PYTHON – GUI APP WITH BACKEND DATA STORAGE

TUTORIAL & ASSIGNMENT

NOTE: This file contains a tutorial and a following assignment which is linked to the existing tutorial. This tutorial will guide you how to leverage the power of Python to create applications suitable for a particular purpose. By completing the assignment, you will be able to make an updated version of the tutorial and give it a fresh look.

The code is also explained, however you may refer to internet or ask me if something troubles you.

I recommend completing it on time. **For each delayed day, 10% marks will be reduced.**

TUTORIAL

1. Install necessary packages
    a. pip install openpyxl – to work with excel files
        i. Read more: https://pypi.org/project/openpyxl/
    b. pip install pathlib – to work with paths
        i. Read more: https://pypi.org/project/pathlib/
2. **[Optional]** However, if you are using older versions of Microsoft excel (.xls) then you should also install:
    a. pip install xlrd
        i. Read more: https://pypi.org/project/xlrd/
3. Do the necessary imports:

```
# For creating GUI applications
from tkinter import *

# For displaying message boxes
from tkinter import messagebox

# For creating themed widgets (such as Combobox)
from tkinter.ttk import Combobox

# For reading from and writing to Excel files
import openpyxl

# For creating a new Excel workbook if it doesn't exist
from openpyxl import Workbook

# For working with file paths
import pathlib
```

    a. Note that some features are by default not included in imports so we have to separately import them.

4. Now let's set the main window:

```
root = Tk()
root.title("Data Entry")
root.geometry('700x400+300+200')
root.resizable(False, False)
root.configure(bg="#326273")
file = pathlib.Path('Backened_data.xlsx')

# All the code will go here.

root.mainloop()
```

a. After running above code, you should be able to see a blank document.
b. Note that all the code must be entered at the correct position.

5. Now we will see first, if the database file (Excel sheet) is already created? If yes, the system will pass. If not, the system will create an excel file.

```
if file.exists():
    pass
else:
    file = Workbook()
    sheet = file.active
    sheet['A1'] = "Full Name"
    sheet['B1'] = "PhoneNumber"
    sheet['C1'] = "Age"
    sheet['D1'] = "Gender"
    sheet['E1'] = "Address"
    file.save('Backened_data.xlsx')
```

a. If you have placed the code in the right position and run it. You will notice and excel sheet appeared in the project folder.
b. Open the excel sheet and see how the code set's up the file.
c. NOTE: You will create a modified form later.

6. After we create our form (designing), we will be needing two buttons for the submission and for clearing the form data. So let's build two functions first which will be called when the buttons will be created later.
7. Submit() function:

```
def submit():
    # Retrieve user input data from the GUI fields
    name = nameValue.get()
    contact = contactValue.get()
    age = AgeValue.get()
    gender = gender_combobox.get()
    address = addressEntry.get(1.0, END)

    # Load the Excel workbook or create a new one if it doesn't exist
    file = openpyxl.load_workbook('Backened_Data.xlsx')
    sheet = file.active

    # Append the user's data to the next available row in the workbook
    sheet.cell(column=1, row=sheet.max_row+1, value=name)
    sheet.cell(column=2, row=sheet.max_row, value=contact)
    sheet.cell(column=3, row=sheet.max_row, value=age)
    sheet.cell(column=4, row=sheet.max_row, value=gender)
    sheet.cell(column=5, row=sheet.max_row, value=address)

    # Save the workbook with the updated data
    file.save(r'Backened_data.xlsx')

    # Display a message box to inform the user that their details have been
added successfully
    messagebox.showinfo('info', 'detail added successfully!')

    # Clear the input fields for the next entry
    nameValue.set('')
    contactValue.set('')
    AgeValue.set('')
    addressEntry.delete(1.0, END)
    # function ends here.
```

   a. The submit() function is responsible for processing the user's input data when they click the "Submit" button in the graphical user interface (GUI). It retrieves the values entered by the user for their name, contact information, age, gender, and address. After retrieving this information, it loads the existing Excel workbook named 'Backened_Data.xlsx' or creates a new one if it doesn't exist. Next, it selects the active sheet in the workbook and appends the user's data to the next available row. Once the data has been added to the workbook, a message box pops up to inform the user that their details have been successfully added. Finally, the function clears the input fields so that the user can enter new data for the next submission. Overall, the submit() function ensures the seamless handling and storage of user input data in an Excel file.

