

```

1 from distutils import command
2 from tkinter import *
3 from tkinter import ttk
4 from PIL import Image,ImageTk
5 import os
6 import pickle
7 import mysql.connector as sql
8 from tkinter import messagebox
9 from datetime import date
10 from datetime import time
11 from datetime import *
12 import requests
13 from bs4 import BeautifulSoup
14 import time
15 import user_inter
16 import csv
17 import course_screen
18
19
20
21 def sec_reg():
22
23     def load_data():
24         f=open("Credentials.csv","r")
25         s=csv.reader(f,delimiter="-")
26         d=[]
27         for i in s:
28             d.append(i)
29         a=d[::-1]
30         return (a[0])
31
32     def click_clear_button():
33         section_name_entry.delete(0, END)
34         section_code_entry.delete(0, END)
35         section_capacity_entry.delete(0,END)
36
37
38
39     def click_enter_submit():
40         validation()
41
42
43     def validation():
44         """this will validate if the section code and name of entry fields are already in
45         database table named
46         section or not if return True, error message is thrown displaying section code/name
47         already exists"""
48         try:
49             #obj_section_database = Model_class.section_registration.GetDatabase('use cms;')
50             #self.db_connection.create(obj_section_database.get_database())
51             a=load_data()
52             host=a[0]
53             username = a[2]
54             password = a[3]
55             port=a[1]
56
57             spec=sql.connect(host=host,user=username,password=password,port=port,database="sms"
58 )
59             mycur=spec.cursor()
60             query = "select * from section;"
61             mycur.execute(query)
62             data = mycur.fetchall()
63             # print(data)

```

```

61         code_list = []
62         name_list = []
63         for values in data:
64             code_data_list = values[1]
65             code_list.append(code_data_list)
66             name_data_list = values[2]
67             name_list.append(name_data_list)
68
69     except BaseException as msg:
70         print(msg)
71
72     if section_code_entry.get() == "" or section_name_entry.get() == "" or \
73        section_capacity_entry.get() == "" :
74         messagebox.showwarning("Warning", "All Fields are Required\n Please fill all
required fields")
75
76     elif section_code_entry.get() in code_list:
77         messagebox.showerror("Already Exists", f"{section_code_entry.get()} Section code
Already Exists")
78
79     elif section_name_entry.get() in name_list:
80         messagebox.showerror("Already Exists", f"{section_name_entry.get()} Section
Already Exists")
81
82     else:
83         click_submit()
84
85     def back():
86         root.destroy()
87
88     def click_submit():
89         """initialize when click submit button, which will take data from entry box
90         and insert those data into student table after successful validation of those data"""
91         try:
92             #obj_section_database = Model_class.section_registration.GetDatabase('use cms;')
93             #self.db_connection.create(obj_section_database.get_database())
94
95             a=load_data()
96             host=a[0]
97             username = a[2]
98             password = a[3]
99             port=a[1]
100
101             spec=sql.connect(host=host,user=username,password=password,port=port,database="sms
")
102             mycur=spec.cursor()
103
104             #obj_section_database = Model_class.section_registration.SectionRegistration(self.
section_code_entry.get(),
105                                                                                                     #self.
section_name_entry.get(),
106                                                                                                     #self.
section_capacity_entry.get(),
107                                                                                                     #self.
reg_date)
108
109             query = f"insert into section (section_code,section_name,section_capacity,reg_date
) values ('{section_code_entry.get()}', '{section_name_entry.get()}', '{section_capacity_entry.
get()}', '{reg_date}');"
110             mycur.execute(query)
111             spec.commit()
112             #values = (obj_section_database.get_code(),obj_section_database.get_name(),
113                       #obj_section_database.get_capacity(),obj_section_database.get_reg_date

