabel = Label(window, image=backgroundImage)
bgLabel.place(x=0,y=0)

# Login Frame
loginFrame = Frame(window, bg='white')
loginFrame.place(x=350, y=200)

# Logo
logoImage = PhotoImage(file='profile.png')
logoLabel = Label(loginFrame, image=logoImage, bg='white')
logoLabel.grid(row=0, column=0, columnspan=2, pady=10)

# Username
usernameImage = PhotoImage(file='user.png')
usernameLabel = Label(loginFrame, image=usernameImage, text='Username',
compound=LEFT, font=('bookman old style', 15, 'bold'), bg='white')
usernameLabel.grid(row=1, column=0)
```

```
usernameEntry = Entry(loginFrame, font=('bookman old style', 15), bd=2)
usernameEntry.grid(row=1, column=1, padx=20, pady=10)

# Password
passwordImage = PhotoImage(file='padlock.png')
passwordLabel = Label(loginFrame, image=passwordImage, text='Password',
compound=LEFT, font=('bookman old style', 15, 'bold'), bg='white')
passwordLabel.grid(row=2, column=0)

passwordEntry = Entry(loginFrame, show='*', font=('bookman old style', 15),
bd=2)
passwordEntry.grid(row=2, column=1, padx=20, pady=10)

# Login
loginButton = Button(loginFrame, text='Login', font=('bookman old style', 14,
'bold'), width=15, fg='white', bg='cornflowerblue', activeforeground='white',
cursor='hand2', command=login)
loginButton.grid(row=3, column=0, columnspan=2, pady=10)

window.mainloop()
```

sms.py

```
from tkinter import *
import time
import ttkthemes
from tkinter import ttk, messagebox, filedialog
import pymysql
import pandas

# Functions
def clock():
    date = time.strftime('%d/%m/%Y')
    currtime = time.strftime('%H:%M:%S')
    datetimeLabel.config(text=f'   Date: {date}\nTime: {currtime}')
    datetimeLabel.after(1000, clock)

def connect_database():
    def connect():
        global mycursor,con
        try:
            con=pymysql.connect(host=hostEntry.get(), user=userEntry.get(),
password=passwordEntry.get())
            mycursor=con.cursor()
        except:
            messagebox.showerror('Error', 'Invalid Details',
parent=connectWindow)
            return
        try:
            query='create database studentmanagement'
            mycursor.execute(query)
            query='use studentmanagement'
            mycursor.execute(query)
            query='create table student(sno int not null, admno int, name
varchar(50), gender varchar(20), dob varchar(20), mobile varchar(10), email
varchar(50))'
            mycursor.execute(query)
        except:
            query='use studentmanagement'
            mycursor.execute(query)
        messagebox.showinfo('Sucess', 'Database Connected!',
parent=connectWindow)
        connectWindow.destroy()
        addstudentButton.config(state=NORMAL)
        searchstudentButton.config(state=NORMAL)
        updatestudentButton.config(state=NORMAL)
        showstudentButton.config(state=NORMAL)
        deletestudentButton.config(state=NORMAL)
        exportdataButton.config(state=NORMAL)
        exitButton.config(state=NORMAL)
```

```
    connectWindow=Toplevel()
    connectWindow.resizable(False, False)
    connectWindow.grab_set()
    connectWindow.geometry('470x250+500+230')
    connectWindow.title('Connect Database | EduData')
    connectWindow.iconphoto(False, logo)
    connectWindow.resizable(0,0)

    hostnameLabel = Label(connectWindow, text='Host Name', font=('bookman old
style', 18))
    hostnameLabel.grid(row=0, column=0, padx=20)

    hostEntry= Entry(connectWindow, font=('bookman old style', 15))
    hostEntry.grid(row=0, column=1, padx=40, pady=20)

    usernameLabel = Label(connectWindow, text='User Name', font=('bookman old
style', 18))
    usernameLabel.grid(row=1, column=0, padx=20)

    userEntry= Entry(connectWindow, font=('bookman old style', 15))
    userEntry.grid(row=1, column=1, padx=40, pady=20)

    passwordLabel = Label(connectWindow, text='Password', font=('bookman old
style', 18))
    passwordLabel.grid(row=2, column=0, padx=20)