8. Clear() function:

```
def clear():
    nameValue.set('')
    contactValue.set('')
    AgeValue.set('')
    addressEntry.delete(1.0, END)
```

   a. This function clears all the fields when the button is clicked.

9. Now that we have set the basic functionality, it's time to actually design the form.
10. Let's set the icon first.

```
# icon
icon_image = PhotoImage(file="logo.png")
root.iconphoto(False, icon_image)
```

   a. Note you will be needing an image here to serve as the icon of the image. We have used (logo.png). Either save some image with that name, or may be download it. It's up to you.

11. Setting the heading

```
# heading
```

```
Label(root, text="Please fill out this entry form",
      font="arial 13", bg="#326273", fg="#fff").place(x=20, y=20)
```

   a. We use using .place here to position the elements at specific co-ordinates.

12. Adding the labels:

```
# label
Label(root, text='Name', font=23, bg="#326273", fg="#fff").place(x=50, y=100)
Label(root, text='Contact No', font=23,
      bg="#326273", fg="#fff").place(x=50, y=150)
Label(root, text='Age', font=23, bg="#326273", fg="#fff").place(x=50, y=200)
Label(root, text='Gender', font=23, bg="#326273", fg="#fff").place(x=370,
y=200)
Label(root, text='Address', font=23, bg="#326273", fg="#fff").place(x=50,
y=250)
```

   a. You may need to take a look back at the Tkinter GUI lecture if you need an understanding of this code.

13. Now set the fields to get user's input:

```
# Entry
nameValue = StringVar()
contactValue = StringVar()
AgeValue = StringVar()
nameEntry = Entry(root, textvariable=nameValue, width=45, bd=2, font=20)
contactEntry = Entry(root, textvariable=contactValue, width=45, bd=2,
font=20)
ageEntry = Entry(root, textvariable=AgeValue, width=15, bd=2, font=20)
```

   a. Here, StringVar is a special variable type provided by the Tkinter library in Python. It is used specifically for storing string values in GUI applications created with Tkinter.
   b. StringVar objects are often associated with certain GUI widgets like Entry, Label, Checkbutton, etc. This association allows the content of the widget to be linked dynamically to the value stored in the StringVar.
   c. When the value of a StringVar changes, any GUI widgets linked to it are automatically updated to reflect the new value, and vice versa. This dynamic updating simplifies the management of data between the GUI elements and the underlying application logic.
   d. Further, nameValue, contactValue, and AgeValue are instances of the StringVar class. These are special variables provided by the Tkinter module that are used to store string values.
   e. nameEntry, contactEntry, and ageEntry are instances of the Entry widget class. These are input fields where the user can enter text.
   f. root is the parent window or frame where these input fields will be placed.
   g. The textvariable parameter in each Entry widget is set to the respective StringVar variable (nameValue, contactValue, AgeValue). This means that the text entered by the user in these fields will be stored in the associated StringVar variable.
   h. The width, bd, and font parameters define the width of the entry field, the border width, and the font style and size, respectively.

14. Setting Gender Selection:

```
# gender
gender_combobox = Combobox(
    root, values=['Male', 'Female'], font='arial 14', state='r', width=14)
gender_combobox.place(x=440, y=200)
gender_combobox.set('Male')
```

