Practical No 3

Aim:-Develop a youtube Dowloader the allows users to download videos in multiple formats and quality options for offline access. The solution should be user friendly ,efficient and legally compliant while ensuring fast downloads and metadata preservation.

Theory:- **YouTube Downloader - Theory**

A **YouTube Downloader** is a software tool that allows users to download videos and audio from YouTube for offline access. The application should be:

1. **User-Friendly:** Simple UI for easy navigation.
2. **Efficient:** Fast downloads with multiple quality options.
3. **Legally Compliant:** Should follow YouTube’s terms of service.
4. **Metadata Preservation:** Saves thumbnails and video details.

**How It Works:**

1. **User Input:** Enter a YouTube URL.
2. **Quality Selection:** Choose between Best Video, Worst Video, or Best Audio.
3. **Format Selection:** Select mp4 (video) or mp3 (audio).
4. **Download Process:**
   * Uses yt\_dlp to fetch video/audio.
   * Saves the file in a specified folder.
   * Extracts metadata and thumbnails.
5. **Completion:** Displays a success or error message.

This tool ensures quick and customizable downloads while maintaining efficiency. 🚀

Code:-

import yt\_dlp

import os

import tkinter as tk

from tkinter import messagebox, ttk

import pyperclip

# Set the download folder

DOWNLOAD\_FOLDER = r'C:\Users\Asus\OneDrive\Desktop\Code\CapStone\prac3\youtube\_downloads'

if not os.path.exists(DOWNLOAD\_FOLDER):

    os.makedirs(DOWNLOAD\_FOLDER)

# Function to download video/audio

def download\_video():

    url = url\_entry.get().strip()

    if not url:

        messagebox.showerror("Input Error", "Please enter a valid YouTube URL.")

        return

    quality = quality\_var.get()

    file\_format = format\_var.get()

    ydl\_opts = {

        'outtmpl': os.path.join(DOWNLOAD\_FOLDER, '%(title)s.%(ext)s'),

        'writethumbnail': True,  # Preserve thumbnail

        'writeinfojson': True,  # Preserve metadata

        'quiet': True,  # Suppress console output

    }

    # Automatically select best available format

    if quality == "Best Video":

        ydl\_opts['format'] = f"bv\*+ba/b"  # Best video + best audio, fallback to best single format

    elif quality == "Worst Video":

        ydl\_opts['format'] = f"wv\*+ba/w"  # Worst video + best audio, fallback to worst single format

    elif quality == "Best Audio":

        ydl\_opts['format'] = 'bestaudio'

        ydl\_opts['postprocessors'] = [{

            'key': 'FFmpegExtractAudio',

            'preferredcodec': file\_format,

            'preferredquality': '192'

        }]

    try:

        with yt\_dlp.YoutubeDL(ydl\_opts) as ydl:

            ydl.download([url])

        messagebox.showinfo("Success", f"Download completed!\nCheck: {os.path.abspath(DOWNLOAD\_FOLDER)}")

    except Exception as e:

        messagebox.showerror("Download Error", f"An error occurred: {str(e)}")

# Function to paste clipboard content

def paste\_from\_clipboard():

    url\_entry.delete(0, tk.END)

    url\_entry.insert(0, pyperclip.paste())

# Tkinter UI Setup

root = tk.Tk()

root.title("YouTube Downloader")

root.geometry("400x350")

tk.Label(root, text="Enter YouTube URL:").pack(pady=5)

url\_entry = tk.Entry(root, width=50)

url\_entry.pack(pady=5)

tk.Button(root, text="Paste from Clipboard", command=paste\_from\_clipboard).pack(pady=5)

# Quality Selection

tk.Label(root, text="Select Quality:").pack(pady=5)

quality\_var = tk.StringVar(value="Best Video")

quality\_menu = ttk.Combobox(root, textvariable=quality\_var, values=["Best Video", "Worst Video", "Best Audio"])

quality\_menu.pack(pady=5)

# Format Selection

tk.Label(root, text="Select Format:").pack(pady=5)

format\_var = tk.StringVar(value="mp4")

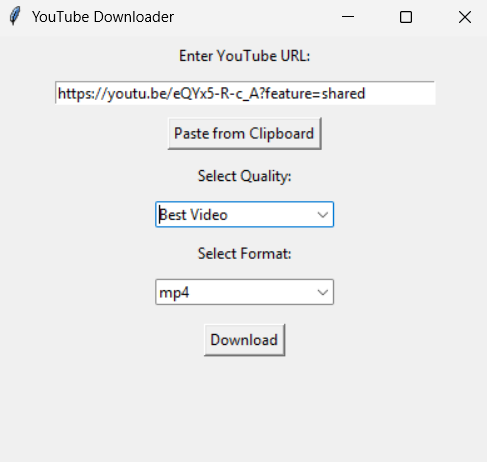
format\_menu = ttk.Combobox(root, textvariable=format\_var, values=["mp4", "mp3"])  # Removed "webm"

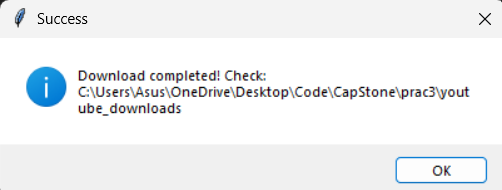
format\_menu.pack(pady=5)

# Buttons

tk.Button(root, text="Download", command=download\_video).pack(pady=10)

root.mainloop()





**Conclusion**

The YouTube Downloader provides a simple, efficient, and user-friendly solution for downloading YouTube videos and audio in multiple formats and quality options. By leveraging yt\_dlp, it ensures fast downloads while preserving metadata and thumbnails. This tool enhances offline accessibility while maintaining compliance with YouTube's policies.