```python
from tkinter import *
from tkinter import ttk, filedialog, messagebox, PhotoImage
import pandas as pd
from PIL import Image, ImageDraw, ImageFont
from datetime import date
import subprocess

# Creating the main tkinter object
root = Tk()
root.geometry("1080x600")
root.title(
    'K.V. No.1 Vadodara Student Tc System - developed by [Aayush & Omkar]')

# icon================================================
logophoto = PhotoImage(file='./assets/logo.png')
root.wm_iconphoto(False, logophoto)

# ===============frame 2==============================
sheet_frame = Frame(bd=4, relief=RIDGE)

# creating A Tree View
sheet_tree = ttk.Treeview(sheet_frame)

# placing the tree
sheet_frame.place(relx=0.35, y=0, relheight=0.97, relwidth=0.62)

# school name variable
school_name = "No. 1, Vadodara"

# Global FIle open/closed variable--------------
fileStatus = False
pdFileObj = None
fileName = None


# A global Reason Button
reason_button = None


# Function for generating the TC form image
def GenerateTcFormImg(student, reason):

    form_img = Image.open("./assets/formimg.jpg")
    form_img_draw = ImageDraw.Draw(form_img)
    myFont = ImageFont.truetype('./assets/Arial.ttf', 20)
    s_address = student['local_address']

    if len(s_address) > 92:
        s_address = str(s_address[:92]) + "..."

    # School Name
    form_img_draw.text((555, 115), school_name, fill=(0,0,0),
font=ImageFont.truetype('./assets/Arial.ttf', 28))

    # Date of Application
    form_img_draw.text((650, 260), str(date.today()),
                       fill=(0, 0, 0), font=myFont)

    # Student Name
    form_img_draw.text(
        (650, 293), student['std_name'], fill=(0, 0, 0), font=myFont)
```

```
61
62      # Class-Section (with year)
63      form_img_draw.text(
64          (650, 327), f"{student['class']}-{student['section']}  ({student['session']})",
    fill=(0, 0, 0), font=myFont)
65
66      # Father's Name 33diff
67      form_img_draw.text(
68          (650, 360), student['f_name'], fill=(0, 0, 0), font=myFont)
69
70      # Mother's name
71      form_img_draw.text(
72          (650, 393), student['m_name'], fill=(0, 0, 0), font=myFont)
73
74      # Local Address
75      form_img_draw.text((650, 433), s_address, fill=(
76          0, 0, 0), font=ImageFont.truetype('./assets/Arial.ttf', 12))
77
78      # Admission Number
79      form_img_draw.text((380, 1272), str(
80          student['adm_no']), fill=(0, 0, 0), font=myFont)
81
82      subs = ""
83      for subject in student['subjects']:
84          subs = subs + subject + ", "
85      subs = subs[:-2]
86
87      # Subjects
88      form_img_draw.text((650, 521), subs, fill=(0, 0, 0),
89                         font=ImageFont.truetype('./assets/Arial.ttf', 14))
90
91      # Reason to leave
92      form_img_draw.text((650, 465), reason,
93                         fill=(0, 0, 0), font=ImageFont.truetype('./assets/Arial.ttf', 12))
94
95      img_name = f"{student['adm_no']}_{student['std_name']}_tc.png"
96      form_img.save(img_name, format='png')
97
98      #using subprocess module to open the saved file
99      subprocess.call(img_name, shell=True)
100
101
102 # Function for generating a student details dictionary
103 def generateFormDetails(st_details) -> dict:
104     subs = []
105
106     # Running a for loop To Append Subject List to A dict
107     for i in range(9, 14):
108         # if Sub is Not None Append it
109         if i == "nan":
110             continue
111
112         # if sub is None Continue the loop ignoring that Sub
113         else:
114             subs.append(st_details[i])
115
116     # creating a Dict that stores Data extracted from the Tkinter Tree  Object
117     st_details_dict = {
118         "adm_no": st_details[0],
119         "std_name": st_details[1],
120         "f_name": st_details[2],
```

```
121         "m_name": st_details[3],
122         "class": int(st_details[4][:-2]),
123         "section": st_details[5],
124         "session": st_details[6],
125         "local_address": st_details[7],
126         "subjects": subs
127     }
128     return st_details_dict
129 
130 # A CallBack Function that helps with calling the Opps functions in tkinter
131 
132 
133 def FormGeneratorCallback(reason):
134     treeFocousData = sheet_tree.item(sheet_tree.focus())
135     st_details = treeFocousData["values"]
136 
137     # calling a function from TcApp module
138     student = generateFormDetails(st_details)
139     GenerateTcFormImg(student=student, reason=reason)
140 
141 
142 def popUpButton():
143     top = Toplevel(root)
144     top.title("Provide a reason For Your tc")
145     # icon=========================================
146     logophoto = PhotoImage(file='./assets/logo.png')
147     top.wm_iconphoto(False, logophoto)
148     top.geometry('350x150')
149 
150     top_heading = Label(top, text="Enter the reason:", font=12)
151     top_heading.place(relx=0.35, rely=0.2)
152 
153     top_entry = Entry(top)
154     top_entry.focus()
155     top_entry.place(relx=0.1, rely=0.4, relwidth=0.8)
156 
157     top_button = Button(top, text="Submit", width=10, height=1,
158                         font=10, command=lambda: kill_main(top_entry))
159     top_button.place(relx=0.35, rely=0.6)
160 
161     def kill_main(top_entry):
162         FormGeneratorCallback(top_entry.get())
163         top.destroy()
164         top.update()
165 
166 
167 # Function for file open dialong for opening the excel file
168 def file_open():
169     global fileStatus
170     global pdFileObj
171     global fileName
172 
173     filename = filedialog.askopenfile(
174         initialdir="./",
175         title="Select students data sheet",
