A SYNOPSIS ON

YOUTUBE VIDEO DOWNLOADER

Submitted in partial fulfillment of the requirement for the award of the degree of

Master of Computer Application
Submitted by:

AYUSH RAWAT
(STD. ID: 22391138)

Under the Guidance of
VANDANA RAWAT
Assistant Professor

Department of Computer Science and Engineering Graphic Era (Deemed to be University) Dehradun, Uttarakhand

Graphic Era Deemed to be University DEHRADUN

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the Synopsis entitled "**YouTube Video Downloader**" in partial fulfillment of the requirements for the award of the Degree of Master of Computer Application in the Department of Computer Science and Engineering of the Graphic Era (Deemed to be University), Dehradun shall be carried out by the undersigned under the supervision of **Vandana Rawat, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

AYUSH RAWAT **22391138**

The above mentioned students shall be working under the supervision of the undersigned on the **"YouTube Video Downloader"**

Supervisor **Head of the Department**

Internal Evaluation (By DPRC Committee)

Status of the Synopsis: Accepted / Rejected

Any Comments:

Name of the Committee Members: **Signature with Date**

1.

2.

Table of Contents

YOUTUBE VIDEO DOWNLOADER

Chapter 1

INTRODUCTION

The YouTube Video Downloader is a mini project developed using a diverse range of modules, including yt_dlp, PySimpleGUI, Pillow, Requests, JSON, and more. This project aims to provide users with a convenient and user-friendly tool to effortlessly download YouTube videos and playlists. With additional functionalities like quality selection, audio-only downloads, subtitle downloads, thumbnail downloads, comment downloads, and the ability to choose the download folder, this project offers an all-in-one solution for YouTube video enthusiasts.

Moreover, the YouTube Video Downloader goes beyond just individual video downloads, as it also caters to the needs of playlist enthusiasts. Users can leverage this tool to download entire playlists with ease. The playlist functionalities include all the features available for individual videos, as well as the ability to download a range of videos within a playlist, auto-numbering videos in the playlist, and organizing them in a folder named after the playlist itself.

User Interface with PySimpleGUI:

The YouTube Video Downloader boasts a sleek and intuitive user interface, developed using the PySimpleGUI module. PySimpleGUI is a powerful library that allows developers to create visually appealing and user-friendly GUIs without the need for complex coding. With its simplicity and flexibility, PySimpleGUI ensures that users can easily navigate through the downloader's various options and settings, even without any prior coding experience. The interface's responsiveness further enhances the user experience, providing a seamless and efficient interaction platform.

Underlying Functionality with yt_dlp:

At the core of the YouTube Video Downloader lies the yt_dlp module, which serves as the backbone for fetching YouTube video metadata. This module provides a comprehensive set of functionalities for extracting essential

information, such as video URLs, titles, descriptions, and more. By leveraging yt_dlp, the project can offer an extensive range of features that enhance the downloading experience. These include quality selection, audio-only downloads, subtitle downloads, thumbnail downloads, and comment downloads. With yt_dlp's versatility, users have the flexibility to personalize their downloads based on their preferences.

Seamless Video Mixing with FFmpeg:

To provide users with high-quality video outputs, the YouTube Video Downloader integrates with FFmpeg, a robust multimedia framework. FFmpeg enables the project to seamlessly combine separately downloaded video and audio files, resulting in an output file with excellent audio and visual synchronization. By leveraging the power of FFmpeg, users can enjoy their downloaded videos with optimal audiovisual quality, ensuring an immersive viewing experience.

Download Progress Indicator:

The YouTube Video Downloader aims to keep users engaged and informed throughout the download process. To achieve this, it incorporates a download progress indicator. This feature provides real-time feedback, allowing users to track the progress of their downloads. Users can view details such as the current download speed, estimated time remaining, and overall completion percentage. By presenting this information in a clear and concise manner, the project ensures transparency and empowers users with valuable insights into their downloads.

Conclusion:

In conclusion, the YouTube Video Downloader is a comprehensive mini project that simplifies the process of downloading YouTube videos and playlists. Its range of functionalities, including video quality selection, audio-only downloads, subtitle downloads, thumbnail downloads, comment downloads, playlist support, auto-numbering, and folder organization, caters to the diverse needs of users. The intuitive user interface, built with PySimpleGUI, ensures a seamless experience, allowing users to navigate through the downloader effortlessly. Furthermore, the integration with FFmpeg enhances the audiovisual quality of the downloaded videos, resulting in an immersive viewing experience. The addition of a download progress indicator keeps users informed and engaged throughout the download process. Whether you are an avid YouTube video watcher or someone who wants to access content offline, the YouTube Video Downloader is a reliable and efficient solution that ensures an optimal downloading experience.