```

```

113 ())
114     # print(values)
115     #self.db_connection.insert(query, values)
116     # print(values)
117     messagebox.showinfo("Success", f"Admin Data inserted Successfully\n Section code={
section_code_entry.get()},\n "
118                                     f"Section name={section_name_entry.get()}")
119
120     except BaseException as msg:
121         print(msg)
122         messagebox.showerror("Error", "There is some error Submitting Credentials ")
123
124
125
126 root = Toplevel()
127 root.title('SECTION REGISTRATION FORM - COLLEGE MANAGEMENT SYSTEM')
128 root.geometry('1067x600')
129 root.config(bg="#f29844")
130
131 # =====Backend connection=====
132 #db_connection = Backend.connection.DatabaseConnection()
133
134 reg_frame = Frame(root, bg="ffffff", width=1000, height=560)
135 reg_frame.place(x=30, y=30)
136
137
138
139 heading = Label(reg_frame, text="Section Registration Form", font=('yu gothic ui', 30, "
bold"), bg="white",
140                 fg='black',
141                 bd=5,
142                 relief=FLAT)
143 heading.place(x=200, y=0, width=600)
144
145
146 section_frame = LabelFrame(reg_frame, text="Section Details", bg="white", fg="#4f4e4d",
height=380,
147                             width=800, borderwidth=2.4,
148                             font=("yu gothic ui", 13, "bold"))
149 section_frame.config(highlightbackground="red")
150 section_frame.place(x=100, y=90)
151
152 # =====
153 # =====Key Bindings=====
154 # =====
155
156 #root.bind("<Return>", click_enter_submit)
157
158 # # =====
159 # # =====Section ID label=====
160 # # =====
161
162 # =====
163 # =====Subject Code =====
164 # =====
165
166 section_code_label = Label(section_frame, text="Section Code ", bg="white", fg="#4f4e4d",
167                             font=("yu gothic ui", 13, "bold"))
168 section_code_label.place(x=160, y=50)
169
170 section_code_entry = Entry(root, highlightthickness=0, relief=FLAT, bg="white", fg="#
6b6a69",
171                             font=("yu gothic ui semibold", 12))

```

```

172 section_code_entry.place(x=405, y=197, width=340) # trebuchet ms
173
174 section_code_line = Canvas(root, width=340, height=1.5, bg="#bdb9b1", highlightthickness=0
175 )
176 section_code_line.place(x=405, y=219)
177
178 # =====
179 # =====Section Name=====
180 # =====
181 section_name_label = Label(section_frame, text="Section Name ", bg="white", fg="#4f4e4d",
182                             font=("yu gothic ui", 13, "bold"))
183 section_name_label.place(x=160, y=100)
184
185 section_name_entry = Entry(root, highlightthickness=0, relief=FLAT, bg="white", fg="#
186 6b6a69",
187                             font=("yu gothic ui semibold", 12))
188 section_name_entry.place(x=410, y=247, width=335) # trebuchet ms
189
190 section_name_line = Canvas(root, width=335, height=1.5, bg="#bdb9b1", highlightthickness=0
191 )
192 section_name_line.place(x=410, y=269)
193
194 # =====
195 # =====Section capacity=====
196 # =====
197 root.option_add("*TCombobox*Listbox*Foreground", '#f29844')
198
199 section_capacity_label = Label(section_frame, text="Section Capacity ", bg="white", fg="#
200 4f4e4d",
201                             font=("yu gothic ui", 13, "bold"))
202 section_capacity_label.place(x=160, y=150)
203
204 section_capacity_entry = Entry(root, highlightthickness=0, relief=FLAT, bg="white", fg="#
205 6b6a69",
206                             font=("yu gothic ui semibold", 12))
207 section_capacity_entry.place(x=430, y=297, width=315) # trebuchet ms
208
209 section_capacity_line = Canvas(root, width=315, height=1.5, bg="#bdb9b1",
210 highlightthickness=0)
211 section_capacity_line.place(x=430, y=319)
212
213 reg_date = time.strftime("%Y/%m/%d")
214
215 # =====
216 # =====Register options=====
217 # =====
218 submit_img = ImageTk.PhotoImage(file='Pics\\submit.png')
219 submit = Button(section_frame, image=submit_img,
220                 font=("yu gothic ui", 13, "bold"), relief=FLAT, activebackground="
221 white"
222                 , borderwidth=0, background="white", cursor="hand2", command=
223 click_enter_submit)
224 submit.image = submit_img
225 submit.place(x=90, y=267)
226
227
228 clear_img = ImageTk.PhotoImage(file='Pics\\clear.png')
229 clear_button = Button(section_frame, image=clear_img,
230                       font=("yu gothic ui", 13, "bold"), relief=FLAT,
231                       activebackground="white"

```

```
226         , borderwidth=0, background="white", cursor="hand2",
227         command=click_clear_button)
228     clear_button.image = clear_img
229     clear_button.place(x=250, y=270)
230
231
232     back_img = ImageTk.PhotoImage(file='Pics\\back.png')
233     back_button = Button(section_frame, image=back_img,
234         font=("yu gothic ui", 13, "bold"), relief=FLAT,
235     activebackground="white"
236         , borderwidth=0, background="white", cursor="hand2", command=
237     back)
238     back_button.image = back_img
239     back_button.place(x=410, y=270)
240
241
242     exit_img = ImageTk.PhotoImage(file='Pics\\exit.png')
243     exit_button = Button(section_frame, image=exit_img,
244         font=("yu gothic ui", 13, "bold"), relief=FLAT,
245     activebackground="white"
246         , borderwidth=0, background="white", cursor="hand2", command=
247     exit)
248     exit_button.image = exit_img
249     exit_button.place(x=570, y=270)
250
251     #root.mainloop()
252
253 if __name__ == "__main__":
254     sec_reg()
```