```python
from distutils import command
from tkinter import *
from tkinter import ttk
from typing_extensions import _AnnotatedAlias
from PIL import Image,ImageTk
import os
import pickle
import mysql.connector  as sql
from tkinter import messagebox
from datetime import date
from datetime import time
from datetime import *
import requests
from bs4 import BeautifulSoup
import time
import dashboard
import csv
import regist_batch

def batch_screen():
    def click_go_to_dashboard():
        """returns AdminDashboard class when clicked go to dashboard"""
        dashboard.dashboard()
        root.withdraw()

    def click_delete_batch():
            """when clicked delete batch, it will require to select the batch and after selecting and
            performing the delete method, it will ask the admin either they are sure they want to delete that batch
            or not if yes then batch containing that id in batch table is deleted."""
            try:
                #obj_batch_registration_database = Model_class.batch_registration.GetDatabase('use cms;')
                #db_connection.create(obj_batch_registration_database.get_database())
                a=load_data()
                host=a[0]
                username = a[2]
                password = a[3]
                port=a[1]

                spec=sql.connect(host=host,user=username,password=password,port=port,database="sms")
                mycur=spec.cursor()

                batch_view_content = batch_tree.focus()
                batch_view_items = batch_tree.item(batch_view_content)
                batch_view_values = batch_view_items['values'][0]
                ask = messagebox.askyesno("Warning",
                                        f"Are you sure you want to delete batch having id {batch_view_values}")

                if ask is True:
                    query = f"delete from batch where batch_id={batch_view_values};"
                    #db_connection.delete(query, (batch_view_values,))
                    mycur.execute(query)
                    spec.commit()
                    messagebox.showinfo("Success", f" batch id {batch_view_values} deleted Successfully")

                    click_view_all()
                else:
                    pass

            except BaseException as msg:
                print(msg)
                messagebox.showerror("Error",
                                    "There is some error deleting the data\n Make sure you have Selected the data")


    def up_sec():

        def upd_sec():
            """updates the data of batch from entry fields"""
            try:
                #obj_batch_database = Model_class.batch_registration.GetDatabase('use cms;')
                #db_connection.create(obj_batch_database.get_database())

                #get_id =.get_id
                data_id = get_id[0]
                # print(data_id)
                a=load_data()
                host=a[0]
                username = a[2]
                password = a[3]
                port=a[1]

                spec=sql.connect(host=host,user=username,password=password,port=port,database="sms")
                mycur=spec.cursor()
                #obj_batch_database = Model_class.batch_registration.BatchRegistration(batch_name_entry.get(),
                                                                                        #batch_year_entry.get(),
                                                                                        #batch_intake_combo.get(),
                                                                                        #reg_date)
                query = f"update batch set batch_name='{batch_name_entry.get()}',batch_year='{batch_year_entry.get()}', batch_intake='{batch_intake_combo.get()}'" \
                        f" where batch_id={data_id}"

                mycur.execute(query)
                spec.commit()
                #values = (obj_batch_database.get_name(), obj_batch_database.get_year(),
                            #obj_batch_database.get_intake(), data_id)
                click_view_all()
                #db_connection.update(query, values)

                ask = messagebox.askyesnocancel("Success",
                                                f"Data having \n Batch Name={batch_name_entry.get()} \n Updated Successfully\n"
                                                f"Do you want to Go Batch Dashboard")
                if ask is True:
                    pass

            except BaseException as msg:
                print(msg)
                messagebox.showerror("Error", f"Error due to{msg}")


        def tree_event_handle():
            try:
                #obj_student_database = Model_class.student_registration.GetDatabase('use cms;')
```

```python
                #db_connection.create(obj_student_database.get_database())

                tree_view_content = batch_tree.focus()
                tree_view_items = batch_tree.item(tree_view_content)
                # print(tree_view_items)
                tree_view_values = tree_view_items['values']
                tree_view_id = tree_view_items['values'][0]
                # print(tree_view_id)
                list_of_tree.clear()
                get_id.clear()
                get_id.append(tree_view_id)
                for i in tree_view_values:
