SCHOOL OF COMPUTER SCIENCE AND APPLICATIONS

A Project Report
On

YouTube Video Downloader by using Python

as part of Python Programming Lab

Master of Computer Applications

Submitted by

1. **D. Datta Sai Malleswarao** **SRN: R22DE033**
2. **B. Venkata Suman** **SRN: R22DE019**
3. **A. Ayyappa Swami** **SRN: R22DE004**

Faculty Incharge

Prof/Dr.Mohamed Abdul KadarJailani

November' 2023

Rukmini Knowledge Park, Kattigenahalli, Yelahanka, Bengaluru-560064
www.reva.edu.in

SCHOOL OF COMPUTER SCIENCE AND APPLICATIONS

CERTIFICATE

The project work titled – YouTube Video Downloader is being carried out under our guidance by D.Datta Sai Malleswararao(R22DE033), A.Ayyappa Swami(R22DE004),B.Venkata Suman(R22DE019) a bonafied student at REVA University, and is submitting the project report in partial fulfillment of Python Programming Lab, for the award of **Master of Computer Applications** during the academic year **2022–23**. The project report has been approved, as it satisfies the academic requirements with respect to the Project Work prescribed for the aforementioned Degree.

Signature with date		**Signature with date**
Internal Guide		**Academic Vertical Head-PG**

Table of Contents

INTRODUCTION

Title: "Ultimate YouTube Video Downloader - Your Gateway to Entertainment"

Introduction:

Welcome, YouTube enthusiasts! Are you tired of buffering issues, unreliable internet connections, or just want to watch your favorite content offline, without ads? Well, you've come to the right place. In today's video, we're going to introduce you to the ultimate YouTube video downloader by using python that will change the way you enjoy your favorite videos forever.

YouTube is a treasure trove of content, from educational tutorials and music videos to your daily dose of entertainment. However, there are times when you want to watch these videos without interruptions, or you might not have access to the internet. That's where our YouTube video downloader comes in to save the day.

In this video, we will guide you through the process of downloading YouTube videos with ease, ensuring you have access to your favorite content whenever and wherever you want. We'll also address the legal aspects and ethics of downloading YouTube videos, so you can stay on the right side of the law.

By the end of this video, you'll be well-equipped with the knowledge and tools you need to download YouTube videos responsibly and enjoy your content without any hassles.

So, whether you're planning a road trip, looking to build your offline video library, or just want to save your data, stick around as we introduce you to the best YouTube video downloader that will transform your YouTube experience. Let's dive right in!

ABSTRACT

YouTube Video Downloader using Python

YouTube is a well-known internet video streaming service. There are millions of videos in categories such as education, entertainment, and travel.

You can quickly watch videos with a few mouse clicks, but downloading videos is difficult. But in a recent upgrade, YouTube now allows you to save videos in its download folder for offline viewing. Still, you are unable to save them locally.

In this Project, you will learn how to use Python code to download YouTube videos

TOOLS AND TECHNOLOGIES:

HARDWARE REQIREMENTS:

-Processor 1.66GHZ

-RAM 256GB

SOFTWARE REQIREMENTS:

-Python 3.11 version

-PyTube module

-Visual Studio code

-Python.exe

EXISTING SYSTEM

1. **User Interface**: This encompasses the user interface or the website through which users interact with the YouTube video downloader. It includes the design, layout, and features available to users for downloading videos.

2. **Downloading Mechanism**: The technology or methods used for video downloading from YouTube. This can include various algorithms and techniques to retrieve video content from YouTube servers.

3. **Video Processing**: How the video content is processed and converted into downloadable formats. It may include the available output formats (e.g., MP4, AVI, MP3) and video quality options.

4. **Authentication and Authorization**: The system's handling of YouTube's terms of service, including user authentication and authorization mechanisms, to ensure compliance with YouTube's policies.

5. **Error Handling**: How errors and exceptions are managed, including how the system handles issues like broken links, unavailable videos, or changes in YouTube's APIs.

6. **Security**: The security measures in place to protect the system and users from potential threats, such as malware, viruses, and unauthorized access.

7. **User Interaction and Feedback**: The existing system may include features for user feedback, reporting issues, or providing suggestions for improvements.

8. **Legal Compliance**: Whether the existing system complies with copyright laws and YouTube's terms of service, as unauthorized video downloading can raise legal concerns.

