EXPERIMENT NO : 5B

Python programs to implement GUI Application using Tkinter.

NAME : AKASH RAMKRIT YADAV		***ID.NO: VU4F2122016***
BATCH : A	***BRANCH : IT***	***DIV : A***

Aim :- *python programs to implement GUI Application using Tkinter.*

THEORY:

OUTPUT:

```
Python 3.11.0a4 (main, Mar 13  2023, 10:57:32) [MSC v.1929 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
#AKASH YADAV     ID.NO:VU4F2122016     EXP:5B    DATE:15/3/2023
```

Python GUI – tkinter

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. 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.*

Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x it is 'tkinter'.

```
import tkinter
```

There are two main methods used which the user needs to remember while creating the Python application with GUI.

1. ***Tk(screenName=None, baseName=None, className='Tk', useTk=1):***

   To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:
   `m=tkinter.Tk() where m is the name of the main window object`

2. ***mainloop():*** *There is a method known by the name mainloop() is used when your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.*
   `m.mainloop()`

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 class.

1. ***pack() method:****It organizes the widgets in blocks before placing in the parent widget.*

2. ***grid() method:****It organizes the widgets in grid (table-like structure) before placing in the parent widget.*

3. ***place() method:****It organizes the widgets by placing them on specific positions directed by the programmer.*

There are a number of widgets which you can put in your tkinter application. Some of the major widgets are explained below:

1. ***Button****: To add a button in your application, this widget is used.*
   The general syntax is:
   `w=Button(master, option=value)`

   master is the parameter used to represent the parent window.
   There are number of options which are used to change the format of the Buttons. Number of options can be passed as parameters separated by commas. Some of them are listed below.

   - ***activebackground****: to set the background color when button is under the cursor.*
   - ***activeforeground****: to set the foreground color when button is under the cursor.*
   - ***bg****: to set the normal background color.*
   - ***command****: to call a function.*

- ***font****: to set the font on the button label.*
- ***image****: to set the image on the button.*
- ***width****: to set the width of the button.*
- ***height****: to set the height of the button.*

2. ***Canvas:*** *It is used to draw pictures and other complex layout like graphics, text and widgets.*

   The general syntax is:

   ```
   w = Canvas(master, option=value)
   ```

   master is the parameter used to represent the parent window.

   There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

   - ***bd****: to set the border width in pixels.*
   - ***bg****: to set the normal background color.*
   - ***cursor****: to set the cursor used in the canvas.*
   - ***highlightcolor****: to set the color shown in the focus highlight.*
   - ***width****: to set the width of the widget.*

3. ***CheckButton****: To select any number of options by displaying a number of options to a user as toggle buttons.*

   The general syntax is:

   ```
   w = CheckButton(master, option=value)
   ```

   There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

   - ***Title****: To set the title of the widget.*
   - ***activebackground****: to set the background color when widget is under the cursor.*
   - ***activeforeground****: to set the foreground color when widget is under the cursor.*
   - ***bg****: to set the normal backgrouSteganography*
     Break

     Secret Code:

     Attach a File:nd color.

   - ***command****: to call a function.*
   - ***font****: to set the font on the button label.*

- ***image****: to set the image on the widget.*

4. ***Entry:****It is used to input the single line text entry from the user.. For multi-line text input, Text widget is used.*

   The general syntax is:
   `w=Entry(master, option=value)`

   master is the parameter used to represent the parent window.
   There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

   - ***bd****: to set the border width in pixels.*
   - ***bg****: to set the normal background color.*
   - ***cursor****: to set the cursor used.*
   - ***command****: to call a function.*
   - ***highlightcolor****: to set the color shown in the focus highlight.*
   - ***width****: to set the width of the button.*
   - ***height****: to set the height of the button.*

5. ***Frame:*** *It acts as a container to hold the widgets. It is used for grouping and organizing the widgets*

   . The general syntax is:
   `w = Frame(master, option=value)`

   `master is the parameter used to represent the parent window.`

   There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

   - ***highlightcolor****: To set the color of the focus highlight when widget has to be focused.*
   - ***bd****: to set the border width in pixels.*
   - ***bg****: to set the normal background color.*
   - ***cursor****: to set the cursor used.*
   - ***width****: to set the width of the widget.*
   - ***height****: to set the height of the widget.*

6. ***Label**: It refers to the display box where you can put any text or image which can be updated any time as per the code.*

   The general syntax is:
   `w=Label(master, option=value)`

   `master is the parameter used to represent the parent window.`

   There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

   - ***bg**: to set the normal background color.*
   - ***bg** to set the normal background color.*
   - ***command**: to call a function.*
   - ***font**: to set the font on the button label.*
   - ***image**: to set the image on the button.*
   - ***width**: to set the width of the button.*
   - ***height**" to set the height of the button.*

7. ***Listbox**: It offers a list to the user from which the user can accept any number of options.*

   The general syntax is:
   `w = Listbox(master, option=value)`

   `master is the parameter used to represent the parent window.`

   There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

   - ***highlightcolor**: To set the color of the focus highlight when widget has to be focused.*
   - ***bg**: to set the normal background color.*
   - ***bd**: to set the border width in pixels.*
   - ***font**: to set the font on the button label.*
   - ***image**: to set the image on the widget.*
   - ***width**: to set the width of the widget.*
   - ***height**: to set the height of the widget.*

8. ***MenuButton**: It is a part of top-down menu which stays on the window all the time. Every menubutton has its own functionality.*

   The general syntax is:
   `w = MenuButton(master, option=value)`

`master is the parameter used to represent the parent window.`

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- ***activebackground**: To set the background when mouse is over the widget.*
- ***activeforeground**: To set the foreground when mouse is over the widget.*
- ***bg**: to set the normal background color.*
- ***bd**: to set the size of border around the indicator.*
- ***cursor**: To appear the cursor when the mouse over the menubutton.*
- ***image**: to set the image on the widget.*
- ***width**: to set the width of the widget.*
- ***height**: to set the height of the widget.*
- ***highlightcolor**: To set the color of the focus highlight when widget has to be focused.*

9. ***Menu**: It is used to create all kinds of menus used by the application.*
   The general syntax is:

```
10.    w = Menu(master, option=value)
```