                    list_of_tree.append(i)

                ask = messagebox.askyesno("Confirm",
                                        f"Do you want to Update Student having id {
tree_view_id}")
                #if ask is True:
                    #update_sec()

            except BaseException as msg:
                print(msg)
                messagebox.showerror("Error",
                                    "There is some error updating the data\n Make sure you
have Selected the data")

        tree_event_handle()



        global batch_name_entry,batch_year_entry,batch_intake_combo

        def click_clear_button():
            batch_name_entry.delete(0, END)
            batch_year_entry.delete(0, END)
            batch_intake_combo.current(0)



        def update_sec():

            global batch_name_entry,batch_year_entry,batch_intake_combo

            root = Toplevel()
            root.title('BATCH REGISTRATION FORM - COLLEGE MANAGEMENT SYSTEM')
            root.geometry('1067x600')
            root.config(bg="#f29844")
            root.resizable(False, False)


            # ======================Backend connection=============
            #db_connection = Backend.connection.DatabaseConnection()

            # creating frame for Register
            # img = img
            # dummylabel = Label(root, image=img)
            # dummylabel.place(x=30, y=30)

            reg_frame = Frame(root, bg="#ffffff", width=1000, height=560)
            reg_frame.place(x=30, y=30)



            heading = Label(reg_frame, text="Batch Registration Form", font=('yu gothic ui', 20, "bold"), bg="white",
                                    fg='black',
                                    bd=5,
                                    relief=FLAT)
            heading.place(x=200, y=0, width=600)



            batch_frame = LabelFrame(reg_frame, text="Batch Details", bg="white", fg="#4f4e4d", height=380,
                                            width=800, borderwidth=2.4,
                                            font=("yu gothic ui", 13, "bold"))
            batch_frame.config(highlightbackground="red")
            batch_frame.place(x=100, y=90)

            # ========================================================================
            # ===========================batch Name==================================
            # ========================================================================

            batch_name_label = Label(batch_frame, text="Batch Name ", bg="white", fg="#4f4e4d",
                                            font=("yu gothic ui", 13, "bold"))
            batch_name_label.place(x=160, y=65)

            batch_name_entry = Entry(root, highlightthickness=0, relief=FLAT, bg="white", fg="#6b6a69",
                                            font=("yu gothic ui semibold", 12))
            batch_name_entry.place(x=400, y=212, width=345)  # trebuchet ms

            batch_name_line = Canvas(root, width=345, height=1.5, bg="#bdb9b1", highlightthickness=0)
            batch_name_line.place(x=400, y=234)

            # ========================================================================
            # ===========================batch YEAR ==================================
            # ========================================================================
            date = time.strftime("%Y")

            batch_year_label = Label(batch_frame, text="Batch Year ", bg="white", fg="#4f4e4d",
                                            font=("yu gothic ui", 13, "bold"))
            batch_year_label.place(x=160, y=115)

            batch_year_entry = Entry(root, highlightthickness=0, relief=FLAT, bg="white", fg="#6b6a69",
                                            font=("yu gothic ui semibold", 12))
            batch_year_entry.place(x=390, y=262, width=355)  # trebuchet ms

            batch_year_line = Canvas(root, width=355, height=1.5, bg="#bdb9b1", highlightthickness=0)
            batch_year_line.place(x=390, y=284)
            batch_year_entry.insert(0, date)

            # ========================================================================
            # ===========================batch Intake==================================
            # ========================================================================
            root.option_add("*TCombobox*Listbox*Foreground", '#f29844')

            batch_intake_label = Label(batch_frame, text="Batch Intake ", bg="white", fg="#4f4e4d",
                                            font=("yu gothic ui", 13, "bold"))
            batch_intake_label.place(x=160, y=165)

            batch_intake_combo = ttk.Combobox(batch_frame, font=('yu gothic ui semibold', 12, 'bold'),
                                                    state='readonly',
                                                    width=35)
            batch_intake_combo['values'] = ("January", "February", "March", "April", "May", "June", "July", "August",
                                                    "September", "October", "November", "December")
            batch_intake_combo.current(0)
            batch_intake_combo.place(x=270, y=167)
            # batch_intake_line.place(x=410, y=424)

            reg_date = time.strftime("%Y/%m/%d")


            # ========================================================================
