```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_course



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




    def click_delete_course():
            """when clicked delete courses, it will require to select the courses and after selecting and
            performing the delete method, it will ask the admin either they are sure they want to delete that course
            or not if yes then course containing that id in course table is deleted."""
            try:
                #obj_course_registration_database = Model_class.course_registration.GetDatabase('use cms;')
                #self.db_connection.create(obj_course_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()

                course_view_content = course_tree.focus()
                course_view_items = course_tree.item(course_view_content)
                course_view_values = course_view_items['values'][0]
                ask = messagebox.askyesno("Warning",
                                          f"Are you sure you want to delete course having id {course_view_values}")

                if ask is True:
                    query = f"delete from course where course_id={course_view_values};"
                    mycur.execute(query)
                    spec.commit()
                    #db_connection.delete(query, (course_view_values,))
```

```
 59                     messagebox.showinfo("Success", f" course id {course_view_values} deleted
    Successfully")
 60                     click_view_all()
 61
 62                 else:
 63                     pass
 64
 65             except BaseException as msg:
 66                 print(msg)
 67                 messagebox.showerror("Error",
 68                                      "There is some error deleting the data\n Make sure you
    have Selected the data")
 69
 70
 71     def updatebutton():
 72
 73
 74         def up_cho():
 75
 76             try:
 77                 #obj_course_database = Model_class.course_registration.GetDatabase('use cms;')
 78                 #self.db_connection.create(obj_course_database.get_database())
 79
 80                 #get_id = get_id
 81                 data_id = get_id[0]
 82                 # print(data_id)
 83                 a=load_data()
 84                 host=a[0]
 85                 username = a[2]
 86                 password = a[3]
 87                 port=a[1]
 88
 89                 spec=sql.connect(host=host,user=username,password=password,port=port,database=
    "sms")
 90                 mycur=spec.cursor()
 91                 #obj_course_database = Model_class.course_registration.CourseRegistration(self
    .course_name_entry.get(),
 92                                                                                          #
    self.course_duration_entry.get(),
 93                                                                                          #
    self.course_credit_entry.get(),
 94
 95                                                                                          #
    self.reg_date)
 96                 print("data id",data_id)
 97                 query = f"update course set course_name='{course_name_entry.get()}',
    course_duration='{course_duration_entry.get()}', course_credit='{course_credit_entry.get()}'"
     \
 98                         f" where course_id={data_id}"
 99                 mycur.execute(query)
100                 spec.commit()
101
102                 #data1 = mycur.fetchall()
103                 #print("data",data1)
104                 click_view_all()
105                 #values = (obj_course_database.get_name(), obj_course_database.get_duration(),
106                             #obj_course_database.get_credit(), data_id)
107
108                 #self.db_connection.update(query, values)
109
110                 ask = messagebox.askyesnocancel("Success", f"Data having \n Course Name=\n
    Updated Successfully\n"
111                                                             f"Do you want to Go obj_course
```

```
111 Dashboard")
112                 if ask is True:
113                     pass
114                     #course_screen()
115                     #root.withdraw()
116 
117             except BaseException as msg:
118                 print(msg)
119                 messagebox.showerror("Error", f"Error due to{msg}")
120 
121 
122 
123         #regist_course.course_reg()
124         #regist_course.course_reg.submit.configure(command=update)
125 
126         #a = list_of_tree
127         #print(a,"printing a")
128 
129 
130 
131         def tree_event_handle():
132             try:
133                 #obj_course_database = Model_class.course_registration.GetDatabase('use cms;')
134                 #self.db_connection.create(obj_course_database.get_database())
135 
136                 tree_view_content = course_tree.focus()
137                 tree_view_items = course_tree.item(tree_view_content)
138                 # print(tree_view_items)
139                 tree_view_values = tree_view_items['values']
140                 tree_view_id = tree_view_items['values'][0]
141                 # print(tree_view_id)
142                 list_of_tree.clear()
143                 get_id.clear()
144                 get_id.append(tree_view_id)
145                 for i in tree_view_values:
146                     list_of_tree.append(i)
147 
148                 ask = messagebox.askyesno("Confirm",
149                                           f"Do you want to Update Student having id {
    tree_view_id}")
150                 #if ask is True:
151                     #click_update_course()
152 
153             except BaseException as msg:
154                 print(msg)
155                 messagebox.showerror("Error",
156                                      "There is some error updating the data\n Make sure you
     have Selected the data")
157         tree_event_handle()
158 
159         def update():
160         #"""updates the data of course from entry fields"""
161 
162             #tree_event_handle()
163 
164             #a = list_of_tree
165             #print(a,"printing a")
166 
167 
168 
169         #count = 0
170         #if count <= len(txt):
171             #count = 0
```

```
172             #text = ''
173             #heading.config(text=text)
174 
175         #heading.place(x=150, y=0, width=800)
176 
177             global course_name_entry,course_duration_entry,course_credit_entry
178 
179             root = Toplevel()
180             root.title('COURSE REGISTRATION FORM - COLLEGE MANAGEMENT SYSTEM')
181             root.geometry('1067x600')
182             root.config(bg="#f29844")
183             #root.iconbitmap('images\\logo.ico')
184             root.resizable(False, False)
185             #manage_student_frame = ImageTk.PhotoImage(file='Pics\\student_frame.png')
186 
187 
188             # =====================Variables======================
189 
190             reg_frame = Frame(root, bg="#ffffff", width=1000, height=560)
191             reg_frame.place(x=30, y=30)
192 
193             heading = Label(reg_frame, text="Course Registration Form", font=('yu gothic ui',
    20, "bold"), bg="white",
194                                     fg='black',
195                                     bd=5,
196                                     relief=FLAT)
197             heading.place(x=200, y=0, width=600)
198 
199 
200             course_frame = LabelFrame(reg_frame, text="Course Details", bg="white", fg="#
    4f4e4d", height=380,
201                                               width=800, borderwidth=2.4,
202                                               font=("yu gothic ui", 13, "bold"))
203             course_frame.config(highlightbackground="red")
204             course_frame.place(x=100, y=90)
205 
206 
207 
208 
209             course_name_label = Label(course_frame, text="Course Name ", bg="white", fg="#
    4f4e4d",
210                                               font=("yu gothic ui", 13, "bold"))
211             course_name_label.place(x=160, y=65)
212 
213             course_name_entry = Entry(root, highlightthickness=0, relief=FLAT, bg="white", fg=
    "#6b6a69",
214                                               font=("yu gothic ui semibold", 12))
215             course_name_entry.place(x=410, y=212, width=335)  # trebuchet ms
216 
217             course_name_line = Canvas(root, width=335, height=1.5, bg="#bdb9b1",
    highlightthickness=0)
218             course_name_line.place(x=410, y=234)
219 
220             # ============================================================================
221             # ============================Course Duration=================================
222             # ============================================================================
223 
224             course_duration_label = Label(course_frame, text="Course Duration ", bg="white",
    fg="#4f4e4d",
225                                                   font=("yu gothic ui", 13, "bold"))
226             course_duration_label.place(x=160, y=115)
227 
228             course_duration_entry = Entry(root, highlightthickness=0, relief=FLAT, bg="white"
```

```
228 , fg="#6b6a69",
229                                                 font=("yu gothic ui semibold", 12))
230             course_duration_entry.place(x=430, y=262, width=315)  # trebuchet ms
231 
232             course_duration_line = Canvas(root, width=315, height=1.5, bg="#bdb9b1",
    highlightthickness=0)
233             course_duration_line.place(x=430, y=284)
234 
235             # ============================================================================
236             # ============================Course Credit=================================
237             # ============================================================================
238 
239             course_credit_label = Label(course_frame, text="Course Credit ", bg="white", fg="#
    4f4e4d",
240                                                 font=("yu gothic ui", 13, "bold"))
241             course_credit_label.place(x=160, y=165)
242 
243             course_credit_entry = Entry(root, highlightthickness=0, relief=FLAT, bg="white",
    fg="#6b6a69",
244                                                 font=("yu gothic ui semibold", 12))
245             course_credit_entry.place(x=410, y=312, width=335)  # trebuchet ms
246 
247             course_credit_line = Canvas(root, width=335, height=1.5, bg="#bdb9b1",
    highlightthickness=0)
248             course_credit_line.place(x=410, y=334)
249 
250             reg_date = time.strftime("%Y/%m/%d")
251 
252             # ============================================================================
253             # ============================Register options
    ====================================
254             # ============================================================================
255             submit_img = ImageTk.PhotoImage(file='Pics\submit.png')
256 
257             submit = Button(course_frame, image=submit_img,
258                                     font=("yu gothic ui", 13, "bold"), relief=FLAT,
    activebackground="white"
259                                     , borderwidth=0, background="white", cursor="hand2",
    command=up_cho)
260             submit.image = submit_img
261             submit.place(x=90, y=267)
262 
263             clear_img = ImageTk.PhotoImage(file='Pics\\clear.png')
264             clear_button = Button(course_frame, image=clear_img,
265                                         font=("yu gothic ui", 13, "bold"), relief=FLAT,
    activebackground="white"
266                                         , borderwidth=0, background="white", cursor="hand2"
267                                         )
268             clear_button.image = clear_img
269             clear_button.place(x=250, y=270)
270 
271             back_img = ImageTk.PhotoImage (file='Pics\\back.png')
272             back_button = Button(course_frame, image=back_img,
273                                         font=("yu gothic ui", 13, "bold"), relief=FLAT,
    activebackground="white"
274                                         , borderwidth=0, background="white", cursor="hand2")
275             back_button.image = back_img
276             back_button.place(x=410, y=270)
277 
278             exit_img = ImageTk.PhotoImage(file='Pics\\exit.png')
279             exit_button = Button(course_frame, image=exit_img,
280                                         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)

            #tree_event_handle()

            a = list_of_tree
            print(a,"printing a")

            try:
                course_name_entry.insert(0, a[1])
                course_duration_entry.insert(0, a[2])
                course_credit_entry.insert(0, a[3])
                #regist_course.course_reg.submit.configure(command=update)
                txt = f" You are updating course '{a[1]}'"

            except IndexError as msg:
                print(msg)

        update()
                #root.mainloop()

        a = list_of_tree
        print(a,"printing a")



        '''
            try:
                #obj_course_database = Model_class.course_registration.GetDatabase('use cms;')
                #self.db_connection.create(obj_course_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_course_database = Model_class.course_registration.CourseRegistration(self.course_name_entry.get(),
                                                                                         #self.course_duration_entry.get(),
                                                                                         #self.course_credit_entry.get(),
                                                                                         #self.reg_date)
                query = f"update course set course_name='{regist_course.course_name_entry.get()}', course_duration='{regist_course.course_duration_entry.get()}', course_credit={regist_course.course_credit_entry.get()}" \
                        f" where course_id={data_id}"
                mycur.execute(query)
                #values = (obj_course_database.get_name(), obj_course_database.get_duration(),
                            #obj_course_database.get_credit(), data_id)

                #self.db_connection.update(query, values)

