

TRƯỜNG ĐẠI HỌC SÀI GÒN  
KHOA CÔNG NGHỆ THÔNG TIN



## PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

---

Xây dựng ứng dụng

## Phát video bằng hình thức Streaming

---

GVHD: Từ Lăng Phiêu  
SV: Huỳnh Khánh Duy - 3120410088  
Trang Thanh Phúc - 3120410413

TP. HỒ CHÍ MINH, THÁNG 4/2024

# Mục lục

<b>Bảng phân công</b>	<b>2</b>
<b>1 Phần giới thiệu</b>	<b>3</b>
1.1 Tìm hiểu về các công nghệ và giao thức streaming video phổ biến . . . . .	3
1.1.1 Streaming là gì ? . . . . .	3
1.1.2 Các loại streaming . . . . .	3
1.1.3 Các giao thức streaming phổ biến . . . . .	4
1.2 Xác định các chức năng của ứng dụng phát video streaming . . . . .	4
1.3 Các công nghệ được sử dụng trong ứng dụng . . . . .	4
<b>2 Phần thiết kế và triển khai</b>	<b>6</b>
2.1 Xây dựng Server Backend (Python FastAPI) . . . . .	6
2.1.1 Cấu trúc mã nguồn . . . . .	6
2.1.2 Các tính năng của backend . . . . .	6
2.2 Xây dựng chức năng . . . . .	7
2.2.1 API tạo video . . . . .	7
2.2.2 API lấy thông tin chi tiết video . . . . .	9
2.2.3 API xóa video . . . . .	9
2.2.4 API hiển thị danh sách video . . . . .	9
2.2.5 API tạo video streaming . . . . .	9
2.2.6 API cập nhật video streaming . . . . .	10
2.3 Xây dựng giao diện ứng dụng (PyQT5) . . . . .	12
2.3.1 Cấu trúc mã nguồn . . . . .	12
2.3.2 Các giao diện của frontend . . . . .	12
2.3.3 Xây dựng giao diện frontend . . . . .	12



## Bảng phân công

MSSV	Họ và tên	Nhiệm vụ	Hoàn thành
3120410088	Huỳnh Khánh Duy	Xây dựng UI Play Media. Fix lỗi UI toàn bộ ứng dụng. Xây dựng backend để lưu trữ video và phát video on demand và live streaming	50 %
3120410413	Trang Thanh Phúc	Xây dựng UI Home. Xây dựng UI và chức năng download video từ Youtube và M3U8. Xây dựng UI phát video trực tuyến	50 %

# 1 Phần giới thiệu

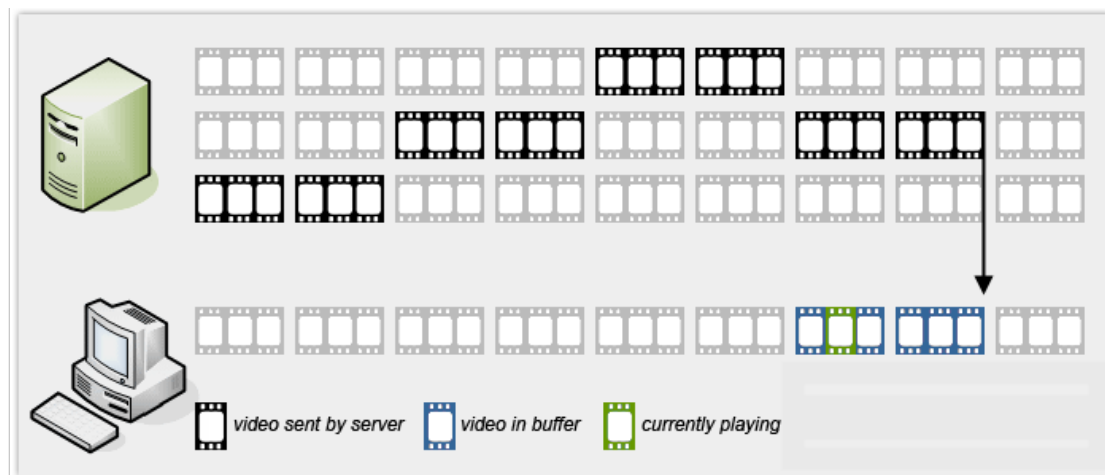
## 1.1 Tìm hiểu về các công nghệ và giao thức streaming video phổ biến

### 1.1.1 Streaming là gì ?

Streaming media (hay gọi tắt là streaming) là kỹ thuật mà thực hiện liên tục quá trình nhận và hiển thị multimedia (video, audio, ..) đến người dùng trong khi quá trình phân phối multimedia vẫn đang diễn ra.