9. **Performance Metrics**: Any data or analytics collected on system performance, such as download speed, success rates, and user statistics.

PROPOSED SYSTEM

In this project, I will be developing a YouTube video downloader with a focus on innovation and user convenience. While YouTube video downloaders already exist, I aim to bring a fresh perspective by implementing several novel features and improvements.

Firstly, I plan to incorporate a smart suggestion system that helps users discover related videos they might want to download, based on their preferences and viewing history. This feature will make the downloader not just a tool for saving videos but also a means of content discovery.

Secondly, I intend to include an intuitive, user-friendly interface with support for multiple platforms, including web, desktop, and mobile applications. This cross-platform accessibility will enhance the user experience and make video downloading more accessible to a wider audience.

Furthermore, I will prioritize maintaining the project's codebase and dependencies to ensure consistent functionality even as YouTube's website evolves. This is vital because YouTube frequently updates its site structure, which can break existing downloaders.

To develop this YouTube video downloader, I will use a combination of programming languages, web scraping techniques, and a robust backend server. Python will be used for web scraping and processing video data, while the frontend will likely involve HTML, CSS, and JavaScript for building a responsive user interface. I will also implement secure and efficient downloading methods, such as utilizing YouTube's API when possible to ensure compliance with their terms of service.

In summary, this project aims to stand out by offering a combination of unique features, cross-platform accessibility, and robust maintenance strategies, all while ensuring compliance with YouTube's policies. The result will be a YouTube video downloader that not only provides a reliable and user-friendly experience but also offers innovative features for a more enriched downloading and content discovery experience.

MODULE DESCRIPTION

YouTube is very popular video sharing website. Downloading a video from YouTube is a tough job. Downloading the Downloader and get the video using that or go to any other website which fetches the video and saves on your computer. Using Python, this task is very easy. Few lines of code will download the video from YouTube for you. For this, there a python library named as 'pytube'. pytube is a lightweight, dependency-free Python library which is used for downloading videos from the web. pytube is not the native library. You need to install it before using it. Installation is easy when you have pip.

In the Terminal or Command Prompt, type the following command to install Pytube.

- **Pip install Pytube**

pytube library makes the video downloading very easy. Create the object of the YouTube module by passing the link as the parameter. Then, get the appropriate extension and resolution of the video. You can set the name of the file as your convenience, in another case original name will be kept. After that, download the file using the download function which has one parameter which is the location where to download the file.