```

```
                ask = messagebox.askyesnocancel("Success", f"Data having \n Course Name=\n Updated Successfully\n"
                                                           f"Do you want to Go obj_course Dashboard")
                #if ask is True:
                    #win = Toplevel()
                    #Frontend.manage_course.ManageCourse(win)
                    #root.withdraw()
                    #win.deiconify()

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



        #regist_course.course_reg()
        #regist_course.course_reg.submit.configure(command=update)

        #a = list_of_tree
        #print(a,"printing a")

        '''
        try:
            regist_course.course_reg.course_name_entry().insert(0, a[1])
            regist_course.course_reg.course_duration_entry().insert(0, a[2])
            regist_course.course_reg.course_credit_entry().insert(0, a[3])
            regist_course.course_reg.submit.configure(command=update)
            txt = f" You are updating course '{a[1]}'"

        except IndexError as msg:
            print(msg)

        count = 0
        if count <= len(txt):
            count = 0
            text = ''
            heading.config(text=text)
        heading.place(x=150, y=0, width=800)'''



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


    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="Course Management Dashboard", font=('yu gothic ui', 20, "bold"), bg="white",
                            fg='black',
```

```
                            bd=5,
                            relief=FLAT)
    heading.place(x=420, y=23)
    #slider()
    #heading_color()


    #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="Course 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 Course button=============================
    # ================================================================

    add_course_r = Image.open('Pics\\add_course.png').resize((225,25),Image.ANTIALIAS)
    add_course = ImageTk.PhotoImage(add_course_r)
    add_course_button = Button(personal_frame, image=add_course, relief=FLAT, borderwidth=0,
activebackground="white", bg="white", cursor="hand2",command=regist_course.course_reg)
    add_course_button.image = add_course
    add_course_button.place(x=8, y=100)
    # add_student_button.place(x=36, y=295)

    # ================================================================
    # ============================Update course button=============================
    # ================================================================
    update_course_r = Image.open('Pics\\update_course.png').resize((225,25),Image.ANTIALIAS)
    update_course = ImageTk.PhotoImage(update_course_r)
    update_course_button = Button(personal_frame, image=update_course, relief=FLAT,
borderwidth=0,activebackground="white", bg="white", cursor="hand2",command=updatebutton)
    update_course_button.image = update_course
    update_course_button.place(x=8, y=165)
    # update_student_button.place(x=36, y=355)

    # ================================================================
    # ============================Delete course button=============================
    # ================================================================

    delete_course_r = Image.open('Pics\\delete_course.png').resize((225,25),Image.ANTIALIAS)
    delete_course = ImageTk.PhotoImage(delete_course_r)
    delete_course_button = Button(personal_frame, image=delete_course, relief=FLAT,
borderwidth=0,activebackground="white", bg="white", cursor="hand2",command=click_delete_course
)
    delete_course_button.image = delete_course
    delete_course_button.place(x=8, y=230)
    # delete_student_button.place(x=36, y=405)

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

