

```

1 from distutils import command
2 from tkinter import *
3 from tkinter import ttk
4 from typing_extensions import _AnnotatedAlias
5 from PIL import Image, ImageTk
6 import os
7 import pickle
8 import mysql.connector as sql
9 from tkinter import messagebox
10 from datetime import date
11 from datetime import time
12 from datetime import *
13 import requests
14 from bs4 import BeautifulSoup
15 import time
16 import dashboard
17 import csv
18 import regist_batch
19
20 def batch_screen():
21     def click_go_to_dashboard():
22         """returns AdminDashboard class when clicked go to dashboard"""
23         dashboard.dashboard()
24         root.withdraw()
25
26     def click_delete_batch():
27         """when clicked delete batch, it will require to select the batch and after
28 selecting and
29 performing the delete method, it will ask the admin either they are sure they want
30 to delete that batch
31 or not if yes then batch containing that id in batch table is deleted."""
32         try:
33             #obj_batch_registration_database = Model_class.batch_registration.GetDatabase('
34             use cms;')
35             #db_connection.create(obj_batch_registration_database.get_database())
36             a=load_data()
37             host=a[0]
38             username = a[2]
39             password = a[3]
40             port=a[1]
41
42             spec=sql.connect(host=host,user=username,password=password,port=port,database="
43             sms")
44             mycur=spec.cursor()
45
46             batch_view_content = batch_tree.focus()
47             batch_view_items = batch_tree.item(batch_view_content)
48             batch_view_values = batch_view_items['values'][0]
49             ask = messagebox.askyesno("Warning",
50                                     f"Are you sure you want to delete batch having id {
51 batch_view_values}")
52
53             if ask is True:
54                 query = f"delete from batch where batch_id={batch_view_values};"
55                 #db_connection.delete(query, (batch_view_values,))
56                 mycur.execute(query)
57                 spec.commit()
58                 messagebox.showinfo("Success", f" batch id {batch_view_values} deleted
59 Successfully")
60
61             click_view_all()
62         else:
63             pass

```

```

58
59         except BaseException as msg:
60             print(msg)
61             messagebox.showerror("Error",
62                                     "There is some error deleting the data\n Make sure you
have Selected the data")
63
64
65     def up_sec():
66
67         def upd_sec():
68             """updates the data of batch from entry fields"""
69             try:
70                 #obj_batch_database = Model_class.batch_registration.GetDatabase('use cms;')
71                 #db_connection.create(obj_batch_database.get_database())
72
73                 #get_id = .get_id
74                 data_id = get_id[0]
75                 # print(data_id)
76                 a=load_data()
77                 host=a[0]
78                 username = a[2]
79                 password = a[3]
80                 port=a[1]
81
82                 spec=sql.connect(host=host,user=username,password=password,port=port,database=
"sms")
83                 mycur=spec.cursor()
84                 #obj_batch_database = Model_class.batch_registration.BatchRegistration(
batch_name_entry.get(),
85                                                                                                     #
batch_year_entry.get(),
86                                                                                                     #
batch_intake_combo.get(),
87                                                                                                     #
reg_date)
88                 query = f"update batch set batch_name='{batch_name_entry.get()}',batch_year='{
batch_year_entry.get()}', batch_intake='{batch_intake_combo.get()}'" \
89                     f" where batch_id={data_id}"
90
91                 mycur.execute(query)
92                 spec.commit()
93                 #values = (obj_batch_database.get_name(), obj_batch_database.get_year(),
94                     #obj_batch_database.get_intake(), data_id)
95                 click_view_all()
96                 #db_connection.update(query, values)
97
98                 ask = messagebox.askyesnocancel("Success",
99                                                 f"Data having \n Batch Name={batch_name_entry.
get()} \n Updated Successfully\n"
100                                                 f"Do you want to Go Batch Dashboard")
101
102                 if ask is True:
103                     pass
104
105             except BaseException as msg:
106                 print(msg)
107                 messagebox.showerror("Error", f"Error due to{msg}")
108
109     def tree_event_handle():
110         try:
111             #obj_student_database = Model_class.student_registration.GetDatabase('use cms
;')