176         filetypes=(('Excel File', '*.xlsx'),
177                    ('CSV File', '*.csv'), ("All Files", "*.*"))
178     )
179 
180     print(str(filename))
181     if filename:
```

```
182         try:
183             df = pd.read_excel(filename.name)
184             pdFileObj = df
185 
186         except Exception as e:
187             print(e)
188 
189         # Clearing tree view if already any
190         clear_tree()
191 
192         # Setting up new treeview
193         sheet_tree["column"] = list(df.columns)
194         sheet_tree["show"] = "headings"
195 
196         # Iterating through column list
197         for col in sheet_tree["column"]:
198             sheet_tree.heading(col, text=col)
199 
200         # Putting data in treeview
201         df_rows = df.to_numpy().tolist()
202         for row in df_rows:
203             sheet_tree.insert("", "end", values=row)
204 
205         # adding scroll bars---------
206         if fileStatus == False:
207             sheet_scrolly = Scrollbar(sheet_frame)
208             sheet_scrollx = Scrollbar(sheet_frame, orient=HORIZONTAL)
209 
210             sheet_scrollx.pack(side=BOTTOM, fill=X)
211             sheet_scrolly.pack(side=RIGHT, fill=Y)
212 
213             # packing the scroll bar ------------------------------
214             sheet_tree.pack()
215 
216             # internal configs for scrollbars
217             sheet_scrolly.config(command=sheet_tree.yview)
218             sheet_scrollx.config(command=sheet_tree.xview)
219 
220             sheet_tree.config(yscrollcommand=sheet_scrolly.set)
221             sheet_tree.config(xscrollcommand=sheet_scrollx.set)
222             sheet_tree.config(selectmode=BROWSE)
223 
224             gen_button.config(state=ACTIVE)
225 
226         else:
227             pass
228 
229         sheet_tree.place(x=0, y=0, relheight=1, relwidth=1)
230     fileStatus = True
231 
232 # showing the file again
233 
234 
235 def show_file_again():
236     global fileStatus
237     global pdFileObj
238 
239     df = pdFileObj
240 
241     # Clearing tree view if already any
242     clear_tree()
```

```
243
244     # Setting up new treeview
245     sheet_tree["column"] = list(df.columns)
246     sheet_tree["show"] = "headings"
247
248     # Iterating through column list
249     for col in sheet_tree["column"]:
250         sheet_tree.heading(col, text=col)
251
252     # Putting data in treeview
253     df_rows = df.to_numpy().tolist()
254     for row in df_rows:
255         sheet_tree.insert("", "end", values=row)
256
257     else:
258         pass
259
260
261 # Some helper functions
262 def clear_tree():
263     sheet_tree.delete(*sheet_tree.get_children())
264
265
266 #helper function that focous on a tkinter tree row
267 def get_selected():
268     row = sheet_tree.item(sheet_tree.focus())
269     print(row)
270
271
272 #Searches for student details in the excel sheet provided by the user
273 def searchStudent():
274     if fileStatus == False:
275         messagebox.showerror(
276             "Error", "No Student Record File is Open\nPlease open a Student record file by
    using the open button")
277         return
278
279     # Clearing tree view if already any
280     clear_tree()
281
282     # Setting up new treeview
283     sheet_tree["column"] = list(pdFileObj.columns)
284     sheet_tree["show"] = "headings"
285
286     # Iterating through column list
287     for col in sheet_tree["column"]:
288         sheet_tree.heading(col, text=col)
289
290     # Putting data in treeview
291     df_rows = pdFileObj.to_numpy().tolist()
292     rowFound = False
293     for row in df_rows:
294         if row[0] == int(search_entry.get()):
295             sheet_tree.insert("", "end", values=row)
296             rowFound = True
297
298     if rowFound == None:
299         messagebox.showerror(
300             "Error", "No Student Data for Provided Admission Number found.\nPlease Check
    the data file or Admission Number Provided")
301         return
302
```

```
303     print(search_entry.get())
304
305
306 # ==================frame1=============================================
307 # SETTING THE BUTTONS IN THE MENU THING
308 menu_frame = Frame(bd=4, relief=RIDGE)
309 menu_frame.place(relx=0, rely=0, relheight=0.97, relwidth=0.35)
310
311 sheet_button = Button(menu_frame, text="Open File", width=40,
312                       height=2, font=20, fg="white", bg="#0078d7", command=file_open)
313 sheet_button.place(x=0, y=0)
314
315 #
316 gen_button = Button(menu_frame, text="Generate Tc", width=40, height=2,
317                     font=20, fg="white", bg="#0078d7", command=popUpButton,
    state=DISABLED)
318 gen_button.place(x=0, y=60)
319
320
321 # search ============================================
322
323 search_heading = Label(
324     menu_frame, text="Search Student details:\n(Enter admission number)", font=12)
325 search_heading.place(x=0, y=280)
326 search_entry = Entry(menu_frame)
327 search_entry.focus()
328 search_entry.place(x=5, y=320, relwidth=0.70)
329 search_button = Button(menu_frame, text="Search", width=10, height=1,
330                        font=10, fg="white", bg="#0078d7", padx=4, command=searchStudent)
331 search_button.place(relx=0.65, y=320)
332 show_button = Button(menu_frame, text="show all", width=10, height=1,
333                      font=10, fg="white", bg="#0078d7", padx=4, command=show_file_again)
334 show_button.place(relx=0.65, y=360)
335
336
337
338 search_heading = Label(
339     menu_frame, text="Made By :-\nAayush Mishra & Omkar Mahindrakar", font=12)
340 search_heading.place(x=0, y=480)
341
342 # Running the tkinter mainloop
343 root.mainloop()
344
```