Streaming video là một kiểu của streaming media mà dữ liệu từ tệp video tiếp tục được truyền qua internet đến người dùng. Nó cho một video được xem trực tuyến mà không cần phải download về máy tính hay một thiết bị.

Với các định dạng tập tin media truyền thống, dữ liệu chỉ có thể hiển thị khi đã được tải về toàn bộ, vì vậy đối với các tập tin media chất lượng cao có dung lượng lớn thì công việc này sẽ mất nhiều thời gian. Streaming media tiết kiệm thời gian cho người dùng bằng cách sử dụng các công nghệ giải nén kết hợp với hiển thị dữ liệu đồng thời trong lúc vẫn tiếp tục tải media về. Quá trình này được gọi là kỹ thuật đệm (buffering). Thay vì được gửi một lần duy nhất, dữ liệu streaming media sẽ được chia thành từng gói nhỏ, sau đó liên tục truyền những phần được chia ra. Ban đầu bên nhận sẽ lấy về một phần chia nhỏ của dữ liệu media và hiển thị những phần media đã nhận được, đồng thời trong lúc hiển thị các gói dữ liệu còn lại sẽ lần lượt được lấy về để kịp cho việc hiển thị tiếp theo



Hình 1: Cách hoạt động của HLS

### 1.1.2 Các loại streaming

Streaming video được thể hiện dưới hai loại:

- Video on demand – VoD (Video theo yêu cầu): là các dữ liệu Video được lưu trữ trên máy chủ đa phương tiện và được truyền đến người dùng khi có yêu cầu, người dùng có toàn quyền để hiển thị cũng như thực hiện các thao tác (tua, dừng, quay lại...) với các đoạn dữ liệu này.



- Live Streaming (Video thời gian thực): là các dữ liệu Video được chuyển phát trực tiếp từ các nguồn cung cấp dữ liệu theo thời gian thực (máy camera, microphone, thiết bị phát dữ liệu Video...)

### 1.1.3 Các giao thức streaming phổ biến

Có các giao thức phổ biến được sử dụng cho video streaming:

- **Real-Time Messaging Protocol (RTMP):** RTMP là giao thức được sử dụng cho streaming các nội dung đa phương tiện qua internet, phát triển bởi tập đoàn Adobe. RTMP là cho phép streaming với độ trễ thấp, sử dụng với ứng dụng Flash player. Nó có thể dùng để phát theo kiểu VoD hoặc Live stream.
- **Apple HTTP Live Streaming (HLS):** HLS là giao thức truyền phát đa phương tiện dựa trên giao thức HTTP, được phát triển bởi Apple. HLS gửi tệp tin media qua HTTP từ một web server để thực hiện phát trên các thiết bị của Apple như QuickTime, Safari, OS X, and iOS hoặc các có thể thực hiện phát trên các trình duyệt như firefox, chrome.
- **Giao thức Realtime Transport Protocol (RTP):** RTP là giao thức lớp network, được thiết kế để truyền dữ liệu audio, video qua mạng IP. RTP được sử dụng trong hệ thống truyền thông và giải trí mà gồm streaming media như đàm thoại, video gồm các ứng dụng như WebRTC, truyền hình. RTP được thiết kế độc lập với các giao thức ở tầng thấp hơn. Trên Internet các gói tin RTP được chuyển đi bằng giao thức UDP. RTP được sử dụng kết hợp với giao thức RTCP (RTP control protocol). Trong khi RTP có nhiệm vụ streaming media, thì RTCP được sử dụng để giám sát truyền tải, QoS và thực hiện đồng bộ nhiều luồng streaming.