    passwordEntry= Entry(connectWindow, show='*', font=('bookman old style',
15))
    passwordEntry.grid(row=2, column=1, padx=40, pady=20)

    connectButton = ttk.Button(connectWindow, text='CONNECT', command=connect)
    connectButton.grid(row=3, columnspan=2)

def add_student():
    def add_data():
        if snoEntry.get()=='' or admnoEntry.get()=='' or nameEntry.get()==''
or genderEntry.get()=='' or dobEntry.get()=='' or mobileEntry.get()=='' or
emailEntry.get()=='':
            messagebox.showerror('Error', 'All fields are required',
parent=addWindow)
        else:
            try:
                query='insert into student values(%s,%s,%s,%s,%s,%s,%s)'
                mycursor.execute(query, (snoEntry.get(), admnoEntry.get(),
nameEntry.get(), genderEntry.get(), dobEntry.get(), mobileEntry.get(),
emailEntry.get()))
                con.commit()
```

```
                result=messagebox.askyesno('Confirm', 'Data added
successfully. Do you want to clean the form?', parent=addWindow)
                if result:
                    snoEntry.delete(0,END)
                    admnoEntry.delete(0,END)
                    nameEntry.delete(0,END)
                    genderEntry.delete(0,END)
                    dobEntry.delete(0,END)
                    mobileEntry.delete(0,END)
                    emailEntry.delete(0,END)
                else:
                    pass
            except:
                messagebox.showerror('Error', 'Adm. No. cannot be repeated')
                return

            query='select *from student order by sno'
            mycursor.execute(query)
            fetched_data=mycursor.fetchall()
            studentTable.delete(*studentTable.get_children())
            for data in fetched_data:
                datalist=list(data)
                studentTable.insert('', END, values=datalist)

    addWindow = Toplevel()
    addWindow.title('Add Student | EduData')
    addWindow.iconphoto(False, logo)
    addWindow.resizable(False, False)
    addWindow.grab_set()
    # S. No.
    snoLabel=Label(addWindow, text='S. No.', font=('bookman old style', 18))
    snoLabel.grid(row=0, column=0, padx=30, pady=15, sticky=W)
    snoEntry=Entry(addWindow, font=('bookman old style', 15), width=24)
    snoEntry.grid(row=0, column=1, pady=15, padx=10)
    # Adm. No.
    admnoLabel=Label(addWindow, text='Adm. No.', font=('bookman old style',
18))
    admnoLabel.grid(row=1, column=0, padx=30, pady=15, sticky=W)
    admnoEntry=Entry(addWindow, font=('bookman old style', 15), width=24)
    admnoEntry.grid(row=1, column=1, pady=15, padx=10)
    # Name
    nameLabel=Label(addWindow, text='Name', font=('bookman old style', 18))
    nameLabel.grid(row=2, column=0, padx=30, pady=15, sticky=W)
    nameEntry=Entry(addWindow, font=('bookman old style', 15), width=24)
    nameEntry.grid(row=2, column=1, pady=15, padx=10)
    # Gender
    genderLabel=Label(addWindow, text='Gender', font=('bookman old style',
18))
```

```
    genderLabel.grid(row=3, column=0, padx=30, pady=15, sticky=W)
    genderEntry=Entry(addWindow, font=('bookman old style', 15), width=24)
    genderEntry.grid(row=3, column=1, pady=15, padx=10)
    # D.O.B.
    dobLabel=Label(addWindow, text='D.O.B.', font=('bookman old style', 18))
    dobLabel.grid(row=4, column=0, padx=30, pady=15, sticky=W)
    dobEntry=Entry(addWindow, font=('bookman old style', 15), width=24)
    dobEntry.grid(row=4, column=1, pady=15, padx=10)
    # Mobile No.
    mobileLabel=Label(addWindow, text='Mobile No.', font=('bookman old style',
18))
    mobileLabel.grid(row=5, column=0, padx=30, pady=15, sticky=W)
    mobileEntry=Entry(addWindow, font=('bookman old style', 15), width=24)
    mobileEntry.grid(row=5, column=1, pady=15, padx=10)
    # Email ID
    emailLabel=Label(addWindow, text='Email ID', font=('bookman old style',
18))
    emailLabel.grid(row=6, column=0, padx=30, pady=15, sticky=W)
    emailEntry=Entry(addWindow, font=('bookman old style', 15), width=24)
    emailEntry.grid(row=6, column=1, pady=15, padx=10)
    # Add Student Button
    add_studentButton=ttk.Button(addWindow, text='ADD STUDENT',
command=add_data)
    add_studentButton.grid(row=7, columnspan=2, pady=15)

