Project Report

Student Name: Rahul Kumar **UID: 24MCA20169**
Branch: MCA General **Section/Group: 3B**
Semester: 1st **Date of Performance: 29-Oct**
Subject Name: PYTHON PROGRAMMING LAB **Subject Code: 24CAH-606**

1. **Aim of the project:** language translator Using Tkinter

2. Software Requirements:

- **Python Interpreter:** Install the desired version of Python from python.org (e.g., Python 3.12.5) Ensure Python is added to your system's PATH.
- **Development Environment:** Choose an integrated development environment (IDE) or text editor that supports python. Popular choices include:
  PyCharm, Visual Studio Code, Jupyter notebook.
- **Anaconda Distribution:**Alternatively, you can use the Anaconda distribution, which comes with Python, Jupyter Notebook, and many scientific libraries pre-installed. It's a comprehensive package that simplifies the installation and management of Python environments

3. Program Logic:

Import Tkinter Modules:

- The program begins by importing `tkinter` and additional required modules like `messagebox` and `font` for enhanced functionality.
- Download Translator Module (pip install googletrans==4.0.0-rc1)

Create Main Window:

- A `Tk()` object is used to create the main window, with properties such as title, geometry, and background color.

Create Task Entry Field:

- A text entry widget for users to input new tasks.

Create Listbox for Tasks:

- A listbox widget to display tasks, with color customization and selection options.

Define Functions:

- `add_task()`: Adds a new task to the listbox and clears the entry field. Shows a warning if no task is entered.
- `delete_task()`: Deletes the selected task from the listbox. Warns the user if no task is selected.
- `mark_done()`: Marks the selected task as completed by appending a checkmark to the task's text. Warns the user if no task is selected.

Run Main Loop:

- The `root.mainloop()` command runs the Tkinter main event loop, keeping the window open and interactive.