            # ===========================Register options ====================================
            # ========================================================================
            submit_img = ImageTk.PhotoImage(file='Pics\\submit.png')
            submit = Button(batch_frame, image=submit_img,
                                    font=("yu gothic ui", 13, "bold"), relief=FLAT, activebackground="white"
                                    , borderwidth=0, background="white", cursor="hand2", command=upd_sec)
            submit.image = submit_img
            submit.place(x=90, y=267)

            clear_img = ImageTk.PhotoImage(file='Pics\\clear.png')
            clear_button = Button(batch_frame, image=clear_img,
                                        font=("yu gothic ui", 13, "bold"), relief=FLAT, activebackground="white"
                                        , borderwidth=0, background="white", cursor="hand2",
                                        command=click_clear_button)
            clear_button.image = clear_img
            clear_button.place(x=250, y=270)

            back_img = ImageTk.PhotoImage(file='Pics\\back.png')
            back_button = Button(batch_frame, image=back_img,
                                        font=("yu gothic ui", 13, "bold"), relief=FLAT, activebackground="white"
                                        , borderwidth=0, background="white", cursor="hand2")
            back_button.image = back_img
            back_button.place(x=410, y=270)

            exit_img = ImageTk.PhotoImage(file='Pics\\exit.png')
            exit_button = Button(batch_frame, image=exit_img,
                                        font=("yu gothic ui", 13, "bold"), relief=FLAT, activebackground="white"
                                        , borderwidth=0, background="white", cursor="hand2", command=exit)
            exit_button.image = exit_img
            exit_button.place(x=570, y=270)

            a = list_of_tree

            try:
                batch_name_entry.insert(0, a[1])
                batch_year_entry.delete(0,END)
                batch_year_entry.insert(0, a[2])
                batch_intake_combo.insert(0, a[3])

            except IndexError as msg:
                    print(msg)

        update_sec()

    root = Toplevel()
    root.geometry("1067x600")
    root.title("Batch Management Dashboard - College Management System")
    #root.iconbitmap('images\\logo.ico')
    root.resizable(False, False)

    list_of_tree = []
    get_id = []

    manage_student_frame_r = Image.open('Pics\\student_frame.png').resize((1067,600),Image.
ANTIALIAS)
    manage_student_frame = ImageTk.PhotoImage(manage_student_frame_r)
    image_panel = Label(root, image=manage_student_frame)
    image_panel.image = manage_student_frame
    image_panel.pack(fill='both', expand='yes')

    #db_connection = Backend.connection.DatabaseConnection()



    heading = Label(root, text="Batch Management Dashboard", font=('yu gothic ui', 20, "bold"
), bg="white",
                            fg='black',
                            bd=5,
                            relief=FLAT)
    heading.place(x=420, y=26, width=640)
    #slider()
    #heading_color()


    # ========================================================================
    # =======================Left frame ======================================
    # ========================================================================

    #left frame
    left_view_frame = Frame(root, bg="white")
    left_view_frame.place(x=35, y=89, height=470, width=250)


    #tree view frame
    tree_view_frame = Frame(root, bg="white")
    tree_view_frame.place(x=301, y=90, height=473, width=730)

    # ========================================================================
    # =======================frame for personal credentials =================
    # ========================================================================

    personal_frame = LabelFrame(left_view_frame, text="Batch Management Options", bg="white",
fg="#4f4e4d",height=460,width=240, borderwidth=2.4,font=("yu gothic ui", 12, "bold"))
    personal_frame.config(highlightbackground="red")
    personal_frame.place(x=5, y=8)

    # =========================================================================
    # ==========================Add Batch button==============================
    # =========================================================================
    add_batch_r = Image.open('Pics\\add_batch.png').resize((225,25),Image.ANTIALIAS)
    add_batch = ImageTk.PhotoImage(add_batch_r)
    add_batch_button = Button(personal_frame, image=add_batch, relief=FLAT, borderwidth=0,
                                     activebackground="white", bg="white", cursor="hand2",
command=regist_batch.regist_batch)
    add_batch_button.image = add_batch
    add_batch_button.place(x=8, y=100)
    # add_student_button.place(x=36, y=295)

    # =====================================================================
    # ===========================Update batch button===============================
    # =====================================================================

    update_batch_r = Image.open('Pics\\update_batch.png').resize((225,25),Image.ANTIALIAS)