def search_student():
    def search_data():
        query='select * from student where sno=%s or admno=%s or name=%s or
gender=%s or dob=%s or mobile=%s or email=%s order by sno'
        mycursor.execute(query, (snoEntry.get(), admnoEntry.get(),
nameEntry.get(), genderEntry.get(), dobEntry.get(), mobileEntry.get(),
emailEntry.get()))
        studentTable.delete(*studentTable.get_children())
        fetched_data=mycursor.fetchall()
        for data in fetched_data:
            studentTable.insert('', END, values=data)

    searchWindow = Toplevel()
    searchWindow.title('Search Student | EduData')
    searchWindow.iconphoto(False, logo)
    searchWindow.resizable(False, False)
    searchWindow.grab_set()
    # S. No.
    snoLabel=Label(searchWindow, text='S. No.', font=('bookman old style',
18))
    snoLabel.grid(row=0, column=0, padx=30, pady=15, sticky=W)
    snoEntry=Entry(searchWindow, font=('bookman old style', 15), width=24)
    snoEntry.grid(row=0, column=1, pady=15, padx=10)
```

```
    # Adm. No.
    admnoLabel=Label(searchWindow, text='Adm. No.', font=('bookman old style',
18))
    admnoLabel.grid(row=1, column=0, padx=30, pady=15, sticky=W)
    admnoEntry=Entry(searchWindow, font=('bookman old style', 15), width=24)
    admnoEntry.grid(row=1, column=1, pady=15, padx=10)
    # Name
    nameLabel=Label(searchWindow, text='Name', font=('bookman old style', 18))
    nameLabel.grid(row=2, column=0, padx=30, pady=15, sticky=W)
    nameEntry=Entry(searchWindow, font=('bookman old style', 15), width=24)
    nameEntry.grid(row=2, column=1, pady=15, padx=10)
    # Gender
    genderLabel=Label(searchWindow, text='Gender', font=('bookman old style',
18))
    genderLabel.grid(row=3, column=0, padx=30, pady=15, sticky=W)
    genderEntry=Entry(searchWindow, font=('bookman old style', 15), width=24)
    genderEntry.grid(row=3, column=1, pady=15, padx=10)
    # D.O.B.
    dobLabel=Label(searchWindow, text='D.O.B.', font=('bookman old style',
18))
    dobLabel.grid(row=4, column=0, padx=30, pady=15, sticky=W)
    dobEntry=Entry(searchWindow, font=('bookman old style', 15), width=24)
    dobEntry.grid(row=4, column=1, pady=15, padx=10)
    # Mobile No.
    mobileLabel=Label(searchWindow, text='Mobile No.', font=('bookman old
style', 18))
    mobileLabel.grid(row=5, column=0, padx=30, pady=15, sticky=W)
    mobileEntry=Entry(searchWindow, font=('bookman old style', 15), width=24)
    mobileEntry.grid(row=5, column=1, pady=15, padx=10)
    # Email ID
    emailLabel=Label(searchWindow, text='Email ID', font=('bookman old style',
18))
    emailLabel.grid(row=6, column=0, padx=30, pady=15, sticky=W)
    emailEntry=Entry(searchWindow, font=('bookman old style', 15), width=24)
    emailEntry.grid(row=6, column=1, pady=15, padx=10)
    # Search Student Button
    search_studentButton=ttk.Button(searchWindow, text='SEARCH STUDENT',
command=search_data)
    search_studentButton.grid(row=7, columnspan=2, pady=15)

def update_student():
    def update_data():
        query='update student set admno=%s, name=%s, gender=%s, dob=%s,
mobile=%s, email=%s where sno=%s'
        mycursor.execute(query, (admnoEntry.get(), nameEntry.get(),
genderEntry.get(), dobEntry.get(), mobileEntry.get(), emailEntry.get(),
snoEntry.get()))
        con.commit()
```