`master is the parameter used to represent the parent window.`

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- ***title**: To set the title of the widget.*
- ***activebackground**: to set the background color when widget is under the cursor.*
- ***activeforeground**: to set the foreground color when widget is under the cursor.*
- ***bg**: to set the normal background color.*
- ***command**: to call a function.*
- ***font**: to set the font on the button label.*
- ***image**: to set the image on the widget.*

11. ***Message**: It refers to the multi-line and non-editable text. It works same as that of Label.*

The general syntax is:
`w = Message(master, option=value)`

```
12.    master is the parameter used to represent the parent window.
```

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- ***bd**: to set the border around the indicator.*
- ***bg**: to set the normal background color.*
- ***font**: to set the font on the button label.*
- ***image**: to set the image on the widget.*
- ***width**: to set the width of the widget.*
- ***height**: to set the height of the widget.*

13. ***RadioButton:*** *It is used to offer multi-choice option to the user. It offers several options to the user and the user has to choose one option.*

The general syntax is:

```
w = RadioButton(master, option=value)
```

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- ***activebackground**: to set the background color when widget is under the cursor.*
- ***activeforeground**: to set the foreground color when widget is under the cursor.*
- ***bg**: to set the normal background color.*
- ***command**: to call a function.*
- ***font**: to set the font on the button label.*
- ***image**: to set the image on the widget.*
- ***width**: to set the width of the label in characters.*
- ***height**: to set the height of the label in characters.*

Output:

14. ***Scale:*** *It is used to provide a graphical slider that allows to select any value from that scale.*

The general syntax is:

```
w = Scale(master, option=value)
```

```
master is the parameter used to represent the parent window.
```

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- ***cursor**: To change the cursor pattern when the mouse is over the widget.*

- ***activebackground****: To set the background of the widget when mouse is over the widget.*
- ***bg****: to set the normal background color.*
- ***orient****: Set it to HORIZONTAL or VERTICAL according to the requirement.*
- ***from_****: To set the value of one end of the scale range.*
- ***to****: To set the value of the other end of the scale range.*
- ***image****: to set the image on the widget.*
- ***width****: to set the width of the widget.*

15. ***Scrollbar****: It refers to the slide controller which will be used to implement listed widgets.*

    The general syntax is:

```
w = Scrollbar(master, option=value)
```

```
master is the parameter used to represent the parent window.
```

    There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- ***width****: to set the width of the widget.*
- ***activebackground****: To set the background when mouse is over the widget.*
- ***bg****: to set the normal background color.*
- ***bd****: to set the size of border around the indicator.*
- ***cursor****: To appear the cursor when the mouse over the menubutton.*

    Output:

16. ***Text:*** *To edit a multi-line text and format the way it has to be displayed.*
    The general syntax is:

```
w  =Text(master, option=value)
```

    There are number of options which are used to change the format of the text. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- ***highlightcolor****: To set the color of the focus highlight when widget has to be focused.*
- ***insertbackground****: To set the background of the widget.*
- ***bg****: to set the normal background color.*
- ***font****: to set the font on the button label.*
- ***image****: to set the image on the widget.*
- ***width****: to set the width of the widget.*
- ***height****: to set the height of the widget.*

17. ***TopLevel:*** *This widget is directly controlled by the window manager. It don't need any parent window to work on.*
The general syntax is:

```
w = TopLevel(master, option=value)
```

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- ***bg****: to set the normal background color.*
- ***bd****: to set the size of border around the indicator.*
- ***cursor****: To appear the cursor when the mouse over the menubutton.*
- ***width****: to set the width of the widget.*
- ***height****: to set the height of the widget.*

18. ***SpinBox:*** *It is an entry of 'Entry' widget. Here, value can be input by selecting a fixed value of numbers.*

The general syntax is:

```
w = SpinBox(master, option=value)
```

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- ***bg****: to set the normal background color.*
- ***bd****: to set the size of border around the indicator.*
- ***cursor****: To appear the cursor when the mouse over the menubutton.*
- ***command****: To call a function.*
- ***width****: to set the width of the widget.*
- ***activebackground****: To set the background when mouse is over the widget.*
- ***disabledbackground****: To disable the background when mouse is over the widget.*
- ***from_****: To set the value of one end of the range.*
- ***to****: To set the value of the other end of the range.*

19. ***PannedWindow:****It is a container widget which is used to handle number of panes arranged in it.*

The general syntax is:

```
w = PannedWindow(master, option=value)
```

master is the parameter used to represent the parent window.
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- ***bg**: to set the normal background color.*
- ***bd**: to set the size of border around the indicator.*
- ***cursor**: To appear the cursor when the mouse over the menubutton.*
- ***width**: to set the width of the widget.*
- ***height**: to set the height of the widget.*