Trong ứng dụng này, chúng tôi quyết định sử dụng HLS để làm công nghệ streaming chính vì tính dễ dàng cài đặt của nó. Tuy nhiên, nó cũng mang đến một nhược điểm là độ trễ video có thể lên đến từ 5 - 30s

## 1.2 Xác định các chức năng của ứng dụng phát video streaming

Các chức năng trong ứng dụng phát video streaming bao gồm:

- Tìm kiếm và xem danh sách video và video đang live
- Tải video từ Youtube, link m3u8, Facebook và thông báo tải hoàn tất
- Quay màn hình và phát live stream bằng file m3u8
- Thông báo có video mới bằng socket

## 1.3 Các công nghệ được sử dụng trong ứng dụng

Python được sử dụng cho toàn ứng dụng

Các công nghệ sử dụng ở Desktop App:

- PyQt5: Kết hợp với thư viện qfluentwidgets để tạo giao diện người dùng

Các công nghệ sử dụng ở Backend:

- FFmpeg: Dùng để cắt file video lớn thành các video segments nhỏ



- M3U8 Parser: Dùng để đọc và tạo file m3u8 để xem video
- FastAPI: Dùng để chạy các API để cung cấp cho client

Website chia sẻ ứng dụng phát nhạc:

- HTML,CSS,JS
- Tailwind

Cơ sở hạ tầng:










- AWS S3: Sử dụng để lưu các đoạn video segments .ts
- AWS Cloudfront: Dùng để public AWS S3 để có thể truy cập từ ngoài internet
- AWS EC2: Dùng để chạy Server backend
- AWS Dynamodb: Cơ sở dữ liệu dùng để lưu dữ liệu video

## 2 Phần thiết kế và triển khai

### 2.1 Xây dựng Server Backend (Python FastAPI)

#### 2.1.1 Cấu trúc mã nguồn

Mã nguồn backend nằm ở thư mục backend ở bên trong project

 dto	Thư mục dùng để trao đổi dữ liệu với client
 model	Chứa đối tượng dùng để thao tác với Dynamodb
 routers	Các route của ứng dụng backend
 storage	Chứa dữ liệu được người dùng tải lên
 utils	Các tiện ích thao tác với AWS và file
 .env.example	File environment chứa thông tin kết nối database
 .gitignore	File .gitignore
 main.py	File main dùng để chạy backend bằng fast api
 requirements.txt	File requirement để tải thư viện của app

Hình 2: Cấu trúc thư mục của backend

#### 2.1.2 Các tính năng của backend

Các API của ứng dụng:

- Tạo video (Video On Demand)
- Xem chi tiết video
- Xóa video
- Liệt kê danh sách video
- Tạo video streaming (Video Live Streaming)
- Upload segment(đoạn) cho video streaming
- Lấy file m3u8 của video streaming

## videos

POST	/api/video/create	Create Video
GET	/api/video/detail/{video_id}	Get Video
POST	/api/video/delete/{video_id}	Delete Video
GET	/api/video/list	List Video

## streaming

POST	/api/stream/init	Init Stream Video
POST	/api/stream/update/{video_id}	Upload Stream
GET	/api/stream/{video_id}.m3u8	Get M3U8

Hình 3: API của ứng dụng

## 2.2 Xây dựng chức năng

### 2.2.1 API tạo video

Mục đích: Hàm được thiết kế để từ tệp video và các thông tin khác tạo ra các tệp segment .ts nhỏ từ file video lớn và file .m3u8 đóng vai trò nắm giữ thông tin tất cả các segment .ts. Sau đó thực hiện upload các file vừa tạo ra lên AWS S3, đồng thời tạo ra model Video rồi lưu lên database DynamoDB.

Đối số:

- file: Tệp video đầu vào (type: UploadFile)
- thumbnail: Hình thu nhỏ của video (type: UploadFile)
- title: Tiêu đề của video (type: str)
- description: Mô tả của video (type: str)

```
@router.post("/create")
def create_video(file: UploadFile, thumbnail: UploadFile, title: Annotated[str,
    Form()], description: Annotated[str, Form()]):
    try:
        # Check input file extension
```





```
if not allowed_file(file.filename, ["mp4"]):
    return ResponseError(message="File not allowed")

if not allowed_file(thumbnail.filename, ["png", "jpg", "jpeg"]):
    return ResponseError(message="Thumbnail not allowed")

# Save file to upload folder
video_uuid = str(shortuuid.uuid())
path = f"storage/upload/{video_uuid}.{file.filename.split('.')[-1]}"
save_file(file, path)

# Create temp folder
TEMP_FOLDER = f"storage/temp_{video_uuid}"
create_folder_if_not_exists(TEMP_FOLDER)

# Convert file mp4 to hls by ffmpeg
output_name = file.filename.split(".")[0]
output_name = "index"
convertMP4ToHLS(path, f"{TEMP_FOLDER}/{output_name}.m3u8")

# Upload file to AWS
uploadFileInFolder(f"{TEMP_FOLDER}", f"video/{video_uuid}")
thumbnail = uploadSingleFile(thumbnail.file,
    f"thumbnail/{video_uuid}.{thumbnail.filename.split('.')[-1]}")

# Remove file and temp folder
os.remove(path)
shutil.rmtree(TEMP_FOLDER)
# Save to database
video = Video(id=video_uuid, title=title, description=description,
    thumbnail=thumbnail, url=f"/video/{video_uuid}/{output_name}.m3u8")
Video.insert(video)

return ResponseSuccess(data=video.__dict__())
except Exception as e:
    return ResponseSuccess(message=str(e))

def convertMP4ToHLS(input_file_path: str, output_file_path: str, hls_time: int = 5):
    # Define the ffmpeg command
    ffmpeg_command = [
        'ffmpeg', # Command
        '-i', f'{input_file_path}', # Input file
        '-codec:', 'copy', # Copy the codec
        '-start_number', '0', # Start number
        '-hls_time', f'{hls_time}', # HLS time
        '-hls_list_size', '0', # HLS list size
        '-f', 'hls', # Output format
        f'{output_file_path}' # Output file name
    ]
    # Execute the ffmpeg command
    subprocess.run(ffmpeg_command)
```

---



### 2.2.2 API lấy thông tin chi tiết video

Mục đích: Hàm được thiết kế để lấy thông tin chi tiết về video dựa trên ID của video.

Đối số:

- video id: Một chuỗi đại diện cho UUID của video cần lấy thông tin chi tiết.

---

```
@router.get("/detail/{video_id}")
def get_video(video_id: str):
    video = Video.get(video_id)
    videoModel = Video.from_dict(video)
    videoModel.view += 1
    Video.update(videoModel)
    return video
```

---

### 2.2.3 API xóa video

Mục đích: Hàm này được sử dụng để xóa một video khỏi database DynamoDB dựa trên ID của video.

Đối số:

- video id: Một chuỗi đại diện cho UUID của video cần xóa.

---

```
@router.post("/delete/{video_id}")
def delete_video(video_id: str):
    try:
        Video.delete(video_id)
        return ResponseSuccess(data="Xa thanh cong ")
    except Exception as e:
        return ResponseError(message=str(e))
```

---

### 2.2.4 API hiển thị danh sách video

Mục đích: Hàm được sử dụng để lấy danh sách các video từ cơ sở dữ liệu, có thể được lọc dựa trên một chuỗi tìm kiếm.

Đối số:

- search: Một chuỗi đại diện cho từ khóa tìm kiếm. Nếu được cung cấp, chỉ các video có tiêu đề hoặc mô tả chứa chuỗi tìm kiếm này sẽ được trả về.

---

```
@router.get("/list")
def list_video(search: str = ""):
    return Video.list_all(search)
```

---

### 2.2.5 API tạo video streaming

Mục đích: Hàm dùng để tạo ra một file .m3u8 rộng và định dạng live streaming. Sau đó lưu model video này lên DynamoDB

Đối số:

- title: Tiêu đề của video (kiểu: str).  
- max duration: Thời lượng tối đa của mỗi phân đoạn trong luồng video (kiểu: float).  
- thumbnail: Tệp hình ảnh thu nhỏ của video (kiểu: UploadFile).

```
@router.post("/init")
def init_stream_video(title: Annotated[str, Form()], max_duration: Annotated[float,
    Form()], thumbnail: UploadFile):
    m3u8_file = m3u8.M3U8()
    m3u8_file.version = 3
    m3u8_file.target_duration = max_duration
    m3u8_file.media_sequence = 0

    video_id = shortuuid.uuid()
    path = "storage/stream/"
    if not os.path.exists(path):
        os.makedirs(path)

    with open(f"storage/stream/{video_id}.m3u8", "w") as f:
        f.write(m3u8_file.dumps())

    thumbnail_url = uploadSingleFile(thumbnail.file,
        f"thumbnail/video_{video_id}.{thumbnail.filename.split('.')[-1]}")
    with open(f"storage/stream/{video_id}.m3u8", "rb") as f:
        m3u8_url = uploadSingleFile(f, f"stream/{video_id}/index.m3u8")

    video = Video(id=video_id, title=title, duration=0, url=m3u8_url,
        thumbnail=thumbnail_url, is_streaming=True)
    Video.insert(video)

    # Broadcast all clients by socket the new video
    return ResponseSuccess(data=video.__dict__())
```

