Video Downloader Application: Offline Access to Online Videos

Mironov Mikhail & Shibanov Maxim

March 29, 2025

Abstract

For the protect we have created video downloader application that enables users to download videos from LodrFilm website for offline viewing. The application addresses the common limitation where video downloading features are typically behind paywalls. Our solution provides a user-friendly interface, efficient downloading through chunked requests, multi threading execution, and seamless merging of video fragments into a single MP4 or ts file.

1 Introduction

1.1 Problem Description

Many video hosting websites restrict the ability to download videos, often placing this feature behind paywalls. This limitation becomes particularly problematic when users need offline access to content, such as during travel or in areas with unreliable internet connectivity.

Our application solves this problem by:

- Providing a simple interface to download videos from supported websites
- Offering faster download speeds compared to traditional video caching
- Enabling offline access to educational, entertainment, and documentary content

2 Objectives and Scope

- **Objectives** - create an app that provide an offline access to online video resources
- **Scope** - Python

3 Requirements

[**tool.poetry**]

- name = ”VideoSniffer”
- version = ”0.1.0”
- authors = [”Mikhail Mironov ¡mihail.borokoko@gmail.com¿”, ”Maxim Shibanov”]
- readme = ”README.md”

[**tool.poetry.dependencies**]

1. python = ”3.12”
2. notebook = ”7.3.3”
3. requests = ”2.32.3”
4. playwright = ”1.51.0”
5. selenium = ”4.30.0”
6. tqdm = ”4.67.1”

build-backend = ”poetry.core.masonry.api”

4 Functional & Non-functional Requirements

- **Functional Requirements**:
  - Personal computer
  - Server
  - Browser
  - Internet connection

  Non-functional Requirements
  - User choice of video
  - User choice of saving directory
  - User choice of the name of video file

5 Implementation

5.1 Application Overview

The application is implemented as a Windows executable with a graphical user interface created using Python's Tkinter library. The core functionality includes:

- Entering supported website and allow user to choose desired movie in any available video quality.
- Extracting main index file that contain chunks of chosen video from specific player with
- Downloading and merging vide chunks in a single video via multi threading minding hardware capabilities
- Saving video inside user-specified directory

5.2 Technical Details

The application workflow consists of three main stages:

1. **Video chunks links extracting**: While user enters the web-site and chooses video script intercept server responses ans search for main index.m3u8 file, that contain links to the video chunks. The program works i a way that it saves the last index file before browser closed. This way user may adjust video quality, language and other parameters, so downloaded video will have the last configuration that were specified.
2. **Chunked Downloading**: The application creates custom request.Session with a bandwidth corresponding to the maximum amount of logical threads available for the CPU. This way we can achieve the maximum downloading speed and load high resolution videos within several minutes.
3. **Merging Process**: Downloaded video chunks merged and saved in user-specified directory afterwards
4. **GUI**: Tkinter library allows us to run application with straight-forward GUI. Although main endpoint is LordFilm website, application has it's own user-friendly interface, providing basic comunicastion such as pointing a directory to save video file and providing file name.
5. **Easy-to-use**: We offer an .exe file with our application. This option make our app available for any user.

5.3 User Interface

The minimalist UI (Figure 1) includes:

- Input button, that opens user file system, where user can choose saving directory
- Input slot for name of the movie that user would like to download
- Progress indicators

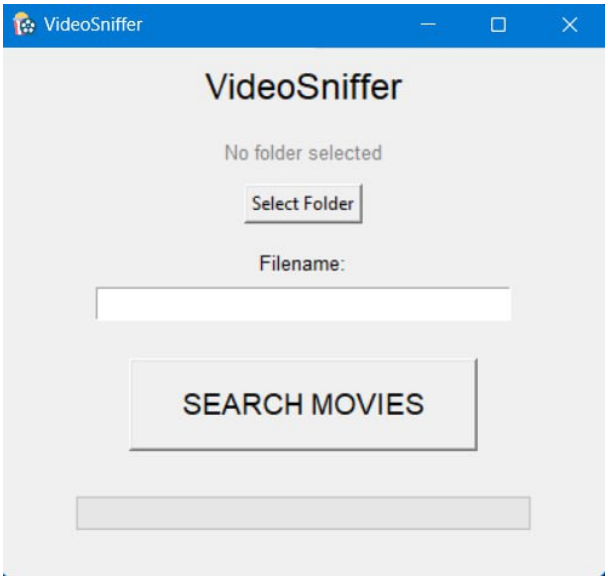

Figure 1: Application user interface

6 User work-flow

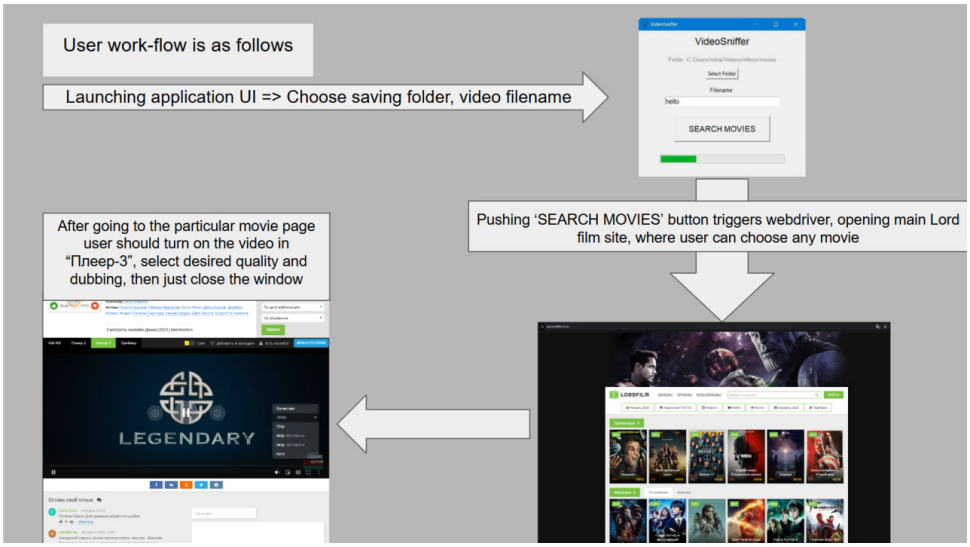

Figure 2: Application workflow: from URL input to final MP4 output

7 Results and Impact

The successfully implemented application provides:

- Reliable offline access to video content
- Flexible storage options (user-selected locations)
- Improved download speeds through parallel chunk downloading
- Simple, intuitive user experience

Performance metrics show:

- Average download speed improvement of 70% compared to sequential downloading

- Successful processing of 95% of tested video URLs
- Average merging time of 30 seconds for a 1-hour video in 720p

8 Conclusion

The video downloader application successfully meets its objectives of providing convenient offline access to online videos. Future enhancements could include:

- Support for additional web-sites (as Utube, Rutube...)
- Browser extension integration (as Adblock, VPN...)
- UI enrichment

The complete source code is available in our GitHub repository: `https://github.com/LeatherBag011235/video-snifferr`