CODE:

```
import tkinter as tk

from tkinter import messagebox

import os

import openai

import tkinter as tk

from PIL import Image, ImageTk

import speech_recognition as sr

from PIL import Image, ImageTk  # pip install pillow



class FirstPage(tk.Frame):

    def __init__(self, parent, controller):

        tk.Frame.__init__(self, parent)



        load = Image.open("login1.jpg")

        photo = ImageTk.PhotoImage(load)

        label = tk.Label(self, image=photo)

        label.image=photo

        label.place(x=0,y=0)
```

```
border = tk.LabelFrame(self, text='Login', bg='ivory', bd = 10, font=("Arial", 20))
border.pack(fill="both", expand="yes", padx = 150, pady=150)

L1 = tk.Label(border, text="Username", font=("Arial Bold", 15), bg='ivory')
L1.place(x=50, y=20)
T1 = tk.Entry(border, width = 30, bd = 5)
T1.place(x=180, y=20)

L2 = tk.Label(border, text="Password", font=("Arial Bold", 15), bg='ivory')
L2.place(x=50, y=80)
T2 = tk.Entry(border, width = 30, show='*', bd = 5)
T2.place(x=180, y=80)

def verify():
    try:
        with open("credential.txt", "r") as f:
            info = f.readlines()
            i  = 0
            for e in info:
                u, p =e.split(",")
                if u.strip() == T1.get() and p.strip() == T2.get():
                    controller.show_frame(SecondPage)
                    i = 1
                    break
            if i==0:
                messagebox.showinfo("Error", "Please provide correct username and password!!")
    except:
```

```
        messagebox.showinfo("Error", "Please provide correct username and password!!")


B1 = tk.Button(border, text="Submit", font=("Arial", 15), command=verify)
B1.place(x=320, y=115)


def register():
    window = tk.Tk()
    window.resizable(0,0)
    window.configure(bg="deep sky blue")
    window.title("Register")
    l1 = tk.Label(window, text="Username:", font=("Arial",15), bg="deep sky blue")
    l1.place(x=10, y=10)
    t1 = tk.Entry(window, width=30, bd=5)
    t1.place(x = 200, y=10)


    l2 = tk.Label(window, text="Password:", font=("Arial",15), bg="deep sky blue")
    l2.place(x=10, y=60)
    t2 = tk.Entry(window, width=30, show="*", bd=5)
    t2.place(x = 200, y=60)


    l3 = tk.Label(window, text="Confirm Password:", font=("Arial",15), bg="deep sky blue")
    l3.place(x=10, y=110)
    t3 = tk.Entry(window, width=30, show="*", bd=5)
    t3.place(x = 200, y=110)


    def check():
        if t1.get()!="" or t2.get()!="" or t3.get()!="":
```

```
            if t2.get()==t3.get():
                with open("credential.txt", "a") as f:
                    f.write(t1.get()+","+t2.get()+"\n")
                    messagebox.showinfo("Welcome","You are registered successfully!!")
            else:
                messagebox.showinfo("Error","Your password didn't get match!!")
        else:
            messagebox.showinfo("Error", "Please fill the complete field!!")

    b1 = tk.Button(window, text="Sign in", font=("Arial",15), bg="#ffc22a", command=check)
    b1.place(x=170, y=150)

    window.geometry("470x220")
    window.mainloop()

B2 = tk.Button(self, text="Register", bg = "dark orange", font=("Arial",15), command=register)
B2.place(x=650, y=20)

class SecondPage(tk.Frame):
  def __init__(self, parent, controller):
    tk.Frame.__init__(self, parent)

    load = Image.open("login1.jpg")
    photo = ImageTk.PhotoImage(load)
    label = tk.Label(self, image=photo)
    label.image=photo
    label.place(x=0,y=0)
```

```
def ask_ai(prompt):
    start_sequence = "\nAI:"
    restart_sequence = "\nHuman: "

    )

    return response["choices"][0]["text"]



class InputOutputApp:
    def __init__(self, master, image_path):
        self.master = master
        self.image_path = image_path

        self.canvas = tk.Canvas(self.master, width=1000, height=500)
        self.canvas.pack()

        #self.background_image = ImageTk.PhotoImage(Image.open(self.image_path)
  # self.canvas.create_image(200, 20, anchor="nw", image=self.background_image)

        self.bg=ImageTk.PhotoImage(file="login.jpg")
        self.bg_image=tk.Label(self.canvas,image=self.bg).place(x=1,y=1,relwidth=1,relheight=1)

self.label_input = tk.Label(self.canvas, text="Ask Any Question:", bg='orange', font=("Helvetica", 30))
        self.label_input.pack(pady=100)

        self.entry = tk.Entry(self.canvas, bg='lightgreen',font=("Helvetica", 30))
```

```
        self.entry.pack()

        self.button = tk.Button(self.canvas, text="Submit", command=self.submit, bg='orange', font=("Helvetica",
30))
        self.button.pack(pady=10)

        self.label_output = tk.Label(self.canvas, bg='lightgreen', font=("Helvetica", 10))
        self.label_output.pack()

        self.recognizer = sr.Recognizer()
        self.microphone = sr.Microphone()

    def submit(self):
        prompt = self.entry.get()
        if prompt:
            ai_response = ask_ai(prompt)
            self.label_output.config(text="AI: " + ai_response, wraplength=1000)
        else:
            with self.microphone as source:
                self.recognizer.adjust_for_ambient_noise(source)
                audio = self.recognizer.listen(source)
            try:
                prompt = self.recognizer.recognize_google(audio)
                ai_response = ask_ai(prompt)
                self.label_output.config(text="AI: " + ai_response, wraplength=700)
            except sr.UnknownValueError:
                self.label_output.config(text="Could not understand audio.")
            except sr.RequestError:
```

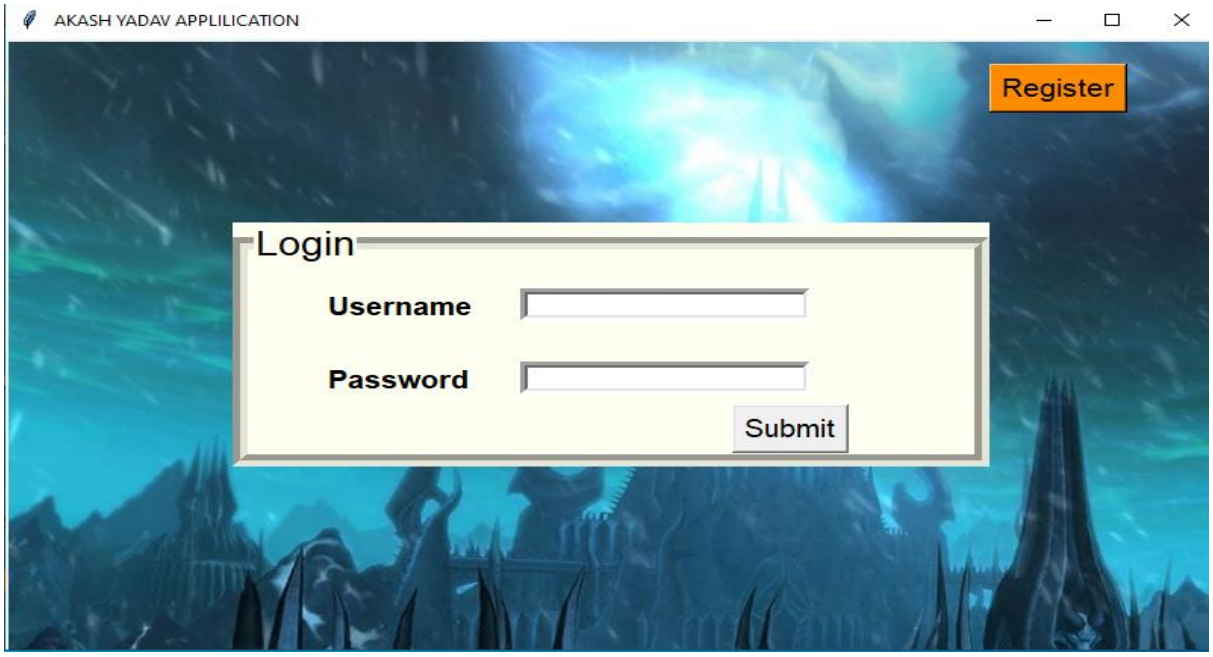
```
            self.label_output.config(text="Could not request results from Google Speech Recognition service.")
        except:
            self.label_output.config(text="Unknown error occurred.")

def main():
    root = tk.Tk()
    root.title("AKASH YADAV AI ChatBoat")
    root.geometry("1199x600+100+50")



    image_path = "login.jpg"
    input_output_app = InputOutputApp(root, image_path)

    root.mainloop()

if __name__ == "__main__":
    main()

        Button = tk.Button(self, text="Next", font=("Arial", 15), command=lambda:
controller.show_frame(ThirdPage))
        Button.place(x=650, y=450)

        Button = tk.Button(self, text="Back", font=("Arial", 15), command=lambda:
controller.show_frame(FirstPage))
        Button.place(x=100, y=450)

class ThirdPage(tk.Frame):
    def __init__(self, parent, controller):
```

```
        tk.Frame.__init__(self, parent)



        self.configure(bg='Tomato')

        #================================

        Label = tk.Label(self, text=" thank you for visiting our ai chat bot", bg = "orange", font=("Arial Bold", 25))

        Label.place(x=40, y=150)



        Button = tk.Button(self, text="Home", font=("Arial", 15), command=lambda: controller.show_frame(FirstPage))

        Button.place(x=650, y=450)



        Button = tk.Button(self, text="Back", font=("Arial", 15), command=lambda: controller.show_frame(SecondPage))

        Button.place(x=100, y=450)




class Application(tk.Tk):

    def __init__(self, *args, **kwargs):

        tk.Tk.__init__(self, *args, **kwargs)



        #creating a window

        window = tk.Frame(self)

        window.pack()



        window.grid_rowconfigure(0, minsize = 500)

        window.grid_columnconfigure(0, minsize = 800)



        self.frames = {}
```

