TASK ASSIGNMENT

PYTHON DEVELOPMENT INTERNSHIP

NEXT24TECH TECHNOLOGY AND SERVICES

CARRIED OUT BY

RACHANA MOGILI

23EG505F01

DATE OF SUBMISSION: AUGUST 5TH, 2024

TABLE OF CONTENTS:

1. EXECUTIVE SUMMARY
   1.1. SIMPLE REGISTRATION FOR USING TKINTER
   1.1.1. PURPOSE
   1.2. BUILDING A CHATBOX
   1.2.1. PURPOSE
   1.3. CREATING A GUI TO EXTRACT LYRICS FROM SONGS
   1.3.1. PURPOSE
2. INTRODUCTION
   2.1. SIMPLE REGISTRATION FORM
   2.2. BUILDING A CHATBOT USING PYTHON
   2.3. CREATING GUI TO EXTRACT LYRICS FROM SONGS
   2.4. KEY COMPONENTS OF LYRICS EXTRACTION
3. TASK DESCRIPTIONS
   3.1. TASK-1: SIMPLE REGISTRATION FORM
   3.1.1. IMPLEMENTATION
   3.2. TASK-2: BUILDING A CHATBOT:
   3.2.1. INSTALLATION
   3.3. TASK-3: CREATING A GUI TO EXTRACT LYRICS FROM SONGS
3.3.1. EXTRACT LYRICS APPLICATION WITH TKINTER:
4. CONCLUSION

1. EXECUTIVE SUMMARY

This executive summary provides an overview of three distinct projects implemented using python: a simple registration form using Tkinter, building a chatbot, and creating a GUI to extract lyrics from songs. Each project showcases Python's versatility and its applications in GUI development and natural language processing.

1.1. SIMPLE REGISTRATION FORM USING TKINTER

1.1.1. PURPOSE:

. To collect user information such as name, email, and password.

. To store and manage user data securely.

1.2. BUILDING A CHATBOX

1.2.1. PURPOSE:

. To automate interactions with users for customer support, information retrieval, or entertainment.

1.3. CREATING A GUI TO EXTRACT LYRICS FROM SONGS

1.3.1. PURPOSE:

. To retrieve lyrics automatically for analysis, display, or other applications.

2. INTRODUCTION

2.1. SIMPLE REGISTRATION FORM

A simple registration form using python can be created to collect and manage user information. This typically involves a graphical user interface (GUI) to input data, and backend to handle data storage. The form can be implemented using various libraries, such as Tkinter for desktop applications or Flask/Django for web applications.

For instance, using Tkinter , python’s standard GUI library, one can build a straightforward desktop application that includes fields for the user’s email, phone number, and password.

2.2. BUILDING A CHATBOT USING PYTHON

Building a chatbot using python is an exciting and practical that can be applied in various domains, such as customer service, information retrieval, and personal assistance. Python's simplicity and extensive libraries make it an ideal language for developing chatbots.

A chatbot is a software application designed to stimulate human conversation through text or voice interactions. Chatbots can be as simple as rule-based systems that follow predefined scripts or as complex as AI-riven models that leverage natural language processing (NLP) and machine learning (ML) to understand and respond to user inputs more naturally and accurately.

2.3. CREATING GUI TO EXTRACT LYRICS FROM SONGS

Extracting lyrics from songs is a fascinating project that involves retrieving textual data from music tracks. This can be particularly useful for various applications, such as creating karaoke systems, building music analysis tools, or developing apps that display lyrics alongside playing, music. Python, with its vast array of libraries and tools, makes it relatively straightforward to automate the process of lyrics extraction.

2.4. KEY COMPONENTS OF LYRICS EXTRACTION

1. APIs and Web Scraping
2. Python Libraries and Tools
3. APIs

3. TASK DESCRIPTIONS

3.1. TASK-1: SIMPLE REGISTRATION FORM

Tkinter is python’s standard GUI (Graphical User Interface) library and is a great tool for creating simple desktop applications. Here, we will create a simple registration form using Tkinter to collect user information.

Python provides the Tkinter toolkit to develop GUI applications. Now, it’s up to the imagination or necessary of developer, what he/she want to develop using this toolkit. Let's make a simple registration form using GUI applications using Tkinter. In this application, user has to fill up the required information, and that information is automatically written into an excel file.

Firstly, create an empty excel file, after that pass an absolute path of the excel file in the program so that the program is able access that excel file.