```

```

112         #db_connection.create(obj_student_database.get_database())
113
114         tree_view_content = batch_tree.focus()
115         tree_view_items = batch_tree.item(tree_view_content)
116         # print(tree_view_items)
117         tree_view_values = tree_view_items['values']
118         tree_view_id = tree_view_items['values'][0]
119         # print(tree_view_id)
120         list_of_tree.clear()
121         get_id.clear()
122         get_id.append(tree_view_id)
123         for i in tree_view_values:
124             list_of_tree.append(i)
125
126         ask = messagebox.askyesno("Confirm",
127                                   f"Do you want to Update Student having id {
tree_view_id}")
128         #if ask is True:
129             #update_sec()
130
131         except BaseException as msg:
132             print(msg)
133             messagebox.showerror("Error",
134                                   "There is some error updating the data\n Make sure you
have Selected the data")
135
136         tree_event_handle()
137
138
139
140         global batch_name_entry, batch_year_entry, batch_intake_combo
141
142         def click_clear_button():
143             batch_name_entry.delete(0, END)
144             batch_year_entry.delete(0, END)
145             batch_intake_combo.current(0)
146
147
148
149         def update_sec():
150
151             global batch_name_entry, batch_year_entry, batch_intake_combo
152
153             root = Toplevel()
154             root.title('BATCH REGISTRATION FORM - COLLEGE MANAGEMENT SYSTEM')
155             root.geometry('1067x600')
156             root.config(bg="#f29844")
157             root.resizable(False, False)
158
159
160             # =====Backend connection=====
161             #db_connection = Backend.connection.DatabaseConnection()
162
163             # creating frame for Register
164             # img = img
165             # dummylabel = Label(root, image=img)
166             # dummylabel.place(x=30, y=30)
167
168             reg_frame = Frame(root, bg="#ffffff", width=1000, height=560)
169             reg_frame.place(x=30, y=30)
170
171
172

```

```

173 heading = Label(reg_frame, text="Batch Registration Form", font=('yu gothic ui',
174 20, "bold"), bg="white",
175 fg='black',
176 bd=5,
177 relief=FLAT)
178 heading.place(x=200, y=0, width=600)
179
180
181 batch_frame = LabelFrame(reg_frame, text="Batch Details", bg="white", fg="#4f4e4d"
182 , height=380,
183 width=800, borderwidth=2.4,
184 font=("yu gothic ui", 13, "bold"))
185 batch_frame.config(highlightbackground="red")
186 batch_frame.place(x=100, y=90)
187
188 # =====
189 # =====batch Name=====
190 # =====
191 batch_name_label = Label(batch_frame, text="Batch Name ", bg="white", fg="#4f4e4d"
192 ,
193 font=("yu gothic ui", 13, "bold"))
194 batch_name_label.place(x=160, y=65)
195 batch_name_entry = Entry(root, highlightthickness=0, relief=FLAT, bg="white", fg=
196 "#6b6a69",
197 font=("yu gothic ui semibold", 12))
198 batch_name_entry.place(x=400, y=212, width=345) # trebuchet ms
199 batch_name_line = Canvas(root, width=345, height=1.5, bg="#bdb9b1",
200 highlightthickness=0)
201 batch_name_line.place(x=400, y=234)
202
203 # =====
204 # =====batch YEAR =====
205 # =====
206 date = time.strftime("%Y")
207 batch_year_label = Label(batch_frame, text="Batch Year ", bg="white", fg="#4f4e4d"
208 ,
209 font=("yu gothic ui", 13, "bold"))
210 batch_year_label.place(x=160, y=115)
211 batch_year_entry = Entry(root, highlightthickness=0, relief=FLAT, bg="white", fg=
212 "#6b6a69",
213 font=("yu gothic ui semibold", 12))
214 batch_year_entry.place(x=390, y=262, width=355) # trebuchet ms
215 batch_year_line = Canvas(root, width=355, height=1.5, bg="#bdb9b1",
216 highlightthickness=0)
217 batch_year_line.place(x=390, y=284)
218 batch_year_entry.insert(0, date)
219
220 # =====
221 # =====batch Intake=====
222 # =====
223 root.option_add("*TCombobox*Listbox*Foreground", '#f29844')
224 batch_intake_label = Label(batch_frame, text="Batch Intake ", bg="white", fg="#
225 4f4e4d",
226 font=("yu gothic ui", 13, "bold"))
227 batch_intake_label.place(x=160, y=165)