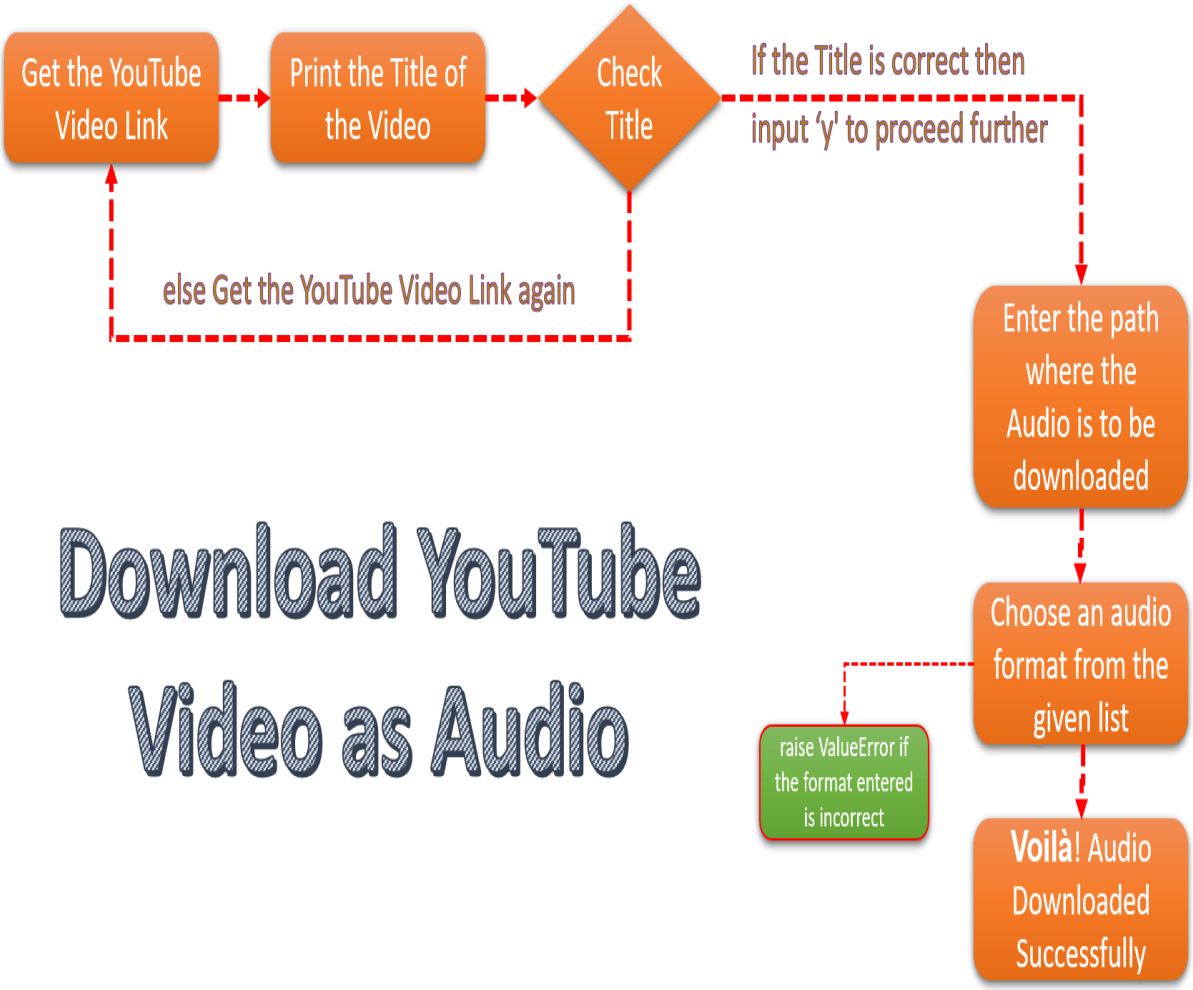
YouTube Video downloader using Tkinter:

In the context of developing a YouTube video downloader using Python, Tkinter plays a crucial role as a GUI module. Tkinter is utilized to create the graphical user interface for the downloader application, allowing users to interact with the software seamlessly. With Tkinter, you can design and implement essential components like buttons, labels, input fields, and windows that make up the user interface of the video downloader. This module simplifies the process of building an intuitive and user-friendly interface where users can input video URLs, select download options, and initiate the download process. Tkinter's cross-platform compatibility ensures that the YouTube video downloader can run on various operating systems with a consistent user experience. This integration of Tkinter into the project enhances the accessibility and usability of the video downloading application.