3.1.1. IMPLEMENTATION

```
From openpyxl import *

From tkinter import *

wb=load_workbook(‘C:\\Users\\Admin\\Desktop\\excel.xlsx’)

Sheet=wb.active

def excel():

    sheet.column_dimensions[‘A’]. width=30

    sheet.column_dimensions[‘B’]. width=10

    sheet.column_dimensions[‘C’]. width=10

    sheet.column_dimensions[‘D’]. width=20

    sheet.column_dimensions[‘E’]. width=20

    sheet.column_dimensions[‘F’]. width=40

    sheet.column_dimensions[‘G’]. width=50

    sheet. cell (row=1, column=1).value=” Name”

    sheet. cell (row=1, column=2).value=” Course”

    sheet. cell (row=1, column=3).value=” semester”
    sheet. cell (row=1, column=4).value=” Form Number”
    sheet. cell (row=1, column=5).value=” Contact Number”
    sheet.cell(row=1, column=6).value=” Email id”
    sheet.cell(row=1, column=7).value=” Address
def focus1(event):
     course_field.focus_set()
def focus2(event):
     sem_field.focus_set()
def focus3(event):
     form_no_field.focus_set()
def focus4(event):
     contact_no_fiels.focus_set()
def focus5(event):
     email_id_field.focus_set()
def focus6(event):
     address_field.focus_set()
def clear ():
     name_fiels.delete(0, END)
     Course_field.delete(0, END)
     Sem_field.delete(0, END)
     Form_no_field.delete(0, END)
     Contact_no_field.delete(0, END)
     Email_id_field.delete(0, END)
     Address_field.delete(0, END)
def insert ():
     if(name_field.get()== "“and
        Course_field.get() == “” and
         Sem_field.get() == “” and
```

```
           form_no_field.get() == “” and
           Contact_no_field.get() == “” and
           Email_id_field.get() == “” and
           Address_field.get() == “” and
          Print (“empty input”)
   else:
          Current_row =  sheet.max_row
          Current_column =  sheet.max_column
          Sheet.cell(row=current_row + 1, column=1).value=name_field.get()
          Sheet.cell(row=current_row + 1, column=2).value=course_field.get()
          Sheet.cell(row=current_row + 1, column=3).value=sem_field.get()
          Sheet.cell(row=current_row + 1, column=4).value=form_no_field.get()
          Sheet.cell(row=current_row + 1, column=5).value=contact_no_field.get()
          Sheet.cell(row=current_row + 1, column=6).value=email_id_field.get()
          Sheet.cell(row=current_row + 1, column=7).value=address_field.get()
         Wb.save(‘C:\\Users\\Admin\\Desktop\\excel.xlsx’)
          Name_field.focus_set()
          Clear ()
__name__==” ==main__”:
Root=Tk ()
Root.configure(background=’light green’)
Root.title(“registration form”)
Root.geometry(“500*300”)
Excel ()
Heading=label (root, text=” form”, bg=”light green”)
Name=label (root, text=”name”, bg=”light green”)
Course=label (root, text=” course” bg=”light green”)
Sem=label (root, text=”sem” bg=”light green”)
```

```
Form-no=label (root, text=”form_no”, bg=”light green”)
Contact_no=label (root, text=”contact_no”, bg=”light green”)
Email_id=label (root, text=”email_id”, bg=”light green”)
Address=label (root, tect=”address”, bg=”light green”)
Heading.grid(row-0, column=1)
Name.grid(row=1, column=0)
Course.grid(row=2, column=0)
Sem.grid(row=3, column=0)
Form_no.grid(row=4, column=0)
Contact_no.grid(row=5, column=0)
Email_id.grid(row=6, column=0)
Address.grid(row=7, column=0)

Name_field=entry(root)
Course_field=entry(root)
Sem_field=entry(root)
Form_no_field=entry(root)
Contact_no_field=entry(root)
Email_id_field=entry(root)
Address_field=entry(root)

Name_field.bind(“<return>”, focus1)
Course_field.bind(“<return>”, focus2)
Sem_field.bind(“<return>”, focus3)
Form_no_bind(“<return>”, focus4)
Contact_no_field(“return>”, focus5)
Email_id_field.bind(“<return>”, focus6)
```

```
name_field.grid(row=1, column=1, ipadx="100")

    course_field.grid(row=2, column=1, ipadx="100")

    sem_field.grid(row=3, column=1, ipadx="100")

    form_no_field.grid(row=4, column=1, ipadx="100")

    contact_no_field.grid(row=5, column=1, ipadx="100")

    email_id_field.grid(row=6, column=1, ipadx="100")

    address_field.grid(row=7, column=1, ipadx=”100”)

    excel ()

    submit = Button (root, text="Submit", fg="Black",

                            bg="Red", command=insert)

    submit.grid(row=8, column=1)

    root.mainloop()
```

3.2. TASK-2: BUILDING A CHATBOT:

Nobody likes to be alone always, but sometimes loneliness could be a better medicine to hunch the thirst for a peaceful environment. Even during such lonely quarantines, we may ignore humans but not humanoids. Yes, if you have guessed this article for a chatbot, then you have cracked it right. We won't require 6000 lines of code to create a chatbot but just a six-letter word "Python" is enough. Let us have a quick glance at Python's Chatterbot to create our bot. Chatterbot is a Python library built based on machine learning with an inbuilt conversational dialog flow and training engine. The bot created using this library will get trained automatically with the response it gets from the user.

Why Chatbots are important for a Business or a Website

Quick resolution for a complaint or a problem.

