FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERIG

**Department of Computer Engineering**

| **Academic Year** | 2023 - 24 | **Estimated Time** | Experiment No. 8 – 02 Hours |
| --- | --- | --- | --- |
| **Course & Semester** | S.E. (COMP) – Sem. IV | **Subject Name** | Python Programming Lab |
| **Module No.** | 04 | **Chapter Title** | Python Basics |
| **Experiment Type** | Software Performance | **Subject Code** | CSL405 |
| **Name of Student** |  | **Roll No.** |  |
| **Date of Performance:** |  | **Date of Submission:** |  |
| **CO Mapping** | CSL405.4: 6. Develop real world application using frameworks and in built libraries in python**.** | | |
| **Timeline**  **(2)** | **Preparedness**  **(2)** | **Effort**  **(2)** | **Result**  **(2)** | **Documentation**  **(2)** | **Total (10)** |
|  |  |  |  |  |  |

# Course Details:Experiment 8 – Creating GUI with Python

# Aim & Objective of Experiment

Develop a registration application form using Tkinter. The application should allow users to input their personal information and submit the form for registration. The form should include the following features:

1. **Input Fields:**

Name: Allows users to input their full name.

Email: Allows users to input their email address.

Password: Allows users to input a password (masked for security).

Date of Birth: Allows users to input their date of birth using a date picker widget.

Gender: Allows users to select their gender from a dropdown menu.

Address: Allows users to input their address in a multi-line text field.

1. **Submit Button:**

A button that users can click to submit their registration form.

1. **Validation:**

Implement validation for each input field to ensure that required fields are filled out correctly. Display error messages if the user's input is invalid (e.g., invalid email format, password too short).

1. **Confirmation:**

Once the form is submitted successfully, display a confirmation message to the user. Optionally, provide a summary of the information submitted for review.

1. **Reset Button:**

Include a button that allows users to reset the form and clear all input fields.

1. **Layout and Design:**

Design the form layout to be visually appealing and easy to navigate. Use appropriate labels, spacing, and alignment to organize the input fields.

1. **Error Handling:**

Handle any errors that may occur during form submission or validation gracefully. Provide feedback to the user if there are issues with submitting the form.

**Objectives**: 1) To create Graphical user interface in Python using Tkinter library.

2) To explore variety of widgets while creating Registration form.

**Pre-Requisite:** Any programming language like C, C++

**Tools:** Python IDE

# Theory:

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. Tkinter is a cross-platform library. It comes with most Python installations and it works on Windows, Mac, and Linux. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

# 

# To create a tkinter app:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.
5. Use tkinter library for creating GUI

# Sample Code:

from tkinter import Tk root = Tk()

l1 = Label(root,text = 'Hello Tkinter is easy') l1.pack()

root.mainloop()

Above code, creates a main window – root. Then creates a Label object and place it in a root container. Label.pack() method automatically positions the Label in a root window.

# Geometric Configuration of widgets:

* tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes.
* pack() method: It organizes the widgets in blocks before placing in the parent widget.
* grid() method: It organizes the widgets in grid (table-like structure) before placing in the parent widget.
* place() method: It organizes the widgets by placing them on specific positions directed by the programmer.

import tkinter as tk

from tkinter import ttk

from tkinter import messagebox

from tkcalendar import DateEntry

class Form(tk.Tk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.title("User Details Form")

# Name

tk.Label(self, text="Name: ").grid(row=0, column=0, sticky="e")

self.name = tk.Entry(self)

self.name.grid(row=0, column=1, padx=5, pady=5)

# Email

tk.Label(self, text="Email: ").grid(row=1, column=0, sticky="e")

self.email = tk.Entry(self)

self.email.grid(row=1, column=1, padx=5, pady=5)

# Password

tk.Label(self, text="Password: ").grid(row=2, column=0, sticky="e")

self.password = tk.Entry(self, show="\*")

self.password.grid(row=2, column=1, padx=5, pady=5)

# DOB

tk.Label(self, text="Date of birth: ").grid(row=3, column=0, sticky="e")

self.dob = DateEntry(self)

self.dob.grid(row=3, column=1, padx=5, pady=5)

# Gender

tk.Label(self, text="Gender: ").grid(row=4, column=0, sticky="e")

self.gender = ttk.Combobox(self, values=["M", "F", "Other"])

self.gender.grid(row=4, column=1, padx=5, pady=5)

# Address

tk.Label(self, text="Address: ").grid(row=5, column=0, sticky="ne")

self.address = tk.Text(self, width=20, height=4)

self.address.grid(row=5, column=1, padx=5, pady=5)