```
453     # ============================================================================
454 
455     goto_dashboard_r = Image.open('Pics\\goto_dashboard.png').resize((225,25),Image.ANTIALIAS)
456     goto_dashboard = ImageTk.PhotoImage (goto_dashboard_r)
457     goto_dashboard_button = Button(personal_frame, image=goto_dashboard, relief=FLAT,
    borderwidth=0,activebackground="white", bg="white", cursor="hand2",command=
    click_go_to_dashboard)
458     goto_dashboard_button.image = goto_dashboard
459     goto_dashboard_button.place(x=5, y=295)
460 
461     list_of_tree = []
462     get_id = []
463 
464 
465     def load_data():
466         f=open("Credentials.csv","r")
467         s=csv.reader(f,delimiter="-")
468         d=[]
469         for i in s:
470             d.append(i)
471         a=d[::-1]
472         return (a[0])
473 
474 
475     def click_view_all():
476         """it will show all the data contains on the course table of cms database, when
    clicked by default this method
477         is called while initializing the class ManageCourse. Exception is handled to avoid run
     time error which may
478         cause by user."""
479         try:
480             #obj_student_database = Model_class.student_registration.GetDatabase('use cms;')
481             #self.db_connection.create(obj_student_database.get_database())
482             a=load_data()
483             host=a[0]
484             username = a[2]
485             password = a[3]
486             port=a[1]
487 
488             spec=sql.connect(host=host,user=username,password=password,port=port,database="sms
    ")
489             mycur=spec.cursor()
490             query = "select * from course;"
491             mycur.execute(query)
492             data = mycur.fetchall()
493             #print(data)
494             course_tree.delete(*course_tree.get_children())
495             for values in data:
496                 data_list = [values[0], values[1], values[2], values[3],values[4]]
497                 print(data_list)
498                 course_tree.insert('', END, values=data_list)
499 
500         except BaseException as msg:
501             print(msg)
502 
503 
504 
505 
506 
507 
508     style = ttk.Style()
509     style.configure("Treeview.Heading", font=('yu gothic ui', 10, "bold"), foreground="red")
510 
```