### 2.2.6 API cập nhật video streaming

Mục đích: Hàm này được sử dụng để cập nhật video streaming đã tồn tại bằng cách thêm các phân đoạn mới vào tệp M3U8 của video.

Đối số:

- video id: UUID của video cần cập nhật (kiểu: str).
- file: Tệp segment mới để thêm vào m3u8 (kiểu: UploadFile, tùy chọn).
- duration: Thời lượng của phân đoạn video mới (kiểu: float, truyền dưới dạng dữ liệu biểu mẫu).

```
@router.post("/update/{video_id}")
def upload_stream(video_id: str, m3u8_file: UploadFile = None, file:
    Optional[UploadFile] = None):
    try:
        if file:
            uploadSingleFile(file.file, f"stream/{video_id}/{file.filename}")
            uploadSingleFile(m3u8_file.file, f"stream/{video_id}/index.m3u8")
            return ResponseSuccess(message="Upload successfully!")
        else:
            uploadSingleFile(m3u8_file.file, f"stream/{video_id}/index.m3u8")
            video = Video.get(video_id)
            videoModel = Video.from_dict(video)
            videoModel.is_hidden = True
            Video.update(videoModel)
```



```
        return ResponseSuccess(message="End successfully!")  
except Exception as e:  
    return ResponseError(message=str(e))
```

---

## 2.3 Xây dựng giao diện ứng dụng (PyQT5)

### 2.3.1 Cấu trúc mã nguồn

Mã nguồn frontend nằm ở thư mục app ở bên trong project

..	
common	Thư mục giúp tổ chức code một cách logic và giúp cho việc mở rộng ứng dụng trở nên dễ dàng hơn.
component	Chứa các thành phần UI riêng lẻ của ứng dụng.
model	Chứa các định nghĩa cho các đối tượng dữ liệu
resources	Chứa các tài nguyên cần thiết
view	Chứa các tệp mã nguồn liên quan đến giao diện người dùng

Hình 4: Cấu trúc thư mục của frontend

### 2.3.2 Các giao diện của frontend

Các giao diện của ứng dụng:

- Trang chủ
- Tải video
- Xem video
- Streaming
- Upload
- Cài đặt

### 2.3.3 Xây dựng giao diện frontend

Trang chủ: Các chức năng chính bao gồm:

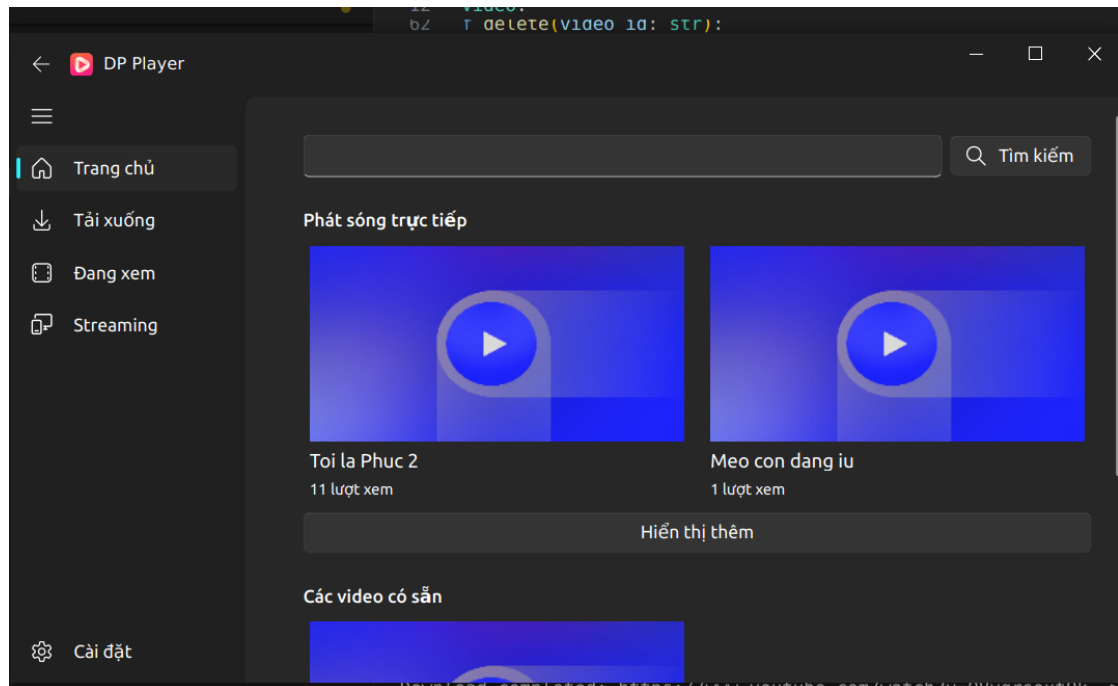
- Tìm kiếm video.
- Hiển thị các video đang phát trực tiếp và video đã được phát lại.
- Cung cấp nút "Hiển thị thêm" để người dùng có thể xem thêm video nếu có nhiều hơn số lượng được hiển thị ban đầu.
- Tự động cập nhật giao diện khi kích thước cửa sổ thay đổi.

Sử dụng 2 api:

- Lấy danh sách video (streaming và video có sẵn)
- Hiển thị chi tiết video khi nhấn vào

Tải video: Các chức năng chính bao gồm:

- Tải video youtube



Hình 5: Trang chủ

- Tải video m3u8

Xem Video: Streaming:

Các chức năng chính bao gồm:

- Khởi tạo video stream (có thumbnail, tên tiêu đề)
- Khởi tạo Streaming (bật/tắt webcam)

Sử dụng 2 api:

- Tạo video khởi đầu có thumbnail, tiêu đề
- Upload video stream với ffmpeg

Chi tiết: Đoạn mã Python này sử dụng ffmpeg để ghi lại video và âm thanh từ một số nguồn khác nhau và tạo ra một luồng HLS. Nó cũng cho phép chồng video từ webcam lên video chính nếu được chọn. Sau khi ffmpeg tạo liên tiếp