```
        messagebox.showinfo('Success', f'Modified Successfully!',
parent=updateWindow)
        updateWindow.destroy()
        show_student()

    updateWindow = Toplevel()
    updateWindow.title('Update Student | EduData')
    updateWindow.iconphoto(False, logo)
    updateWindow.resizable(False, False)
    updateWindow.grab_set()
    # S. No.
    snoLabel=Label(updateWindow, text='S. No.', font=('bookman old style',
18))
    snoLabel.grid(row=0, column=0, padx=30, pady=15, sticky=W)
    snoEntry=Entry(updateWindow, font=('bookman old style', 15), width=24)
    snoEntry.grid(row=0, column=1, pady=15, padx=10)
    # Adm. No.
    admnoLabel=Label(updateWindow, text='Adm. No.', font=('bookman old style',
18))
    admnoLabel.grid(row=1, column=0, padx=30, pady=15, sticky=W)
    admnoEntry=Entry(updateWindow, font=('bookman old style', 15), width=24)
    admnoEntry.grid(row=1, column=1, pady=15, padx=10)
    # Name
    nameLabel=Label(updateWindow, text='Name', font=('bookman old style', 18))
    nameLabel.grid(row=2, column=0, padx=30, pady=15, sticky=W)
    nameEntry=Entry(updateWindow, font=('bookman old style', 15), width=24)
    nameEntry.grid(row=2, column=1, pady=15, padx=10)
    # Gender
    genderLabel=Label(updateWindow, text='Gender', font=('bookman old style',
18))
    genderLabel.grid(row=3, column=0, padx=30, pady=15, sticky=W)
    genderEntry=Entry(updateWindow, font=('bookman old style', 15), width=24)
    genderEntry.grid(row=3, column=1, pady=15, padx=10)
    # D.O.B.
    dobLabel=Label(updateWindow, text='D.O.B.', font=('bookman old style',
18))
    dobLabel.grid(row=4, column=0, padx=30, pady=15, sticky=W)
    dobEntry=Entry(updateWindow, font=('bookman old style', 15), width=24)
    dobEntry.grid(row=4, column=1, pady=15, padx=10)
    # Mobile No.
    mobileLabel=Label(updateWindow, text='Mobile No.', font=('bookman old
style', 18))
    mobileLabel.grid(row=5, column=0, padx=30, pady=15, sticky=W)
    mobileEntry=Entry(updateWindow, font=('bookman old style', 15), width=24)
    mobileEntry.grid(row=5, column=1, pady=15, padx=10)
    # Email ID
    emailLabel=Label(updateWindow, text='Email ID', font=('bookman old style',
18))
```

```
    emailLabel.grid(row=6, column=0, padx=30, pady=15, sticky=W)
    emailEntry=Entry(updateWindow, font=('bookman old style', 15), width=24)
    emailEntry.grid(row=6, column=1, pady=15, padx=10)
    # Update Student Button
    search_studentButton=ttk.Button(updateWindow, text='UPDATE STUDENT',
command=update_data)
    search_studentButton.grid(row=7, columnspan=2, pady=15)

    # Updating
    indexing=studentTable.focus()
    content=studentTable.item(indexing)
    listdata=content['values']
    snoEntry.insert(0,listdata[0])
    admnoEntry.insert(0,listdata[1])
    nameEntry.insert(0,listdata[2])
    genderEntry.insert(0,listdata[3])
    dobEntry.insert(0,listdata[4])
    mobileEntry.insert(0,listdata[5])
    emailEntry.insert(0,listdata[6])


def show_student():
    query='select * from student order by sno'
    mycursor.execute(query)
    fetched_data=mycursor.fetchall()
    studentTable.delete(*studentTable.get_children())
    for data in fetched_data:
        studentTable.insert('', END, values=data)

def delete_student():
    indexing=studentTable.focus()
    content=studentTable.item(indexing)
    content_sno=content['values'][0]
    query='delete from student where sno=%s'
    mycursor.execute(query, content_sno)
    con.commit()
    messagebox.showinfo('Deleted',f'S. No.: {content_sno} is deleted
successfully!')
    query='select * from student'
    mycursor.execute(query)
    fetched_data=mycursor.fetchall()
    studentTable.delete(*studentTable.get_children())
    for data in fetched_data:
        studentTable.insert('', END, values=data)