a. This is how we use a Combobox from Tkinter.ttk

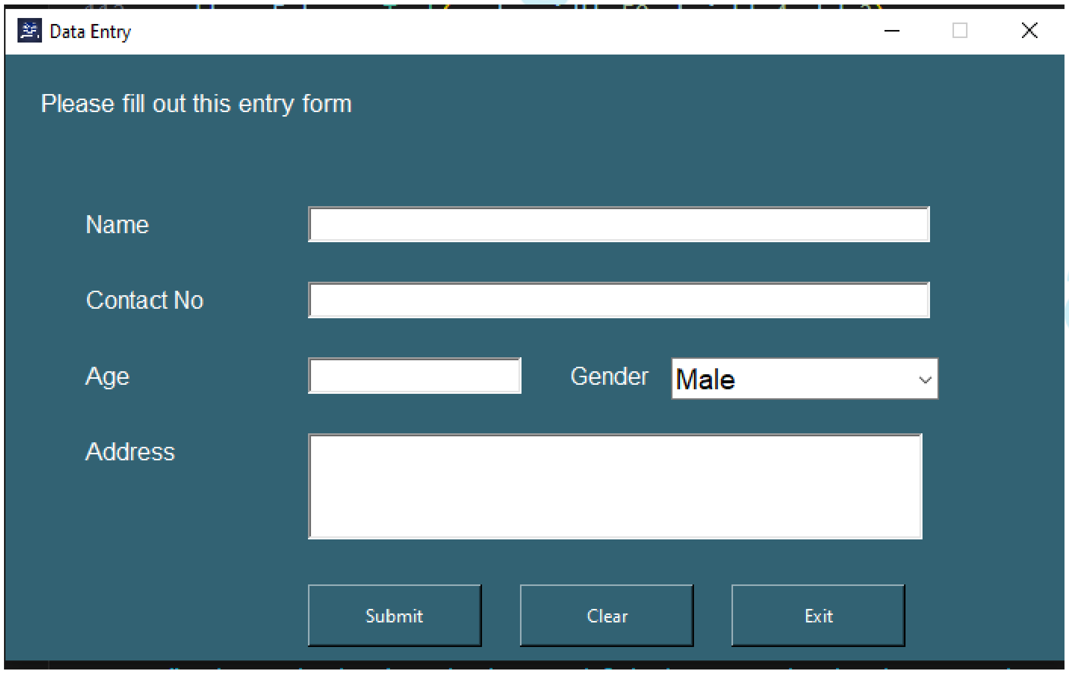
15. Now the text area for address:

```
# address
addressEntry = Text(root, width=50, height=4, bd=2)
nameEntry.place(x=200, y=100)
contactEntry.place(x=200, y=150)
ageEntry.place(x=200, y=200)
addressEntry.place(x=200, y=250)
```

16. Lastly, we will add buttons to submit the data.

```
# buttons
Button(root, text="Submit", bg="#326273", fg="white",
       width=15, height=2, command=submit).place(x=200, y=350)
Button(root, text="Clear", bg="#326273", fg="white",
       width=15, height=2, command=clear).place(x=340, y=350)
Button(root, text="Exit", bg="#326273", fg="white", width=15,
       height=2, command=lambda: root.destroy()).place(x=480, y=350)
```

17. The program should run and following display should appear:



18. Till this, you have completed the first part of the assignment. Good Job!

ASSIGNMENT DETAILS

Now this assignment consists of 10% weightage. Which will be divided as per the following scheme:

1. Completion of the above tutorial. (40%)
2. Update the code so that the following information can be saved: (40%)
    a. Merit N Merit Registration Form
    b. Full Name, Father Name, Personal Mobile No, Parent's Mobile Number, Gender (Radio), Class (list), Section (list), Date of Registration, Fees, A checkbox that will ask if the user also wants to get updates, plus three buttons.
3. Update the look of the form using CustomTkinter. (20%)
    a. Quick Introduction: https://www.youtube.com/watch?v=Miydkti_QVE
    b. Official Documentation: https://customtkinter.tomschimansky.com/
4. **Due Date: Saturday, March 2, 2024.**
5. **NOTE:** This is team assignment.
    a. **Team of 3 members** is required.
    b. **Connect via:** Discord
    c. Cross team help is allowed.
    d. Usage of AI is allowed.
    e. Copy pasting of the relevant code (that can fit and work) is also allowed.