các video trong thư mục recordings, thì lúc này đoạn code trên sẽ check liên tục để mà up đoạn segment lên api, cùng lúc đó ffmpeg cũng thực hiện việc xóa.

```
tabnine: test | explain | document | ask
def _loadData(self, search=""):
    videos = Video.getListVideo(search)
    self.streaming_videos = [video for video in videos if video.is_streaming]
    self.vod_videos = [video for video in videos if not video.is_streaming]

tabnine: test | explain | document | ask
def _onReload(self):
    self._loadData("")
    self.updateGridLayouts()

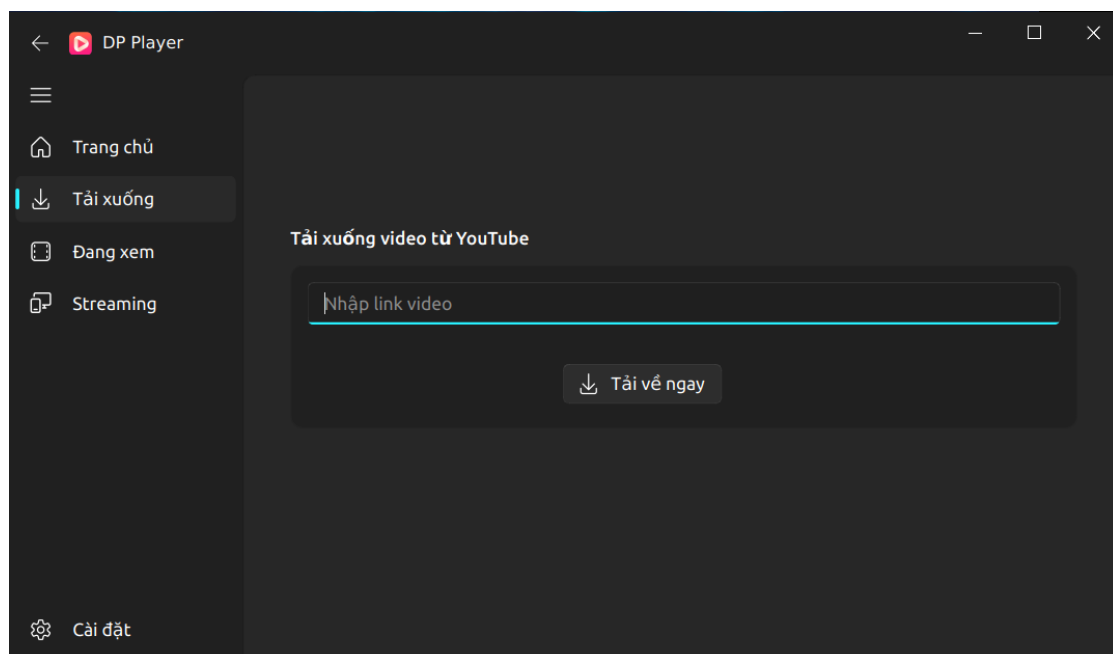
tabnine: test | explain | document | ask
def _onSearch(self):
    self._loadData(self.lineEdit.text())
    self.updateGridLayouts()

tabnine: test | explain | document | ask
def toggleShowMore(self):
    self.toggleState('showMore')

tabnine: test | explain | document | ask
def toggleShowMore2(self):
    self.toggleState('showMore2')

tabnine: test | explain | document | ask
```