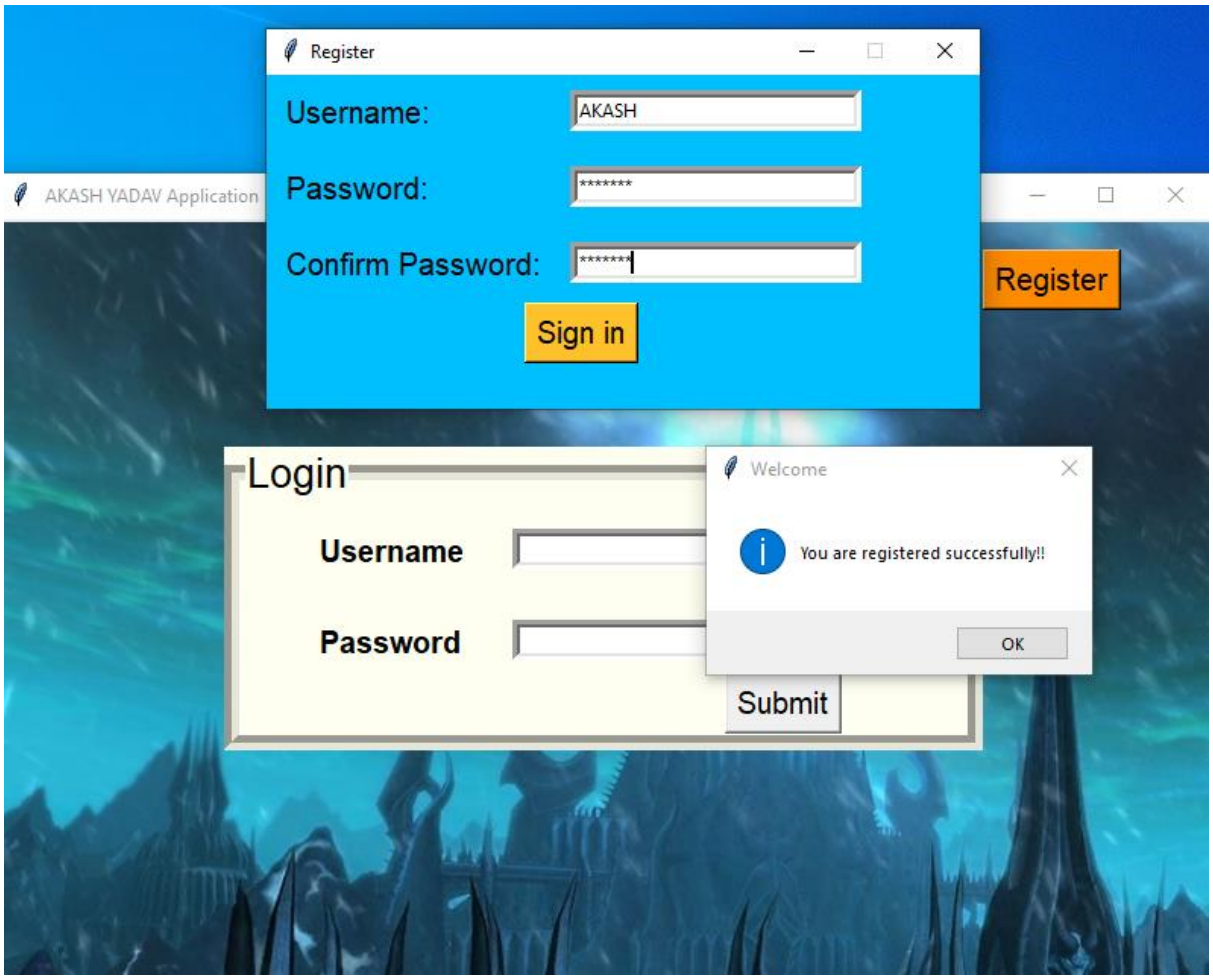
```
        for F in (FirstPage, SecondPage, ThirdPage):
            frame = F(window, self)
            self.frames[F] = frame
            frame.grid(row = 0, column=0, sticky="nsew")

        self.show_frame(FirstPage)

    def show_frame(self, page):
        frame = self.frames[page]
        frame.tkraise()
        self.title(" AKASH YADAV APPLILICATION")

app = Application()
app.maxsize(800,500)
app.mainloop()
```

