**Senapati's Song Lyrics Extractor: Documentation**

**Overview**

Senapati's Song Lyrics Extractor is a Python-based GUI application that allows users to fetch the lyrics of any song by providing the song name and artist name. This application uses the lyrics.ovh API to retrieve lyrics and displays the result in the command terminal while providing feedback through a GUI interface built with Tkinter.

**Features**

* **User-Friendly Interface**: Easy-to-use GUI for entering song details.
* **Real-Time Lyrics Fetching**: Retrieves lyrics quickly from an online database.
* **Error Handling**: Notifies users of network errors, incorrect input, or unavailable songs.
* **Platform Compatibility**: Works on Windows, macOS, and Linux as long as Python is installed.

**Requirements**

* Python 3.x
* Libraries:
  + requests (for HTTP requests)
  + tkinter (for the GUI)

**Installation of Required Libraries**

To install the requests library, use the following command:

pip install requests

Note: tkinter comes pre-installed with Python on most systems.

**Application Usage**

1. **Run the Script**: Execute the Python script in a terminal or command prompt.
2. **Enter Details**: Enter the song name and artist's name in the provided input fields.
3. **Fetch Lyrics**: Click the "Extract lyrics" button to initiate the lyrics search.
4. **View Results**: If the lyrics are found, they will be printed in the terminal, and a confirmation message will appear on the GUI.

**Code Walkthrough**

**1. Importing Libraries**

from tkinter import \*

import tkinter.messagebox as mb

import json

import requests

* **tkinter**: The standard Python interface to the Tk GUI toolkit.
* **tkinter.messagebox**: Provides various pop-up dialogs to show information or error messages.
* **json**: To handle the JSON response from the lyrics API.
* **requests**: Allows HTTP requests to fetch lyrics from the lyrics.ovh API.

**2. Function: extract\_lyrics**

def extract\_lyrics():

global artist, song

artist\_name = str(artist.get())

song\_name = str(song.get()).lower()

link = 'https://api.lyrics.ovh/v1/' + artist\_name.replace(' ', '%20') + '/' + song\_name.replace(' ', '%20')

* **artist\_name and song\_name**: Retrieves the input values from the GUI fields and formats them for the API request.
* **link**: Constructs the API request URL by replacing spaces with %20 for URL encoding.

**3. Sending the API Request and Handling the Response**

try:

req = requests.get(link)

req.raise\_for\_status()

json\_data = req.json()

lyrics = json\_data.get('lyrics', None)

if lyrics:

print(lyrics)

mb.showinfo('Lyrics printed', 'The lyrics to the song you wanted have been extracted and printed on your command terminal.')

else:

raise ValueError('Lyrics not found')

except requests.exceptions.RequestException as e:

mb.showerror('Network Error', f'An error occurred while trying to fetch the lyrics: {e}')

except Exception as e:

mb.showerror('No such song found', 'We were unable to find such a song in our directory. Please recheck the name of the artist and the song, and if correct, we apologize because we do not have that song available with us.')

* **requests.get(link)**: Sends an HTTP GET request to the API.
* **req.raise\_for\_status()**: Raises an error for HTTP status codes 4xx or 5xx.
* **json\_data.get('lyrics', None)**: Extracts the lyrics from the JSON response.
* **Error Handling**:
  + **Network Errors**: Displays a message box for any network-related issues.
  + **Other Exceptions**: Displays a message if the song or artist is not found.

**4. Creating the GUI**

root = Tk()

root.title("Senapati's Song Lyrics Extractor")

root.geometry("600x200")

root.resizable(0, 0)

root.config(bg='CadetBlue')

* **Tk()**: Initializes the main application window.
* **title**: Sets the title of the application window.
* **geometry**: Specifies the size of the window.
* **resizable(0, 0)**: Disables resizing of the window.
* **config(bg='CadetBlue')**: Sets the background color of the window.

**5. Adding GUI Components**

Label(root, text="Senapati's Song Lyrics Extractor", font=("Comic Sans MS", 16, 'bold'), bg='CadetBlue').pack(side=TOP, fill=X)

Label(root, text='Enter the song name: ', font=("Times New Roman", 14), bg='CadetBlue').place(x=20, y=50)

song = StringVar()

Entry(root, width=40, textvariable=song, font=('Times New Roman', 14)).place(x=200, y=50)

Label(root, text="Enter the artist's name: ", font=("Times New Roman", 14), bg='CadetBlue').place(x=20, y=100)

artist = StringVar()

Entry(root, width=40, textvariable=artist, font=('Times New Roman', 14)).place(x=200, y=100)

Button(root, text='Extract lyrics', font=("Georgia", 10), width=15, command=extract\_lyrics).place(x=220, y=150)

* **Labels**: Provide text for instructions to the user.
* **Entry Fields**: Allow users to input the song name and artist name.
* **Button**: A button to trigger the extract\_lyrics function when clicked.

**6. Running the Main Event Loop**

root.mainloop()

**root.mainloop()**: Starts the Tkinter event loop, which waits for user interaction and keeps the window open.

**Usage Examples**

* **Example 1**:
  + **Song Name**: "Anyone Who Loves Me"
  + **Artist Name**: "Charlotte Cardin"
* **Example 2**:
  + **Song Name**: "Shape of You"
  + **Artist Name**: "Ed Sheeran"
* **Example 3**:
  + **Song Name**: "Love Me"
  + **Artist Name**: "Justin Bieber"

**Error Handling Scenarios**

* **Network Issues**: Displays a message when there's a network error or the server is unreachable.
* **Invalid Song/Artist Names**: Alerts the user when the input song or artist name is incorrect or unavailable.
* **Missing Lyrics**: Informs the user if lyrics are not found in the API database.

**Potential Future Enhancements**

* **Save Lyrics**: Add functionality to save the lyrics to a file.
* **Multi-API Support**: Integrate multiple lyrics APIs for a wider database.
* **Enhanced UI**: Improve the graphical interface with more styling options.