# Submit and reset buttons

self.submitB = tk.Button(self, text="Submit", command=self.submitForm)

self.submitB.grid(row=6, column=0, columnspan=2, pady=10)

self.resetB = tk.Button(self, text="Reset", command=self.resetForm)

self.resetB.grid(row=6, column=1, columnspan=2, pady=10)

def submitForm(self):

user\_name = self.name.get()

user\_email = self.email.get()

user\_password = self.password.get()

user\_dob = self.dob.get()

user\_gender = self.gender.get()

user\_address = self.address.get("1.0", "end-1c")

if not all([user\_name, user\_email, user\_password, user\_dob, user\_gender, user\_address]):

messagebox.showerror("Error", "Kindly Fill all fields.")

return

if len(user\_password) < 6:

messagebox.showerror("Error", "Password should be 6 characters long.")

return

messagebox.showinfo("Success", "User registration successful!")

def resetForm(self):

self.name.delete(0, tk.END)

self.email.delete(0, tk.END)

self.password.delete(0, tk.END)

self.dob.delete(0, tk.END)

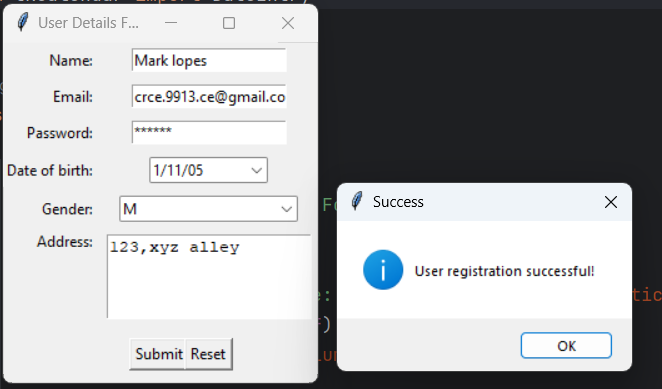
self.gender.current('')

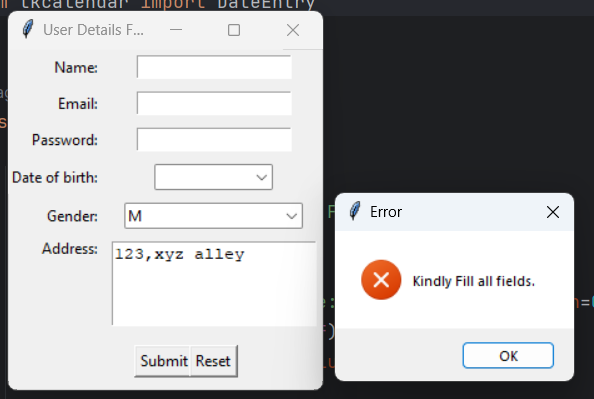
self.address.delete("1.0", tk.END)

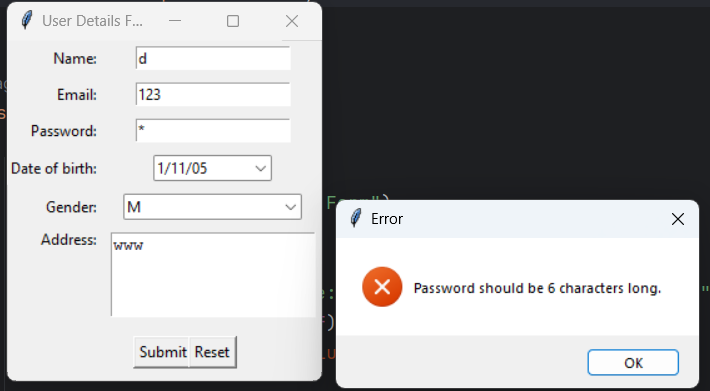
if \_\_name\_\_ == "\_\_main\_\_":

app = Form()

app.mainloop()







# Post Lab questions:

1. What are the different libraries available in Python to create GUI?

Tkinter: Tkinter is the standard GUI toolkit for Python. It is included with most Python installations, making it readily available for developers. Tkinter is easy to use and offers a wide range of widgets for creating desktop applications.

PyQt: PyQt is a set of Python bindings for the Qt application framework. It provides a comprehensive set of tools for developing cross-platform applications with a modern and professional look. PyQt is powerful and feature-rich but may have a steeper learning curve compared to Tkinter.

PyGTK: PyGTK is a set of Python bindings for the GTK+ toolkit. It allows developers to create applications with a native look and feel on Unix-like operating systems, including Linux. PyGTK is suitable for building desktop applications with complex user interfaces.

1. Explain how do you create dialog box (example Message Box or Input Box) using PyAutoGUI?

import tkinter as tk

from tkinter import messagebox

import pyautogui

def show\_message\_box():

messagebox.showinfo("Title", "This is a message box")

# Create a Tkinter window

root = tk.Tk()

root.geometry("300x200")

# Create a button to display the message box

btn = tk.Button(root, text="Show Message Box", command=show\_message\_box)

btn.pack(pady=20)

# Start the Tkinter event loop

root.mainloop()