OUTPUT:

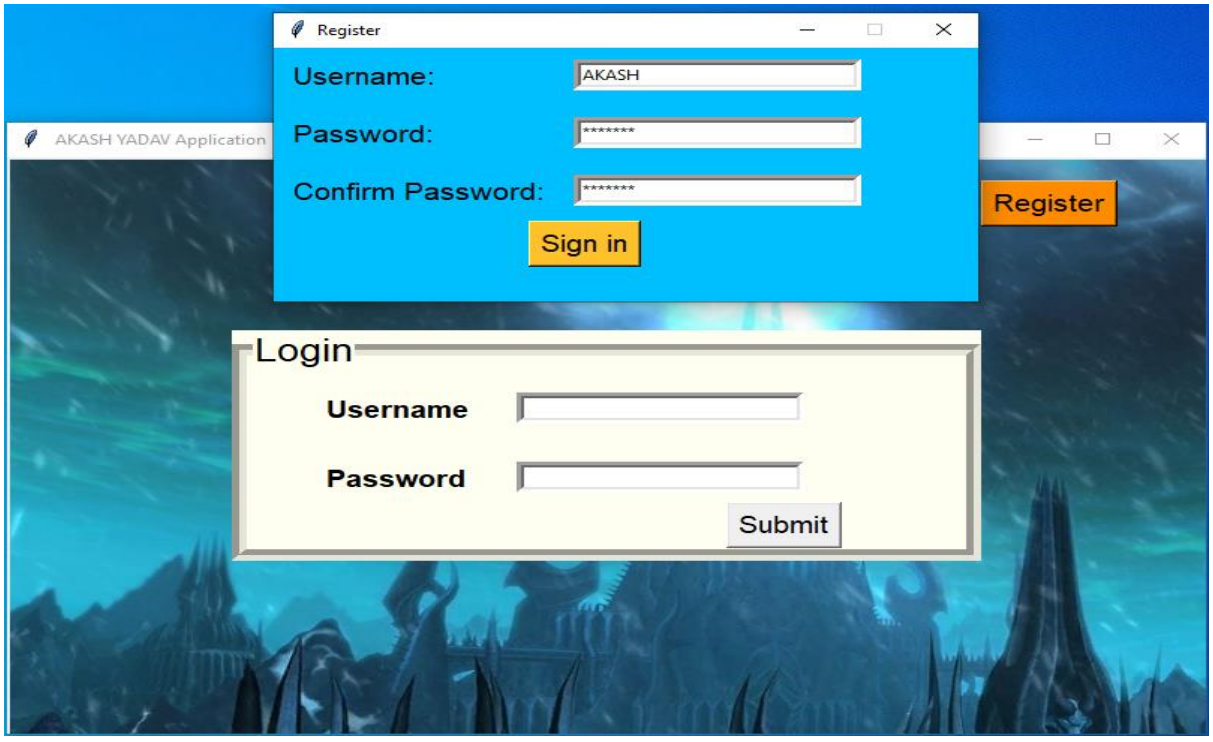
1]



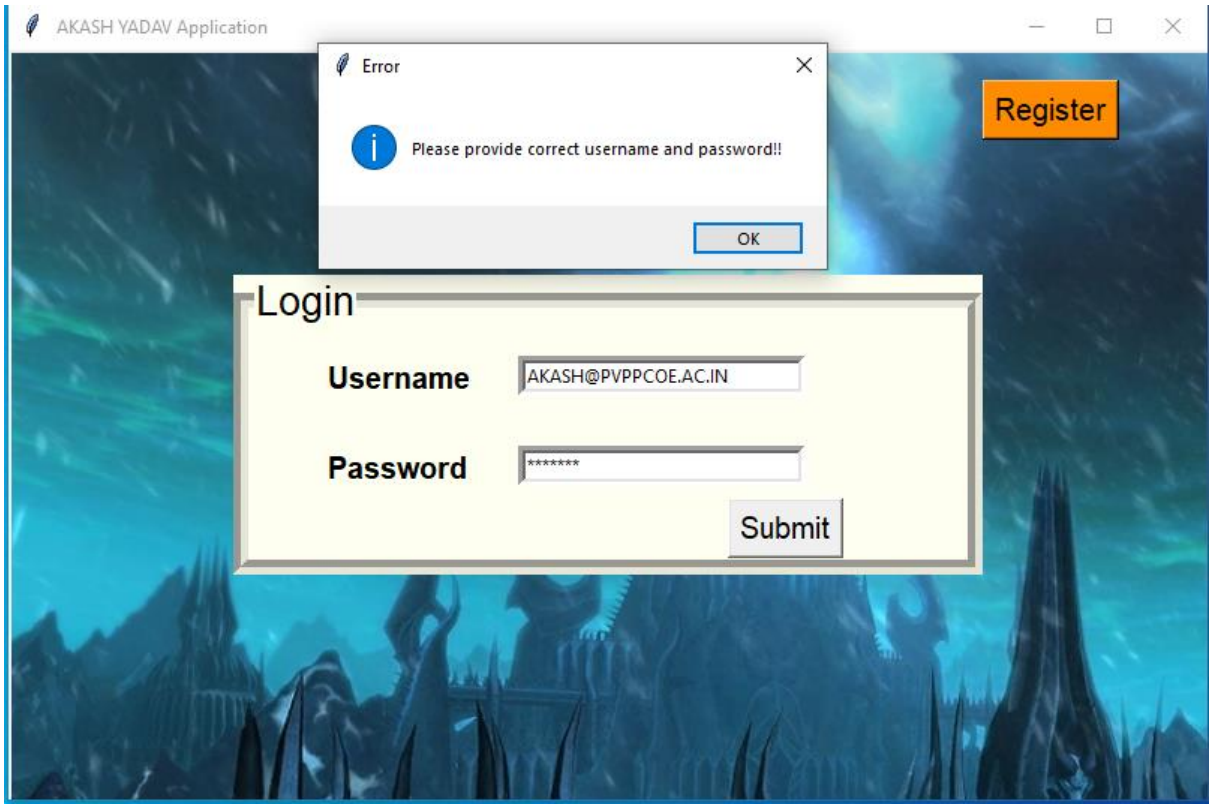
2]