Chapter 2

OBJECTIVE

The objectives of this YouTube video downloader project are:

1. Provide an easy to use GUI for users to download YouTube videos. The GUI will allow users to select the video quality, download audio only, subtitles, comments, thumbnails and choose the download folder.
2. Download entire YouTube playlists with options to select a range of videos in the playlist, autonumber the downloaded videos and organize them into a folder named after the playlist.
3. Show the download progress to the user so they can monitor the status of ongoing downloads.
4. Utilize the yt_dlp module to do the actual downloading of YouTube videos and playlists.
5. Create a simple yet useful tool for users to easily download and organize their desired YouTube videos and playlists.

Chapter 3

HARDWARE REQUIREMENTS:

- Processor**:** Dual-core processor with a clock speed of >=2 GHz
- RAM: 2 GB
- Storage: 100 GB
- Input Devices: Keyboard and Mouse
- Internet

SOFTWARE REQUIREMENTS:

- Programming Language: Python >= 3.9
- OS: Windows 10/11
- Main Modules: yt_dlp, PySimpleGUI, PIL (pillow), json, requests, os, datetime, traceback
- Additional Software: ffmpeg

Chapter 4

Algorithm (Steps taken/Functions Used)

1. Function: `video_screen(json_data)`
   - This function is needed to display a GUI window for a single video.
   - It takes in a `json_data` dictionary containing information about the video.
   - The function creates a GUI window using the `PySimpleGUI` library.
   - The window displays information about the video, such as title, duration, and thumbnail.
   - It provides options for selecting video quality and additional download options like subtitles and comments.
   - The function returns a dictionary containing the window object and other relevant data.

2. Function: `playlist_screen(json_data)`
   - This function is needed to display a GUI window for a playlist.
   - It takes in a `json_data` dictionary containing information about the playlist.
   - The function creates a GUI window using the `PySimpleGUI` library.
   - The window displays information about the playlist, such as title, video count, and thumbnail.
   - It provides options for selecting video quality, setting playlist range, and additional download options like subtitles and comments.
   - The function returns a dictionary containing the window object and other relevant data.

3. Function: `folderr(folder_path, playlist_flag, put_in_folder_flag, numbering_flag)`
   - This function is needed to generate the output filename template for downloaded videos.
   - It takes in parameters such as folder path, playlist flag, put in folder flag, and numbering flag.
   - Based on these parameters, the function generates a template for the output filename.
   - The template includes the video title, playlist index (if applicable), and numbering (if applicable).
   - The function returns the output filename template.

4. Function: `download_video(data, values, json_data, url)`
   - This function is needed to handle the actual downloading of a single video.
   - It takes in data, values, json_data, and the URL of the video.
   - The function configures the download options based on the selected values and the provided `json_data`.
   - It uses the `yt_dlp` library to download the video.
   - Progress hooks are used to update the GUI window with download progress information.
   - The function displays a completion message once the download is finished.

5. Function: `download_playlist(data, values, json_data, url)`

- This function is needed to handle the downloading of a playlist.
- It takes in data, values, json_data, and the URL of the playlist.
- The function configures the download options and playlist range based on the selected values and the provided `json_data`.
- It uses the ` yt_dlp ` library to download the playlist.
- Progress hooks are used to update the GUI window with download progress information.
- The function displays a completion message once the download is finished.

6. Function: `popup_continue_or_not()`
- This function is responsible for displaying a popup window after the completion of a download.
- It creates a GUI window using the `sg.Window` class.
- The window layout consists of a message indicating that the download is complete.
- Two buttons are included in the popup: "Same Video/Playlist" and "New Video/Playlist".
- The buttons allow the user to choose whether to download the same video/playlist again or start downloading a new video/playlist.

7. Function: `main()`
- This function serves as the entry point of the program.
- It initializes the GUI window by calling the `initial_screen` function.
- The function enters a loop to handle events and user interactions with the window.
- It checks for events such as clicking the "proceed" button, "download" button for a single video, or "download_playlist" button for a playlist.
- Depending on the event, the function calls the corresponding functions to handle the download process.
- After the download process is complete, a popup appears and asks the user to choose whether to download the same video/playlist again or start downloading a new video/playlist.

8. The program execution starts by calling the `main()` function at the end of the code.