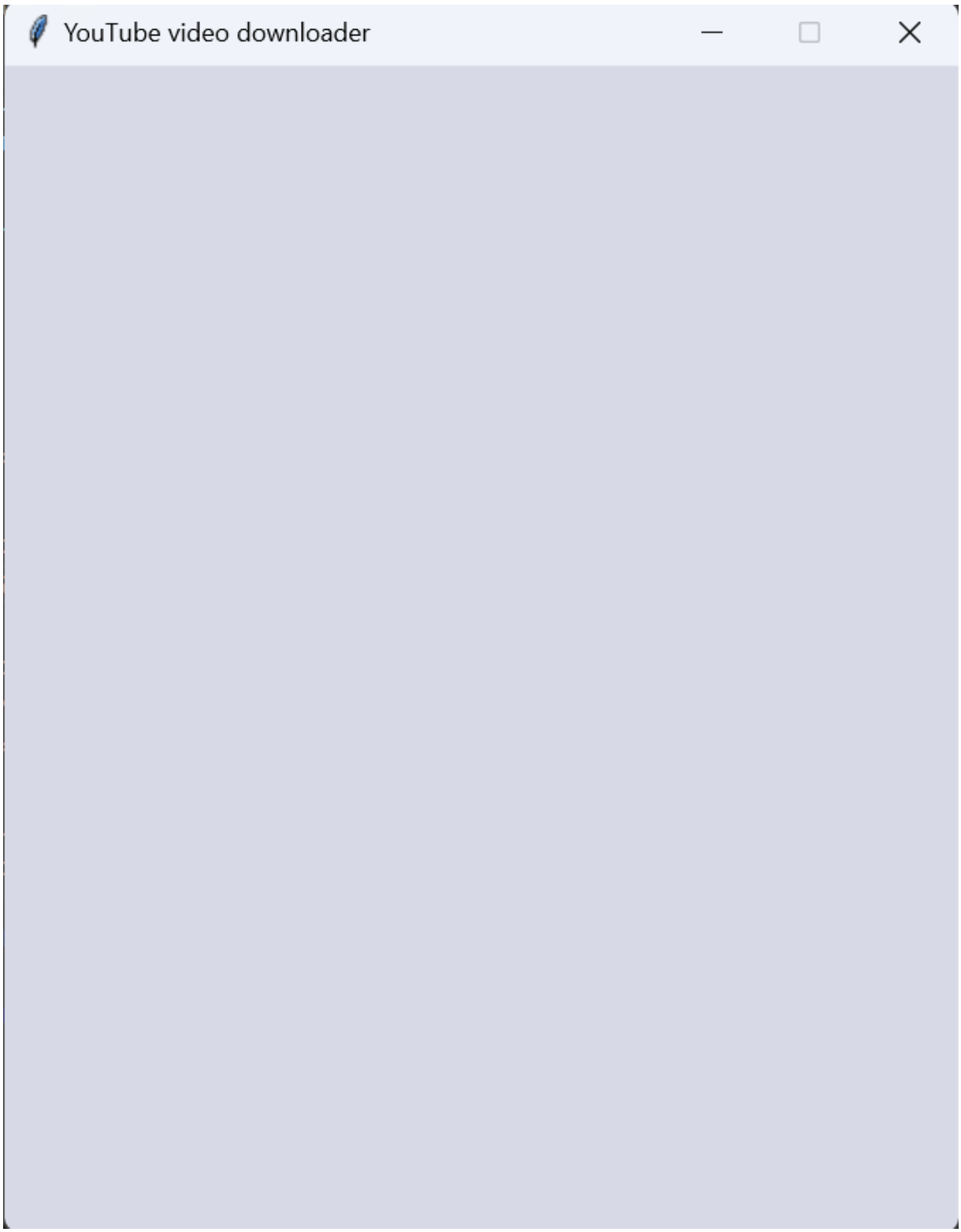
```
import tkinter as tk

from pytube import YouTube
```