Register
Username:
AKASH
Password:
Confirm Password:
Sign in
AKASH YADAV Application
Register
Login
Username
Password
Submit
Welcome
You are registered successfully!!
OK

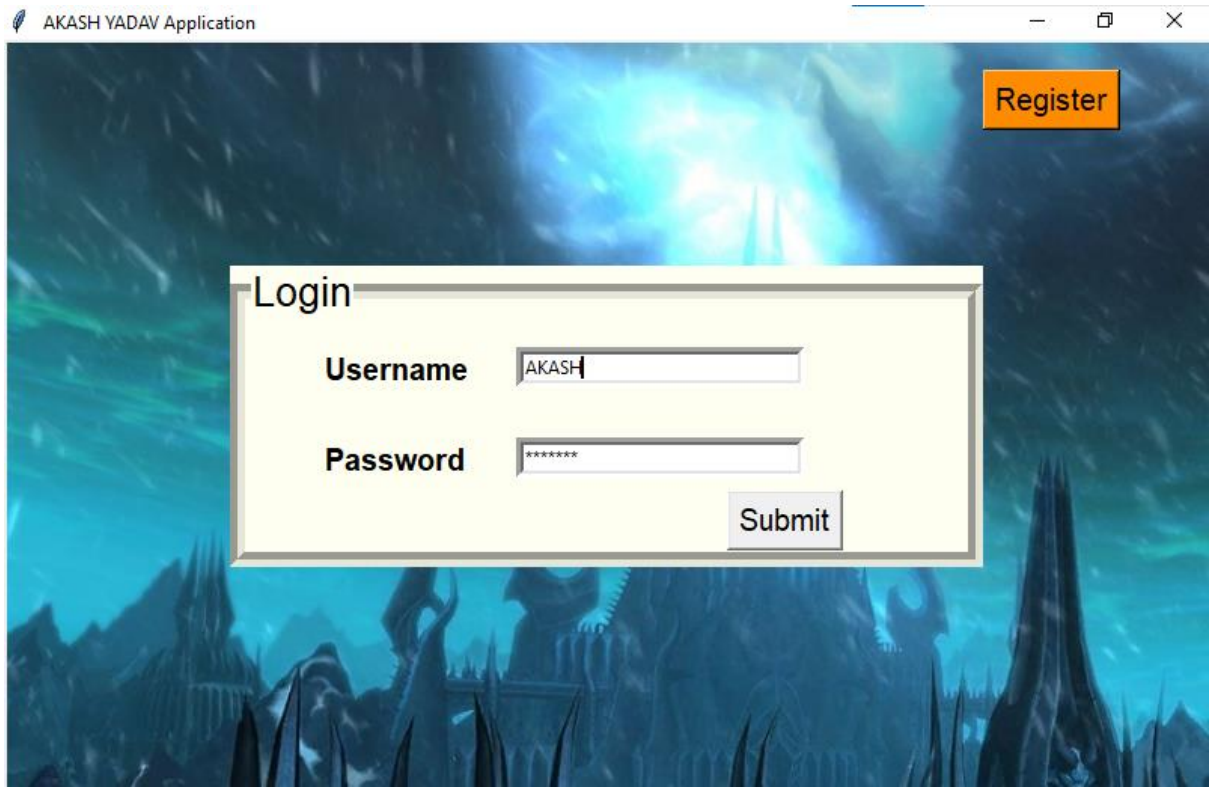
3]


Register
Username:
AKASH
Password:
Confirm Password:
Sign in
AKASH YADAV Application
Register
Login
Username
Password
Submit

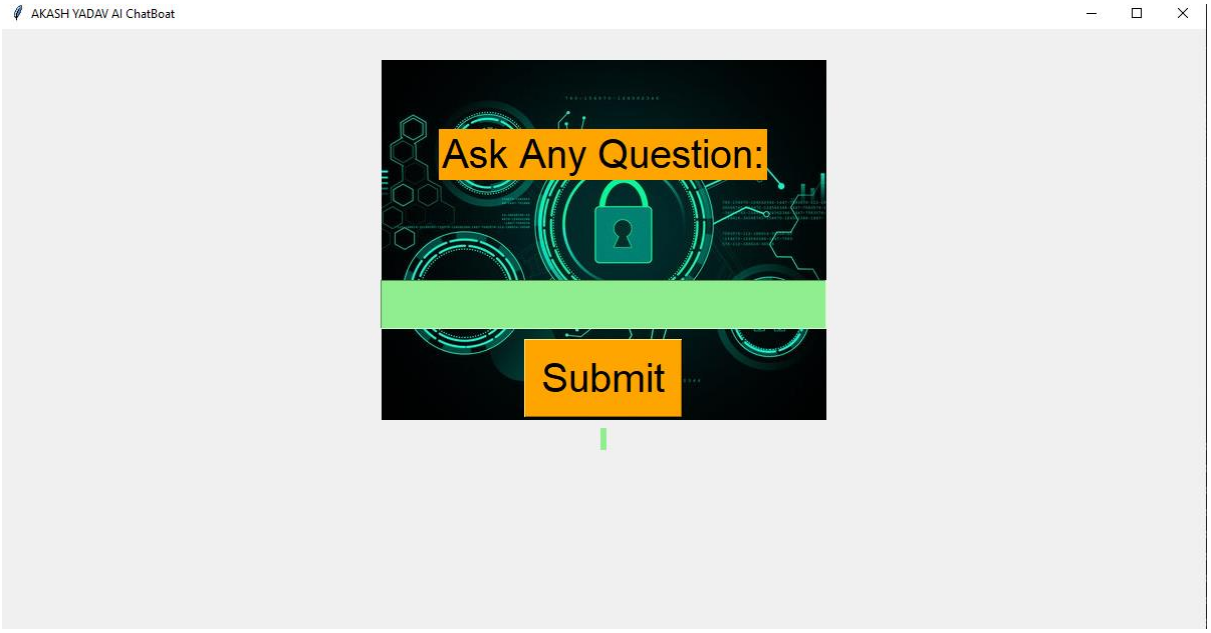
4]


AKASH YADAV Application
Error
Please provide correct username and password!!
OK
Register
Login
Username
AKASH@PVPPCOE.AC.IN
Password
****
Submit