def export_data():
    url=filedialog.asksaveasfilename(defaultextension='.csv')
    indexing=studentTable.get_children()
    newlist=[]
```

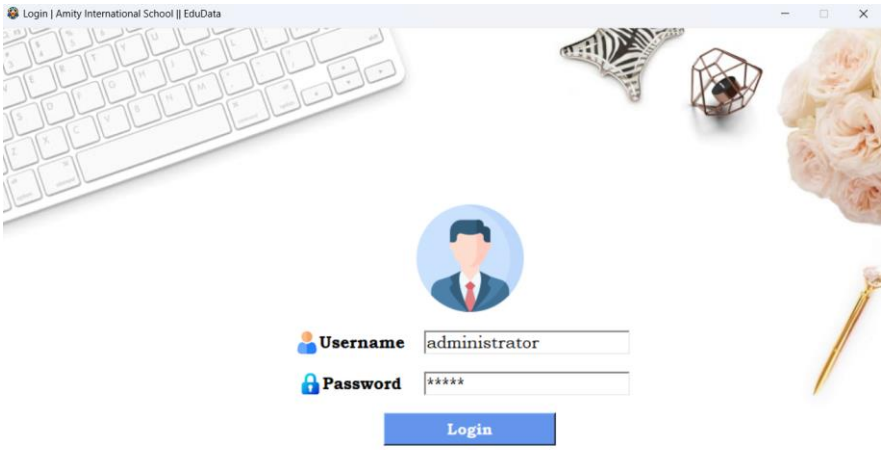
```
    for index in indexing:
        content=studentTable.item(index)
        datalist=content['values']
        newlist.append(datalist)

    table=pandas.DataFrame(newlist, columns=['S. No.', 'Adm. No', 'Name',
'Gender', 'D.O.B.', 'Mobile No.', 'Email ID'])
    table.to_csv(url, index=False)
    messagebox.showinfo('Success', 'Data Saved Successfully!')

def iexit():
    result=messagebox.askyesno('Confirm', 'Do you want to exit?')
    if result:
        root.destroy()
    else:
        pass



root = ttkthemes.ThemedTk()
root.get_themes()
root.set_theme('radiance')
root.geometry('1174x680+190+55')
root.resizable(0,0)
root.title('Class XII - L | Amity International School || EduData')

# Window Logo
logo=PhotoImage(file='ais.png')
root.iconphoto(False, logo)

# Date and Time
datetimeLabel = Label(root, font=('bookman old style', 18))
datetimeLabel.place(x=5, y=5)
clock()

# Heading
headLabel = Label(root, text='Amity International School', font=('bookman old
style', 28, 'bold'), width=30)
headLabel.place(x=250, y=0)

# Connect
connectButton = ttk.Button(root, text='Connect Database',
command=connect_database)
connectButton.place(x=980, y=0)

# Left Frame
leftframe = Frame(root)
leftframe.place(x=50, y=80, width=300, height=600)
```

```
# Left Frame Image
logo_image = PhotoImage(file='students.png')
logo_Label = Label(leftframe, image=logo_image)
logo_Label.grid(row=0, column=0)

# Buttons
addstudentButton = ttk.Button(leftframe, text='Add Student', width=25,
command=add_student, state=DISABLED)
addstudentButton.grid(row=1, column=0, pady=20)

searchstudentButton = ttk.Button(leftframe, text='Search Student', width=25,
command=search_student, state=DISABLED)
searchstudentButton.grid(row=2, column=0, pady=20)

updatestudentButton = ttk.Button(leftframe, text='Update Student', width=25,
command=update_student, state=DISABLED)
updatestudentButton.grid(row=3, column=0, pady=20)

showstudentButton = ttk.Button(leftframe, text='Show Student', width=25,
command=show_student, state=DISABLED)
showstudentButton.grid(row=4, column=0, pady=20)

deletestudentButton = ttk.Button(leftframe, text='Delete Student', width=25,
command=delete_student, state=DISABLED)
deletestudentButton.grid(row=5, column=0, pady=20)

exportdataButton = ttk.Button(leftframe, text='Export Data', width=25,
command=export_data, state=DISABLED)
exportdataButton.grid(row=6, column=0, pady=20)

exitButton = ttk.Button(leftframe, text='Exit', width=25, command=iexit)
exitButton.grid(row=7, column=0, pady=20)