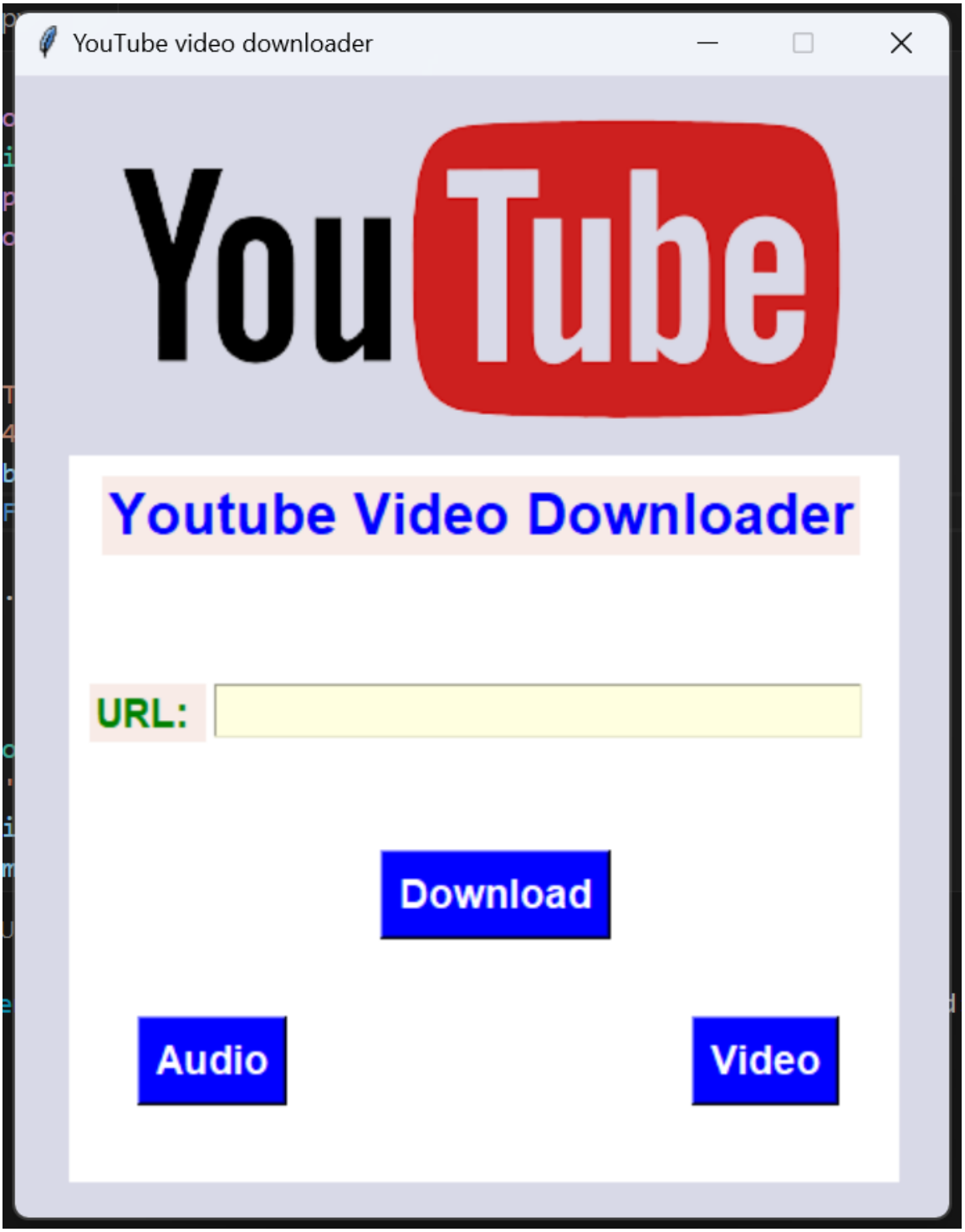
SCREEN DESIGN

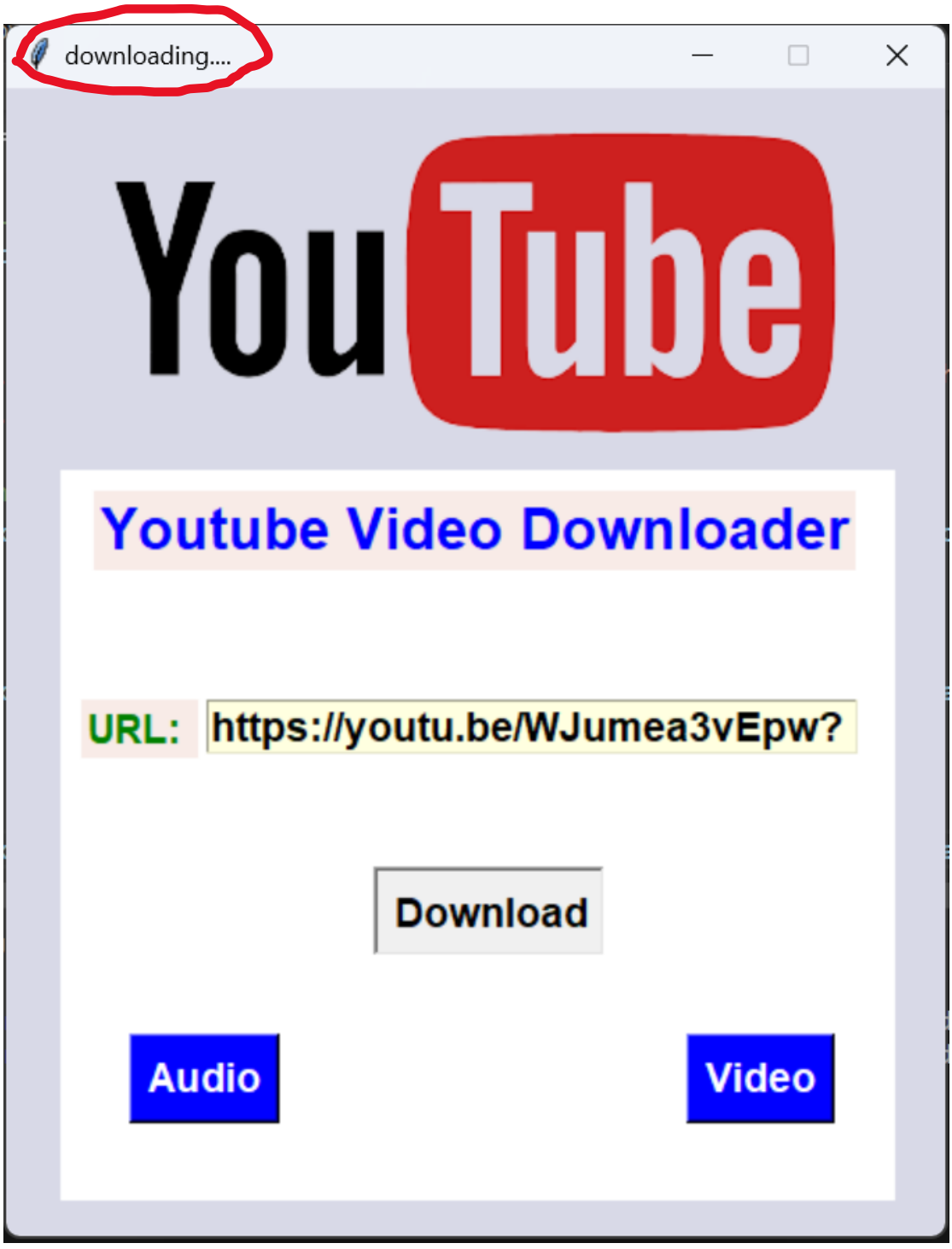


SCREEN DESIGN




YouTube video downloader
YouTube


YouTube video downloader
YouTube
Youtube Video Downloader
URL:
Download
Audio
Video


downloading....
YouTube
Youtube Video Downloader
URL:
https://youtu.be/WJumea3vEpw?
Download
Audio
Video


Successfully downloaded
YouTube
Youtube Video Downloader
URL:
https://youtu.be/WJumea3vEpw?
Download
Audio
Video

CODE

Normal code for YouTube Video downloader using Python:

```python
#code for Youtube video download
'''YouTube Video Downloader'''

#import the package
from pytube import YouTube

url = 'Your URL goes here'
my_video = YouTube(url)

print("**********Video Title**********")
#get Video Title
print(my_video.title)

print("**********Tumbnail Image**********")
#get Thumbnail Image
print(my_video.thumbnail_url)

print("**********Download video**********")
#get all the stream resolution for the
for stream in my_video.streams:
    print(stream)

#set stream resolution
my_video = my_video.streams.get_highest_resolution()

#or
#my_video = my_video.streams.first()

#Download video
my_video.download()
```

Normal code for YouTube Video downloader using Python:

```
#code for Youtube video download
'''YouTube Video Downloader'''
#import the package
from pytube import YouTube
url = 'Your URL goes here'
my_video = YouTube(url)
print("**********Video Title**********")
#get Video Title
print(my_video.title)
print("**********Tumbnail Image**********")
#get Thumbnail Image
print(my_video.thumbnail_url)
print("**********Download video**********")
#get all the stream resolution for the
for stream in my_video.streams:
    print(stream)
#set stream resolution
my_video = my_video.streams.get_highest_resolution()
#Download video
my_video.download()
```