    update_batch = ImageTk.PhotoImage(update_batch_r)
    update_batch_button = Button(personal_frame, image=update_batch, relief=FLAT, borderwidth=
0,
                                         activebackground="white", bg="white", cursor="hand2",
command=up_sec)
    update_batch_button.image = update_batch
    update_batch_button.place(x=8, y=165)
    # update_student_button.place(x=36, y=355)

    # =====================================================================
    # ===========================Delete batch button===============================
    # =====================================================================

    delete_batch_r = Image.open('Pics\\delete_batch.png').resize((225,25),Image.ANTIALIAS)
    delete_batch = ImageTk.PhotoImage(delete_batch_r)
    delete_batch_button = Button(personal_frame, image=delete_batch, relief=FLAT, borderwidth=
0,
                                         activebackground="white", bg="white", cursor="hand2",
command=click_delete_batch)
    delete_batch_button.image = delete_batch
    delete_batch_button.place(x=8, y=230)

    # delete_student_button.place(x=36, y=405)

    # =====================================================================
    # ===========================Goto Main dashboard button===============================
    # =====================================================================

    goto_dashboard_r = Image.open('Pics\\goto_dashboard.png').resize((225,25),Image.ANTIALIAS)
    goto_dashboard = ImageTk.PhotoImage (goto_dashboard_r)
    goto_dashboard_button = Button(personal_frame, image=goto_dashboard, relief=FLAT,
borderwidth=0,activebackground="white", bg="white", cursor="hand2",command=
click_go_to_dashboard)
    goto_dashboard_button.image = goto_dashboard
    goto_dashboard_button.place(x=5, y=295)

    # ====================================================================
    # =======================Starting Tree View============================
    # ====================================================================

    def load_data():
        f=open("Credentials.csv","r")
        s=csv.reader(f,delimiter="-")
        d=[]
        for i in s:
            d.append(i)
        a=d[::-1]
        return (a[0])

    def click_view_all():
            """it will show all the data contains on the batch table of cms database, when
clicked by default this method
```

```
            is called while initializing the class ManageBatch. Exception is handled to avoid run time error which may
            cause by user."""
            try:
                #obj_student_database = Model_class.student_registration.GetDatabase('use cms;')
                #db_connection.create(obj_student_database.get_database())
                a=load_data()
                host=a[0]
                username = a[2]
                password = a[3]
                port=a[1]

                spec=sql.connect(host=host,user=username,password=password,port=port,database="sms")
                mycur=spec.cursor()
                query = "select * from batch;"
                mycur.execute(query)
                data = mycur.fetchall()
                # print(data)
                batch_tree.delete(*batch_tree.get_children())
                for values in data:
                    data_list = [values[0], values[1], values[2], values[3], values[4]]
                    print(data_list)
                    batch_tree.insert('', END, values=data_list)

            except BaseException as msg:
                print(msg)



    style = ttk.Style()
    style.configure("Treeview.Heading", font=('yu gothic ui', 10, "bold"), foreground="red")

    scroll_x = Scrollbar(tree_view_frame, orient=HORIZONTAL)
    scroll_y = Scrollbar(tree_view_frame, orient=VERTICAL)
    batch_tree = ttk.Treeview(tree_view_frame,
                                    columns=(
                                        "BATCH ID", "BATCH NAME", "BATCH YEAR", "BATCH INTAKE", "REGISTRATION DATE"),
                                    xscrollcommand=scroll_x.set, yscrollcommand=scroll_y.set)
    scroll_x.pack(side=BOTTOM, fill=X)
    scroll_y.pack(side=RIGHT, fill=Y)
    scroll_x.config(command=batch_tree.xview)
    scroll_y.config(command=batch_tree.yview)

    # ==========================TreeView Heading====================
    batch_tree.heading("BATCH ID", text="BATCH ID")
    batch_tree.heading("BATCH NAME", text="BATCH NAME")
    batch_tree.heading("BATCH YEAR", text="BATCH YEAR")
    batch_tree.heading("BATCH INTAKE", text="BATCH INTAKE")
    batch_tree.heading("REGISTRATION DATE", text="REGISTRATION DATE")
    batch_tree["show"] = "headings"

    # ==========================TreeView Column====================
    batch_tree.column("BATCH ID", width=50)
    batch_tree.column("BATCH NAME", width=150)
    batch_tree.column("BATCH YEAR", width=100)
    batch_tree.column("BATCH INTAKE", width=100)
    batch_tree.column("REGISTRATION DATE", width=100)
    batch_tree.pack(fill=BOTH, expand=1)

    #batch_tree.bind("<Delete>", click_delete_key)
    #batch_tree.bind("<Button-3>", do_popup)
    #batch_tree.bind("<Double-Button-1>", tree_double_click)
    #batch_tree.bind("<Return>", tree_double_click)
    #search_start()
    #search_by.current(0)

    click_view_all()


    #root.mainloop()

if __name__ =="__main__":
    batch_screen()
```