# Right Frame
rightframe = Frame(root)
rightframe.place(x=350, y=80, width=820, height=570)

# Scroll Bar
sbx = Scrollbar(rightframe, orient=HORIZONTAL)
sbx.pack(side=BOTTOM, fill=X)
sby = Scrollbar(rightframe, orient=VERTICAL)
sby.pack(side=RIGHT, fill=Y)

# Student Table
studentTable = ttk.Treeview(rightframe, columns=('S. No.', 'Adm. No.', 'Name',
'Gender', 'D.O.B.', 'Mobile No.', 'Email ID'), xscrollcommand=sbx.set,
yscrollcommand=sby.set)
studentTable.pack(fill=BOTH, expand=1)
```

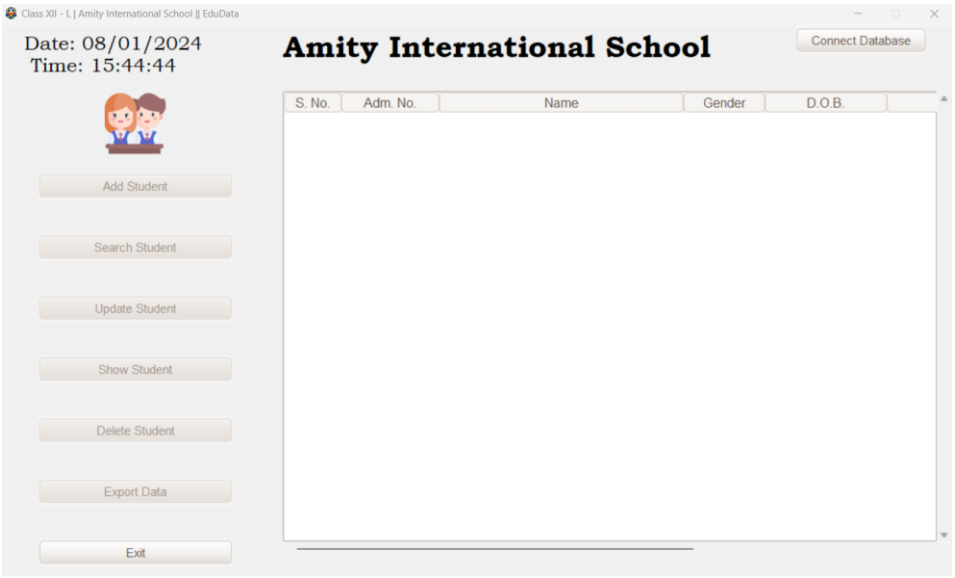
```
sbx.config(command=studentTable.xview)
sby.config(command=studentTable.yview)

# Column Headings
studentTable.heading('S. No.', text='S. No.')
studentTable.heading('Adm. No.', text='Adm. No.')
studentTable.heading('Name', text='Name')
studentTable.heading('Gender', text='Gender')
studentTable.heading('D.O.B.', text='D.O.B.')
studentTable.heading('Mobile No.', text='Mobile No.')
studentTable.heading('Email ID', text='Email ID')
studentTable.config(show='headings')

studentTable.column('S. No.', width=70, anchor=CENTER)
studentTable.column('Adm. No.', width=120, anchor=CENTER)
studentTable.column('Name', width=300)
studentTable.column('Gender', width=100, anchor=CENTER)
studentTable.column('D.O.B.', width=150, anchor=CENTER)
studentTable.column('Mobile No.', width=200, anchor=CENTER)
studentTable.column('Email ID', width=350)

style=ttk.Style()
style.configure('Treeview', rowheight=35, font=('bookaman old style', 12))
style.configure('Treeview.Heading', rowheight=40, font=('bookaman old style',
12))