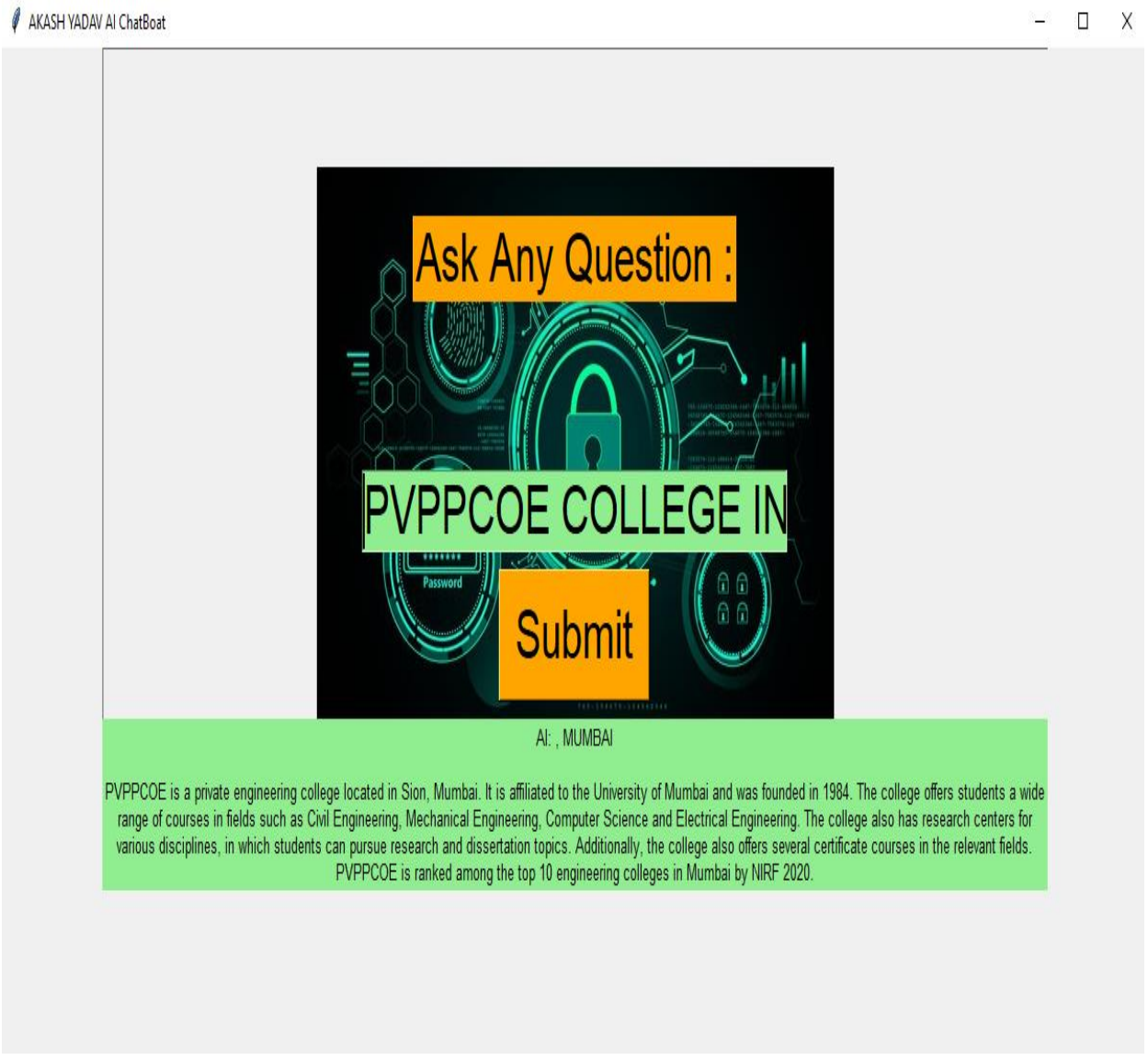
5]


AKASH YADAV Application
Register
Login
Username
AKASH
Password
****
Submit

6]


AKASH YADAV AI ChatBoat
Ask Any Question:
Submit

7]


AKASH YADAV AI ChatBoat
Ask Any Question :
PVPPCOE COLLEGE IN
Password
Submit
AI: , MUMBAI
PVPPCOE is a private engineering college located in Sion, Mumbai. It is affiliated to the University of Mumbai and was founded in 1984. The college offers students a wide range of courses in fields such as Civil Engineering, Mechanical Engineering, Computer Science and Electrical Engineering. The college also has research centers for various disciplines, in which students can pursue research and dissertation topics. Additionally, the college also offers several certificate courses in the relevant fields. PVPPCOE is ranked among the top 10 engineering colleges in Mumbai by NIRF 2020.