```
    scroll_x = Scrollbar(tree_view_frame, orient=HORIZONTAL)
    scroll_y = Scrollbar(tree_view_frame, orient=VERTICAL)
    course_tree = ttk.Treeview(tree_view_frame,
                                       columns=(
                                           "COURSE ID", "COURSE NAME","COURSE DURATION","
COURSE CREDIT",
                                           "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=course_tree.xview)
    scroll_y.config(command=course_tree.yview)

    # ==========================TreeView Heading====================
    course_tree.heading("COURSE ID", text="COURSE ID")
    course_tree.heading("COURSE NAME", text="COURSE NAME")
    course_tree.heading("COURSE DURATION", text="COURSE DURATION")
    course_tree.heading("COURSE CREDIT", text="COURSE CREDIT")
    course_tree.heading("REGISTRATION DATE", text="REGISTRATION DATE")
    course_tree["show"] = "headings"

    # ==========================TreeView Column====================
    course_tree.column("COURSE ID", width=50)
    course_tree.column("COURSE NAME", width=150)
    course_tree.column("COURSE DURATION", width=100)
    course_tree.column("COURSE CREDIT", width=100)
    course_tree.column("REGISTRATION DATE", width=100)
    course_tree.pack(fill=BOTH, expand=1)

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




    #root.mainloop()
```