Improve business branding thereby achieving great customer satisfaction.

Answering questions and answers for customers.

Making a reservation at hotel or at restaurant.

Save human effort 24×7.

Enhance business revenue by providing ideas and inspirations.

Finding details about business such as hours of operation, phone number and address.

Automate sales and lead generation process.

Reduce customer agents waiting time answering phone calls.

- **TYPES OF CHATBOT:**

Chatbots deliver instantly by understanding the user requests with pre-defined rules and AI based chatbots. There are two types of chatbots.

- **RULE BASED CHATBOT:**

This type of chatbots answer the customer queries using the pre-defined rules. These bots answer common queries such as hours of operation of business, addresses, phone numbers and tracking status.

- **CONVERSATIONAL AI CHATBOT:**

This type of chatbots using Natural language Processing (NLP) to understand the context and intent of a user input before providing the response. These Bots train themselves as per the user inputs and more they learn, more they become user interactive.

3.2.1. INSTALLATION

Install chatterbot using Python Package Index(PyPi) with this command

```
pip install chatterbot
```

Below is the implementation.

```
# Import "chatbot" from
# chatterbot package.

from chatterbot import ChatBot



# Inorder to train our bot, we have
# to import a trainer package
# "ChatterBotCorpusTrainer"

from chatterbot.trainers import ChatterBotCorpusTrainer




# Give a name to the chatbot “corona bot”
# and assign a trainer component.
```

```
chatbot=ChatBot('corona bot')

# Create a new trainer for the chatbot

trainer = ChatterBotCorpusTrainer(chatbot)


# Now let us train our bot with multiple corpus

trainer.train("chatterbot.corpus.english.greetings",

              "chatterbot.corpus.english.conversations" )



response = chatbot.get_response('What is your Number')

print(response)


response = chatbot.get_response('Who are you?')

print(response)
```

3.3. TASK-3: CREATING A GUI TO EXTRACT LYRICS FROM SONGS

In this article, we are going to write a python script to extract lyrics from the song and bind with its GUI application. We will use lyrics-extractor to get lyrics of a song just by passing in the song name, it extracts and returns the song's title and song lyrics from various websites. Before starting, install the lyrics-extractor module. Run this command into your terminal.

pip install lyrics-extractor

Requirements

Need an API Key and Engine ID of Google Custom Search JSON API.

Engine ID

Create a Custom Search Engine to get your Engine ID here.

We have to create our own Programmable Search Engine (Google Custom Search Engine) and add Link to fetch lyrics.

Programmable Search Engine is based on Google's core search technology.

It is a search engine for your website and has a task to find information as the user choose.

Choose any link of one to get your search engine:

https://genius.com/

http://www.lyricsted.com/

http://www.lyricsbell.com/

https://www.glamsham.com/

http://www.lyricsoff.com/

http://www.lyricsmint.com/

JSON API:

The custom search JSON API is able to retrieve and display search result from Programmable Search Engine.

To use the custom search JSON API, we have to create Programmable Search Engine.

Visit here to get your API key.

Approach:

Import the modules.

```
from lyrics_extractor import SongLyrics
```

Pass the Google Custom Search JSON API key and Engine ID into SongLyrics().

```
extract_lyrics = SongLyrics(Your_API_KEY, GCS_ENGINE_ID)
```

Get the lyrics by passing the song name as a parameter to extract_lyrics.get_lyrics() method.

```
extract_lyrics.get_lyrics("Shape of You")
```

3.3.1. EXTRACT LYRICS APPLICATION WITH TKINTER:

```
# Import modules
from tkinter import *
from lyrics_extractor import SongLyrics


# user defined function
def get_lyrics():


    extract_lyrics = SongLyrics(
        "Aerwerwefwdssdj-nvN3Oq0oTixw", "werwerewcxzcsda")


    temp = extract_lyrics.get_lyrics(str(e.get()))
    res = temp['lyrics']
    result.set(res)



# object of tkinter
# and background set to light grey
master = Tk ()
master.configure(bg='light grey')

# Variable Classes in tkinter
result = StringVar()

# Creating label for each information
# name using widget Label
Label (master, text="Enter Song name: ",
      bg="light grey"). grid (row=0, sticky=W)

Label (master, text="Result:",
      bg="light grey"). grid (row=3, sticky=W)



# Creating label for class variable
# name using widget Entry
Label (master, text="", textvariable=result,
      bg="light grey"). grid (row=3, column=1, sticky=W)

e = Entry (master, width=50)
e.grid(row=0, column=1)
```

```
# creating a button using the widget
b = Button(master, text="Show",
           command=get_lyrics, bg="Blue")

b.grid(row=0, column=2, columnspan=2,
       rowspan=2, padx=5, pady=5,)

mainloop()
```

4. CONCLUSION:

In conclusion, each of these tasks leverages Python's versatility and a combination of different libraries and frameworks to achieve specific goals in web development, conversational AI, and data extraction.