```

```

227
228         batch_intake_combo = ttk.Combobox(batch_frame, font=('yu gothic ui semibold', 12,
'bold'),
229                                           state='readonly',
230                                           width=35)
231         batch_intake_combo['values'] = ("January", "February", "March", "April", "May", "
June", "July", "August",
232                                       "September", "October", "November", "
December")
233         batch_intake_combo.current(0)
234         batch_intake_combo.place(x=270, y=167)
235         # batch_intake_line.place(x=410, y=424)
236
237         reg_date = time.strftime("%Y/%m/%d")
238
239
240         # =====
241         # =====Register options
=====
242         # =====
243         submit_img = ImageTk.PhotoImage(file='Pics\\submit.png')
244         submit = Button(batch_frame, image=submit_img,
245                        font=("yu gothic ui", 13, "bold"), relief=FLAT,
activebackground="white"
246                        , borderwidth=0, background="white", cursor="hand2",
command=upd_sec)
247         submit.image = submit_img
248         submit.place(x=90, y=267)
249
250         clear_img = ImageTk.PhotoImage(file='Pics\\clear.png')
251         clear_button = Button(batch_frame, image=clear_img,
252                             font=("yu gothic ui", 13, "bold"), relief=FLAT,
activebackground="white"
253                             , borderwidth=0, background="white", cursor="hand2",
254                             command=click_clear_button)
255         clear_button.image = clear_img
256         clear_button.place(x=250, y=270)
257
258         back_img = ImageTk.PhotoImage(file='Pics\\back.png')
259         back_button = Button(batch_frame, image=back_img,
260                             font=("yu gothic ui", 13, "bold"), relief=FLAT,
activebackground="white"
261                             , borderwidth=0, background="white", cursor="hand2")
262         back_button.image = back_img
263         back_button.place(x=410, y=270)
264
265         exit_img = ImageTk.PhotoImage(file='Pics\\exit.png')
266         exit_button = Button(batch_frame, image=exit_img,
267                             font=("yu gothic ui", 13, "bold"), relief=FLAT,
activebackground="white"
268                             , borderwidth=0, background="white", cursor="hand2",
command=exit)
269         exit_button.image = exit_img
270         exit_button.place(x=570, y=270)
271
272         a = list_of_tree
273
274         try:
275             batch_name_entry.insert(0, a[1])
276             batch_year_entry.delete(0,END)
277             batch_year_entry.insert(0, a[2])
278             batch_intake_combo.insert(0, a[3])
279

```

```

280         except IndexError as msg:
281             print(msg)
282
283         update_sec()
284
285         root = Toplevel()
286         root.geometry("1067x600")
287         root.title("Batch Management Dashboard - College Management System")
288         #root.iconbitmap('images\\logo.ico')
289         root.resizable(False, False)
290
291         list_of_tree = []
292         get_id = []
293
294         manage_student_frame_r = Image.open('Pics\\student_frame.png').resize((1067,600),Image.
ANTIALIAS)
295         manage_student_frame = ImageTk.PhotoImage(manage_student_frame_r)
296         image_panel = Label(root, image=manage_student_frame)
297         image_panel.image = manage_student_frame
298         image_panel.pack(fill='both', expand='yes')
299
300         #db_connection = Backend.connection.DatabaseConnection()
301
302
303
304         heading = Label(root, text="Batch Management Dashboard", font=('yu gothic ui', 20, "bold"
), bg="white",
305                             fg='black',
306                             bd=5,
307                             relief=FLAT)
308         heading.place(x=420, y=26, width=640)
309         #slider()
310         #heading_color()
311
312
313         # =====
314         # =====Left frame =====
315         # =====
316
317         #left frame
318         left_view_frame = Frame(root, bg="white")
319         left_view_frame.place(x=35, y=89, height=470, width=250)
320
321
322         #tree view frame
323         tree_view_frame = Frame(root, bg="white")
324         tree_view_frame.place(x=301, y=90, height=473, width=730)
325
326         # =====
327         # =====frame for personal credentials =====
328         # =====
329
330         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"))
331         personal_frame.config(highlightbackground="red")
332         personal_frame.place(x=5, y=8)
333
334         # =====
335         # =====Add Batch button=====
336         # =====
337         add_batch_r = Image.open('Pics\\add_batch.png').resize((225,25),Image.ANTIALIAS)
338         add_batch = ImageTk.PhotoImage(add_batch_r)
339         add_batch_button = Button(personal_frame, image=add_batch, relief=FLAT, borderwidth=0,