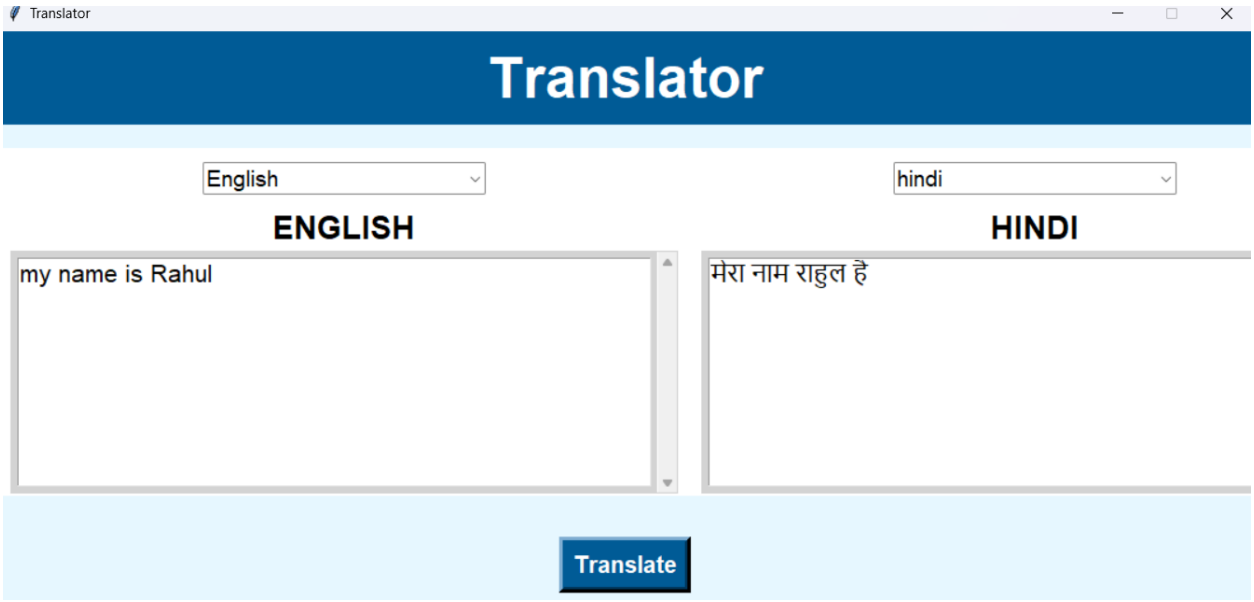
4. Code:

```
from tkinter import *
from tkinter import ttk
import googletrans
from googletrans import Translator

root = Tk()
root.title("Translator")
root.geometry("1080x500")
root.resizable(False, False)
root.configure(background="#e6f7ff")  # Light blue background for a fresh look

header = Label(root, text="Translator", font="Helvetica 36 bold", bg="#005b96", fg="white", padx=20, pady=10)
header.pack(fill=X)

def custom_messagebox(title, message):
    top = Toplevel(root)
    top.title(title)
    top.geometry("400x200")
    top.resizable(False, False)
    top.config(bg="#ffffff")

    canvas = Canvas(top, width=400, height=200, bg="#ffffff", highlightthickness=0)
```

```
canvas.pack()

    canvas.create_rectangle(20, 20, 380, 180, outline="#005b96", width=2, fill="#e6f7ff")

   label = Label(top, text=message, font="Roboto 14", bg="#e6f7ff", wraplength=350)
   label.place(x=50, y=60)

   ok_button = Button(top, text="OK", font="Roboto 12 bold", command=top.destroy, bg="#005b96", fg="white")
   ok_button.place(x=170, y=140)

def label_change():
   c = combo1.get()
   c1 = combo2.get()
   label1.configure(text=c.upper())
   label2.configure(text=c1.upper())
   root.after(1000, label_change)

# Function to handle text translation
def translate_now():
   text_ = text1.get(1.0, END).strip()
   if text_:
      try:
         t1 = Translator()
         trans_text = t1.translate(text_, src=combo1.get(), dest=combo2.get())
         trans_text = trans_text.text
         text2.delete(1.0, END)
         text2.insert(END, trans_text)
      except Exception as e:
         custom_messagebox("Translation Error", f"An error occurred: {e}")

# Get language list from googletrans
language = googletrans.LANGUAGES
languageV = list(language.values())

# Frame for input and output areas
frame = Frame(root, bg="white", bd=2)
frame.pack(pady=20)

# Source language combobox
combo1 = ttk.Combobox(frame, values=languageV, font="Roboto 14", state="readonly", width=20)
combo1.grid(row=0, column=0, padx=10, pady=10)
combo1.set("English")

# Source language label
label1 = Label(frame, text="ENGLISH", font="Segoe 20 bold", bg="white")
label1.grid(row=1, column=0, padx=10)
```

```
f = Frame(frame, bg="#d3d3d3", bd=1)
f.grid(row=2, column=0, padx=10)
text1 = Text(f, font="Roboto 16", bg="white", relief=GROOVE, wrap=WORD, height=8, width=45)

text1.pack(side=LEFT, padx=5, pady=5)

scrollbar1 = Scrollbar(f)
scrollbar1.pack(side="right", fill="y")
scrollbar1.configure(command=text1.yview)
text1.configure(yscrollcommand=scrollbar1.set)

combo2 = ttk.Combobox(frame, values=languageV, font="Roboto 14", state="readonly", width=20)
combo2.grid(row=0, column=1, padx=10, pady=10)
combo2.set("SELECT LANGUAGE")
label2 = Label(frame, text="SELECT LANGUAGE", font="Segoe 20 bold", bg="white")
label2.grid(row=1, column=1, padx=10)

f1 = Frame(frame, bg="#d3d3d3", bd=1)
f1.grid(row=2, column=1, padx=10)
text2 = Text(f1, font="Roboto 16", bg="white", relief=GROOVE, wrap=WORD, height=8, width=45)
text2.pack(side=LEFT, padx=5, pady=5)

scrollbar2 = Scrollbar(f1)
scrollbar2.pack(side="right", fill="y")
scrollbar2.configure(command=text2.yview)
text2.configure(yscrollcommand=scrollbar2.set)

translate = Button(root, text="Translate", font="Roboto 15 bold", activebackground="#4caf50",
                   cursor="hand2", bd=5, bg='#005b96', fg="white", command=translate_now)
translate.pack(pady=15)

label_change()


root.mainloop()
```

Output Result:



Learning outcomes (What I have learnt):

1. User-Friendly Interface for Language Translation

2. Real-Time Translation Feedback

3. You can enhance your translator by connecting it with translation APIs like Google Translate or DeepL, enabling support for more languages and more accurate translations.

4. As a Tkinter application, your translator can be run on Windows, macOS, and Linux, making it widely accessible to various users who need a desktop translation tool.