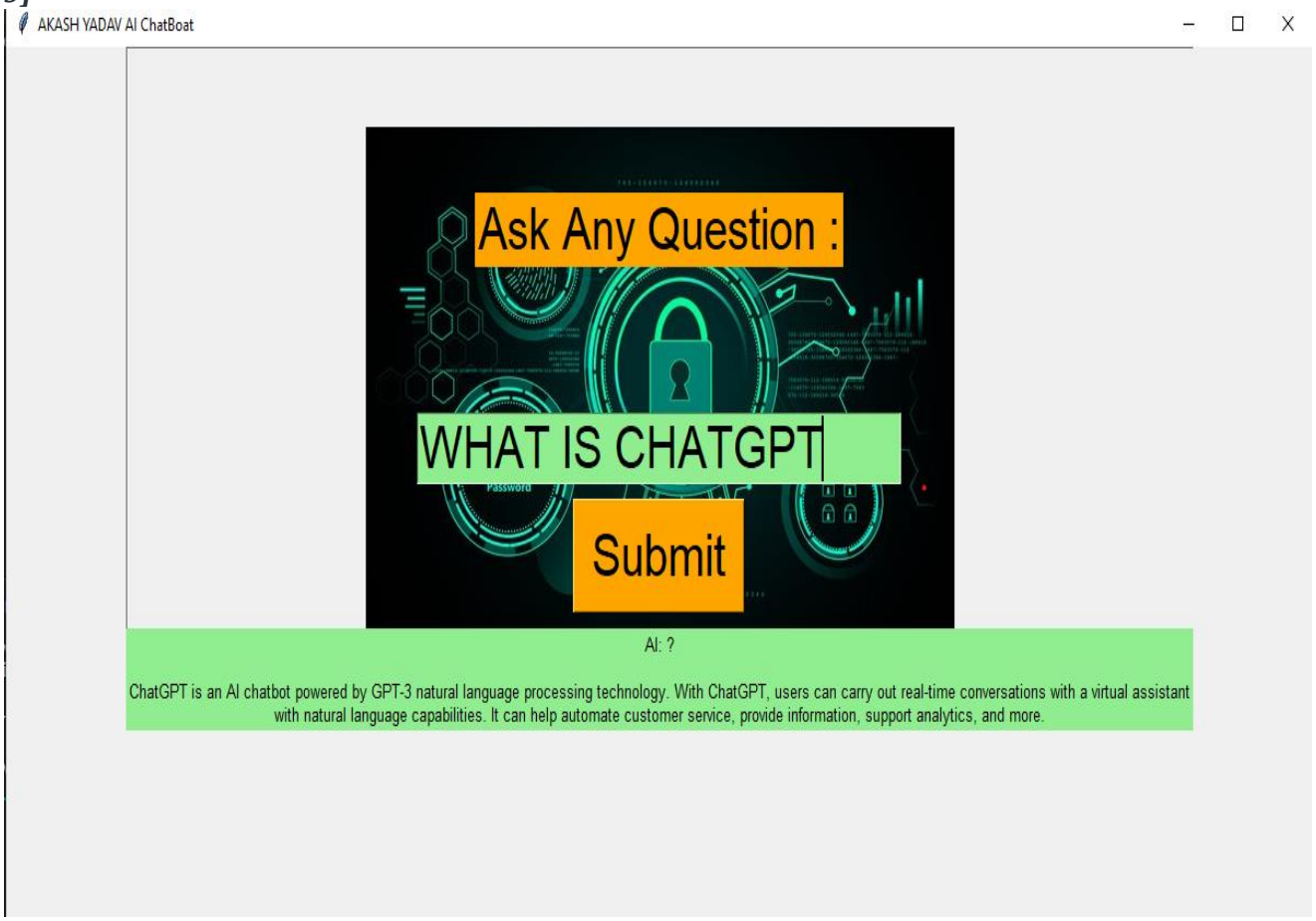
8]

AKASH YADAV ChatBoat

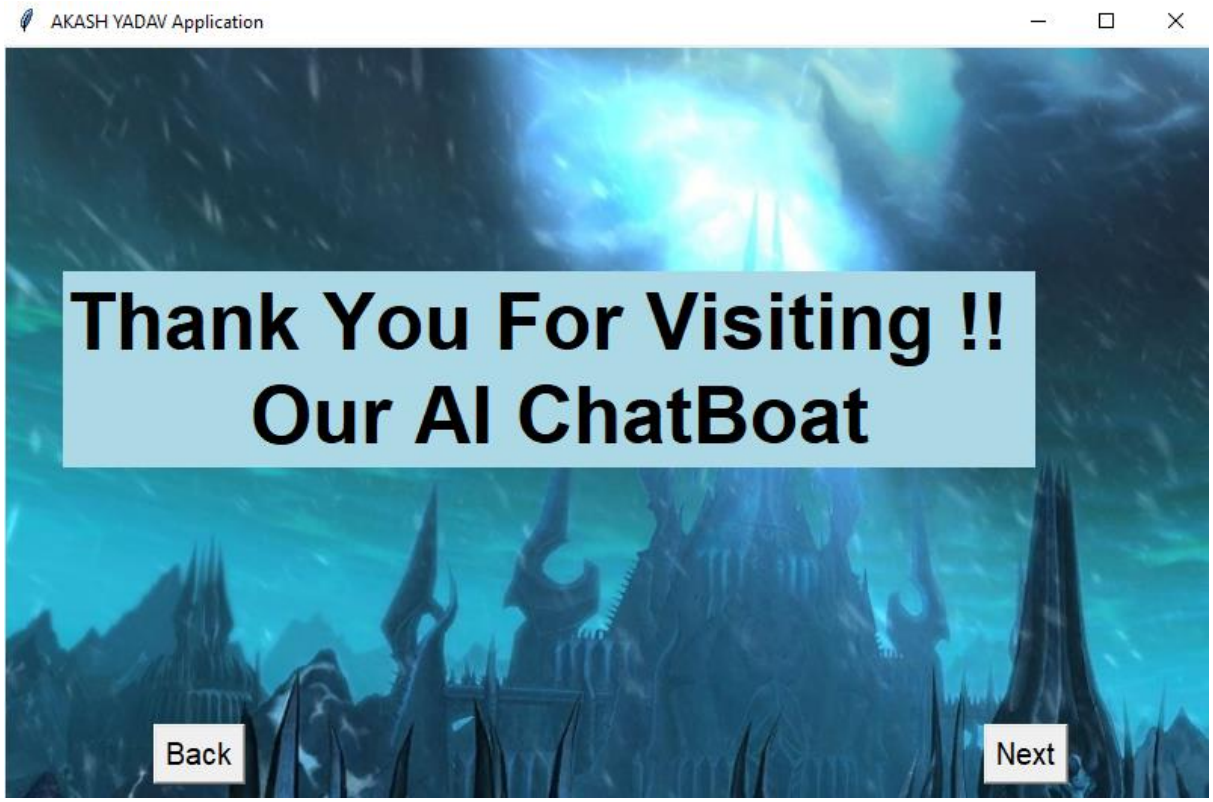
AI: ?

Python is a high-level, interpreted programming language with dynamic semantics that emphasizes code readability. It is used for general-purpose programming and has a wide variety of applications including web development, scripting, software development, system automation, data analysis, artificial intelligence, scientific computing, and game development. Python is an easy to learn, powerful language that is used by many developers around the world. The design philosophy of the language is simple - to make it as easy to use and understand as possible. Python has a large standard library that supports many common programming tasks, including string manipulation, regular expressions, and text processing.

9]


AKASH YADAV AI ChatBoat
Ask Any Question :
WHAT IS CHATGPT
Submit
AI: ?
ChatGPT is an AI chatbot powered by GPT-3 natural language processing technology. With ChatGPT, users can carry out real-time conversations with a virtual assistant with natural language capabilities. It can help automate customer service, provide information, support analytics, and more.

10]


AKASH YADAV Application
Thank You For Visiting !!
Our AI ChatBoat
Back
Next