Code for GUI YouTube Video downloader using Python:

```
#code for GUI YouTube Video downloader

from tkinter import *
from pytube import YouTube
from moviepy.editor import VideoFileClip
from turtle import bgcolor
from moviepy.editor import VideoFileClip
from tkinter import *
from pytube import YouTube

root = Tk()
root.title('YouTube video downloader')
root.geometry('450x550+400+50')
root.configure(bg='#d8d9e7')
root.resizable(False,False)
def download():
    v_link=link.get('1.0','end')


    #download
    my_video= YouTube(v_link)
    root.title('downloading....')
    stream=my_video.streams.get_highest_resolution()
    video=stream.download()
    root.title('Successfully downloaded')
    clip=VideoFileClip(video)
    clip.close()
    pass
```

```
#for audio
def audio():
    v_link=link.get('1.0','end')



    #download
    my_video= YouTube(v_link)
    root.title('downloading....')
    stream=my_video.streams.filter(only_audio=True, file_extension='mp4').first()
    video=stream.download()
    root.title('Successfully downloaded')
    clip=VideoFileClip(video)
    clip.close()
    pass

#for video only
def video():
    v_link=link.get('1.0','end')
    my_video= YouTube(v_link)
    root.title('downloading....')
    stream=my_video.streams.filter(only_video=True, file_extension='mp4').first()
    video=stream.download()
    root.title('Successfully downloaded')
    clip=VideoFileClip(video)
    clip.close()
    pass

#load image
img=PhotoImage(file='you.png')
Label(image=img,bd=1,bg='#d8d9e7').place(x=50,y=20)
```

```
frame=Frame(root,width=400,height=350,bg='#fff')

frame.place(x=26,y=183)


#passing text with in a frame

heading=Label(frame,text='Youtube Video
Downloader',fg='blue',bg='#f8ebe7',font=('Microsoft yahie UI light',20,'bold'))

heading.place(x=16,y=10)


#passing text with in a frame

Label(frame,text='URL: ',fg='green',bg='#f8ebe7',font=('Microsoft yahie UI
light',14,'bold')).place(x=10,y=110)


#taking link

link=Text(frame,fg='black',bg='light yellow',font=('Microsoft yahie UI
light',14,'bold'),bd=1,height=1,width='28')

link.place(x=70,y=110)


#download Button

Button(frame,text='Download',padx=1,pady=3,border=2,fg='white',bg='blue',command=dow
nload,font=('Microsoft yahie UI light',14,'bold')).place(x=150,y=190)



#Audio Button

Button(frame,text='Audio',padx=1,pady=3,border=2,fg='white',bg='blue',command=audio,fo
nt=('Microsoft yahie UI light',14,'bold')).place(x=33,y=270)



#Video Button

Button(frame,text='Video',padx=1,pady=3,border=2,fg='white',bg='blue',command=video,fo
nt=('Microsoft yahie UI light',14,'bold')).place(x=300,y=270)


root.mainloop()
```

RESULTS

The provided Python script offers a straightforward method for downloading YouTube videos using the `pytube` library. The script begins by importing the `pytube` library, and it's crucial to ensure that you have this library installed via `pip`. You need to specify the URL of the YouTube video you intend to download, replacing `"https://www.youtube.com/watch?v=YOUR_VIDEO_ID"` with the actual video URL. It's essential to respect YouTube's terms of service and copyright laws, ensuring you have the proper rights to download the video.

The script creates a `YouTube` object using the provided URL, enabling you to access information about the video and its available streams. It then selects the highest resolution stream for downloading using the `get_highest_resolution()` method, ensuring that you get the best quality. To determine where the downloaded video is saved, set the `download_path` variable to your desired directory, ensuring that you have the necessary write permissions.

After setting the download location, the script uses the selected stream to initiate the download process with `stream.download()`. It wraps up by displaying a message indicating the completion of the download. In practice, it's advisable to implement error handling for issues like network problems or invalid URLs. Additionally, keep an eye on potential updates to the `pytube` library and adjust your script accordingly based on the latest documentation and best practices.

CONCLUSION

In conclusion, the Python script presented for downloading YouTube videos using the `pytube` library offers a simple and effective solution for obtaining video content from the platform. While the script is user-friendly and easy to implement, it's imperative to exercise caution and adhere to YouTube's terms of service and copyright regulations by ensuring you have the proper permissions to download the content.

This project serves as a practical example of leveraging Python libraries to automate a task that might be useful in various contexts, such as content creation, research, or educational purposes. To expand on this project, one could consider implementing additional features like batch downloading of multiple videos, integrating user input for video selection, or creating a user-friendly graphical interface. In any case, users should remain vigilant for potential updates or changes to the `pytube` library and reference its documentation for the latest information and guidelines. Overall, this project exemplifies the versatility of Python for handling web-related tasks and data acquisition.

REFERENCES

From:

- ✓ https://www.section.io/engineering-education/youtube-video-downloader-using-python/
- ✓ https://youtu.be/7BXJIjfJCsA?si=_I4sfWt-OJsJHFf7
- ✓ https://youtu.be/vWENQefNTTM?si=5lhE78b8kS8gi-Gz
- ✓ https://youtu.be/Oyp3EsyAvPo?si=Sc7grMbQcaqtFwHI

THANK YOU