root.mainloop()
```

Output

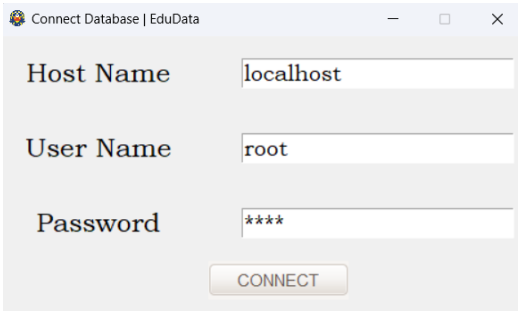
1. Run ‘main.py’ and login using credentials:



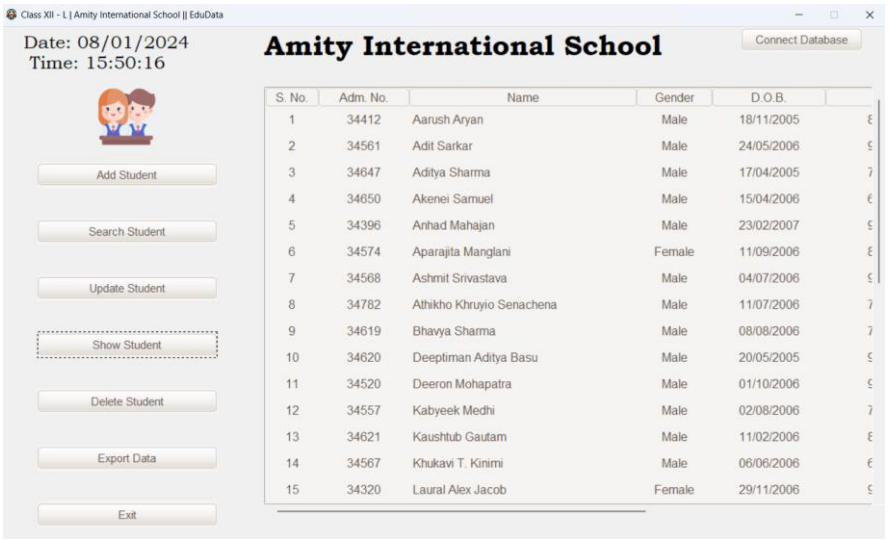
2. You’ll be directed to the ‘Dashboard’ i.e. sms.py.



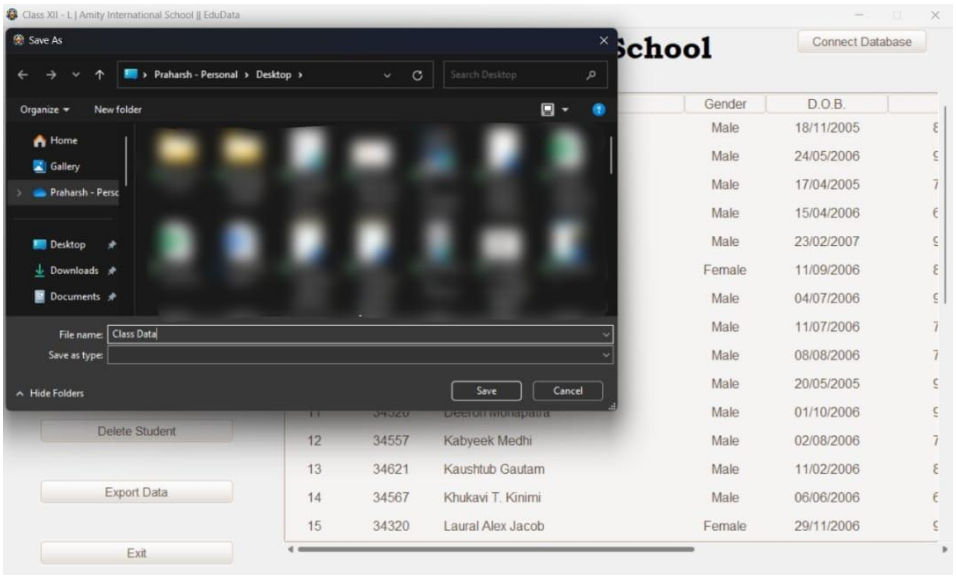
3. Now, let’s connect database using ‘Connect Database’ button.



4. Now ‘Show Student’ will display the list of students along with their data.



5. You can now search for any student using the ‘Search Student’ button. Also, use ‘Add Student’, ‘Update Student’ or ‘Delete Student’ to add, update delete the student’s data in the table.
6. The data can also be exported as excel file using ‘Export Data’ button.



.

User Manual

EduData Student Management System User Manual

Table of Contents

1. Introduction

1.1 Purpose of EduData

EduData is a comprehensive Student Management System designed to help educational institutions efficiently manage student data. It provides a centralized platform for storing, updating, and retrieving student information, ultimately enhancing operational efficiency and decision-making processes.

1.2 System Requirements

- Operating System: Windows, Linux, or macOS
- Python: Version 3.6 or higher
- Additional Python Libraries: tkinter, PIL, ttkthemes, pymysql, pandas

1.3 Installation

- Download the Software: Obtain the EduData software package from the provided source.
- Install Python: Ensure that Python 3.6 or a higher version is installed on your system.
- Install Required Libraries: Install the necessary Python libraries using the following command:

  pip install pillow ttkthemes pymysql pandas
- Run the Software: Execute the main script (e.g., main.py) to launch EduData.

2. Getting Started

2.1 Logging In

1. Open the software application.
2. Enter your username and password.

   Username: administrator

   Password: admin
3. Click the "Login" button.

2.2 Connecting to the Database

- After logging in, click the "Connect Database" button.
- Enter the required database connection details (hostname, username, password) and click "Connect."

3. Main Features

3.1 Add Student

- Click the "Add Student" button.
- Fill in the student details in the provided form.
- Click "Add Student" to save the information.

- Optionally, click "Clear Form" to reset the form.

3.2 Search Student

- Click the "Search Student" button.
- Enter the search criteria in the form.
- Click "Search Student" to retrieve matching records.

3.3 Update Student

- Select a student record from the displayed table.
- Clock the “Update Student” button.
- Modify the information in the form.
- Click "Update Student" to save the changes.

3.4 Show Student