```

```

340         activebackground="white", bg="white", cursor="hand2",
command=regist_batch.regist_batch)
341     add_batch_button.image = add_batch
342     add_batch_button.place(x=8, y=100)
343     # add_student_button.place(x=36, y=295)
344
345     # =====
346     # =====Update batch button=====
347     # =====
348
349     update_batch_r = Image.open('Pics\\update_batch.png').resize((225,25),Image.ANTIALIAS)
350     update_batch = ImageTk.PhotoImage(update_batch_r)
351     update_batch_button = Button(personal_frame, image=update_batch, relief=FLAT, borderwidth=
0,
352         activebackground="white", bg="white", cursor="hand2",
command=up_sec)
353     update_batch_button.image = update_batch
354     update_batch_button.place(x=8, y=165)
355     # update_student_button.place(x=36, y=355)
356
357     # =====
358     # =====Delete batch button=====
359     # =====
360
361     delete_batch_r = Image.open('Pics\\delete_batch.png').resize((225,25),Image.ANTIALIAS)
362     delete_batch = ImageTk.PhotoImage(delete_batch_r)
363     delete_batch_button = Button(personal_frame, image=delete_batch, relief=FLAT, borderwidth=
0,
364         activebackground="white", bg="white", cursor="hand2",
command=click_delete_batch)
365     delete_batch_button.image = delete_batch
366     delete_batch_button.place(x=8, y=230)
367
368     # delete_student_button.place(x=36, y=405)
369
370     # =====
371     # =====Goto Main dashboard button=====
372     # =====
373
374     goto_dashboard_r = Image.open('Pics\\goto_dashboard.png').resize((225,25),Image.ANTIALIAS)
375     goto_dashboard = ImageTk.PhotoImage (goto_dashboard_r)
376     goto_dashboard_button = Button(personal_frame, image=goto_dashboard, relief=FLAT,
borderwidth=0,activebackground="white", bg="white", cursor="hand2",command=
click_go_to_dashboard)
377     goto_dashboard_button.image = goto_dashboard
378     goto_dashboard_button.place(x=5, y=295)
379
380     # =====
381     # =====Starting Tree View=====
382     # =====
383
384     def load_data():
385         f=open("Credentials.csv","r")
386         s=csv.reader(f,delimiter="-")
387         d=[]
388         for i in s:
389             d.append(i)
390         a=d[:-1]
391         return (a[0])
392
393     def click_view_all():
394         """it will show all the data contains on the batch table of cms database, when
clicked by default this method

```

```

395         is called while initializing the class ManageBatch. Exception is handled to avoid
run time error which may
396         cause by user.""
397     try:
398         #obj_student_database = Model_class.student_registration.GetDatabase('use cms
;')
399         #db_connection.create(obj_student_database.get_database())
400         a=load_data()
401         host=a[0]
402         username = a[2]
403         password = a[3]
404         port=a[1]
405
406         spec=sql.connect(host=host,user=username,password=password,port=port,database=
"sms")
407         mycur=spec.cursor()
408         query = "select * from batch;"
409         mycur.execute(query)
410         data = mycur.fetchall()
411         # print(data)
412         batch_tree.delete(*batch_tree.get_children())
413         for values in data:
414             data_list = [values[0], values[1], values[2], values[3], values[4]]
415             print(data_list)
416             batch_tree.insert('', END, values=data_list)
417
418     except BaseException as msg:
419         print(msg)
420
421
422
423     style = ttk.Style()
424     style.configure("Treeview.Heading", font=('yu gothic ui', 10, "bold"), foreground="red")
425
426     scroll_x = Scrollbar(tree_view_frame, orient=HORIZONTAL)
427     scroll_y = Scrollbar(tree_view_frame, orient=VERTICAL)
428     batch_tree = ttk.Treeview(tree_view_frame,
429                               columns=(
430                                   "BATCH ID", "BATCH NAME", "BATCH YEAR", "BATCH INTAKE"
, "REGISTRATION DATE"),
431                                   xscrollcommand=scroll_x.set, yscrollcommand=scroll_y.set)
432     scroll_x.pack(side=BOTTOM, fill=X)
433     scroll_y.pack(side=RIGHT, fill=Y)
434     scroll_x.config(command=batch_tree.xview)
435     scroll_y.config(command=batch_tree.yview)
436
437     # =====TreeView Heading=====
438     batch_tree.heading("BATCH ID", text="BATCH ID")
439     batch_tree.heading("BATCH NAME", text="BATCH NAME")
440     batch_tree.heading("BATCH YEAR", text="BATCH YEAR")
441     batch_tree.heading("BATCH INTAKE", text="BATCH INTAKE")
442     batch_tree.heading("REGISTRATION DATE", text="REGISTRATION DATE")
443     batch_tree["show"] = "headings"
444
445     # =====TreeView Column=====
446     batch_tree.column("BATCH ID", width=50)
447     batch_tree.column("BATCH NAME", width=150)
448     batch_tree.column("BATCH YEAR", width=100)
449     batch_tree.column("BATCH INTAKE", width=100)
450     batch_tree.column("REGISTRATION DATE", width=100)
451     batch_tree.pack(fill=BOTH, expand=1)
452
453     #batch_tree.bind("<Delete>", click_delete_key)

```



```
454     #batch_tree.bind("<Button-3>", do_popup)
455     #batch_tree.bind("<Double-Button-1>", tree_double_click)
456     #batch_tree.bind("<Return>", tree_double_click)
457     #search_start()
458     #search_by.current(0)
459
460     click_view_all()
461
462
463     #root.mainloop()
464
465 if __name__ == "__main__":
466     batch_screen()
```