- Click the "Show Student" button to display all student records.

3.5 Delete Student

- Select a student record from the displayed table.
- Click the "Delete Student" button to remove the selected record.

3.6 Export Data

- Click the "Export Data" button.
- Choose a location to save the CSV file.

3.7 Exit

- Click the "Exit" button to close the application.

4. Best Practices

4.1 Data Entry Guidelines

- Ensure all mandatory fields are filled during data entry.
- Avoid duplicate admission numbers to prevent errors.

4.2 Regular Backups

- Periodically export data for backup purposes.

4.3 Secure Access

- Keep login credentials confidential.
- Use proper access controls to restrict user privileges.

5. Troubleshooting

5.1 Connection Issues

- Verify database connection details.
- Check network connectivity.

5.2 Data Entry Errors

- Review error messages for guidance.
- Ensure data adheres to guidelines.

5.3 Exporting Data

- Confirm the destination folder is writable.
- Check for any error messages during export.

6. FAQs

6.1 General Questions

- Q: How do I reset my password?
  A: Passwords can only be reset by an administrator. Please contact your system administrator for assistance.

6.2 Technical Issues

- Q: The software is not starting. What should I do?
  A: Ensure that you have met all system requirements and have followed the installation steps correctly. Check for any error messages and consult the troubleshooting section.

This user manual provides a comprehensive guide to using the EduData Student Management System. If you encounter any issues or have additional questions, please refer to the troubleshooting section or contact your system administrator for assistance.

Bibliography

1. **Tkinter Documentation:**
   Author: Python Software Foundation
   URL: https://docs.python.org/3/library/tkinter.html
   Description: Official documentation for Tkinter, the standard GUI toolkit for Python.
2. **Pillow Documentation:**
   Author: Python Imaging Library (PIL) Fork Maintainers
   URL: https://pillow.readthedocs.io/en/stable/
   Description: Documentation for the Pillow library, used for working with images in the project.
3. **MySQL Connector/Python Documentation:**
   Author: Oracle Corporation
   URL: https://dev.mysql.com/doc/connector-python/en/
   Description: Official documentation for MySQL Connector/Python, which facilitates the connection between Python and MySQL database.
4. **CBSE Class XII Computer Science Syllabus:**
   Author: Central Board of Secondary Education (CBSE)
   URL: http://cbseacademic.nic.in/curriculum.html
   Description: Official syllabus for Computer Science in CBSE Class XII.