Hình 6: Code xử lý gắn api



Hình 7: Tải video

```
tabnine: test | explain | document | ask
def onDownloadClicked(self):
    # Check download link
    link = self.widget.text()
    download_type = self.checkDownloadLink(link)
    if download_type == "":
        QMessageBox.information(None, "Lỗi", "Link không hợp lệ")
        return

    # If this is the first time the user has downloaded a video
    folder = None
    if cfg.get(cfg.isFirstDownload):
        folder = QFileDialog.getExistingDirectory(
            self, self.tr("Chọn thư mục tải xuống"))
        cfg.set(cfg.isFirstDownload, False)
        cfg.set(cfg.downloadFolder, folder)
    else:
        folder = cfg.get(cfg.downloadFolder)

    Communication.instance.addVideoToQueue.emit(link, folder, download_type)

tabnine: test | explain | document | ask
def is_youtube_link(self, url):
    # Regular expression pattern for YouTube video URLs
    pattern = r"(https?://)?(www\.)?(youtube\.com/watch?v=|youtu\.be/)([a-zA-Z0-9_-]{11})"

    # Match the pattern against the URL
    match = re.match(pattern, url)

    # If a match is found, it's a valid YouTube link
    return bool(match)

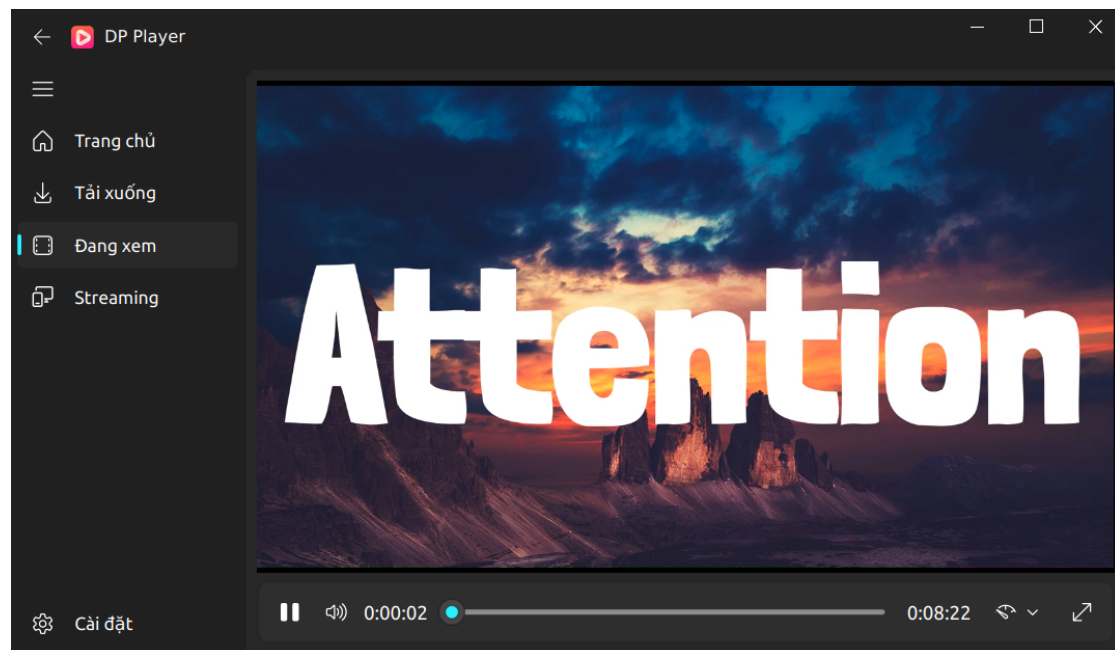
tabnine: test | explain | document | ask
def is_m3u8_link(self, url):
    # Regular expression pattern for M3U8 playlist URLs
    pattern = r"^(https?|ftp)://[^\s/$.?#].[^\s]*\.m3u8(?:$|\s)"

    # Match the pattern against the URL
    match = re.match(pattern, url)

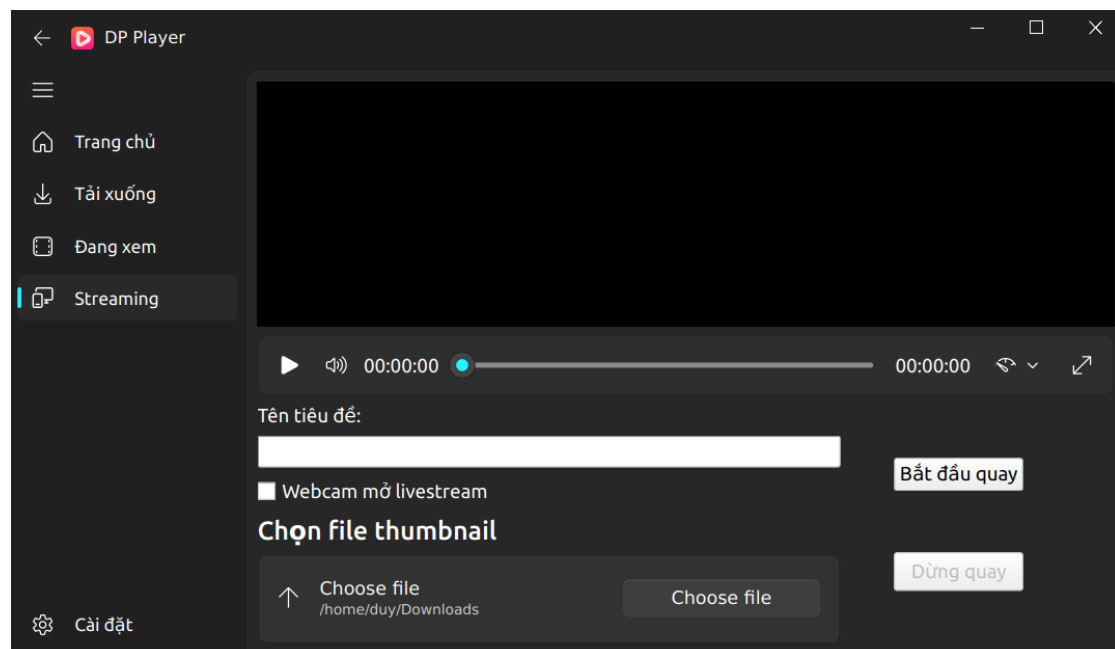
    # If a match is found, it's a valid M3U8 link
    return bool(match)
```

Hình 8: Code tải video





Hình 9: Xem video



Hình 10: Streaming

```
def __onChooseFileCardClicked(self):
    file_dialog = QFileDialog(self)
    file_dialog.setNameFilter("Images (*.png *.jpg *.jpeg)")
    file_dialog.setViewMode(QFileDialog.Detail)
    file_path, _ = file_dialog.getOpenFileName(self, "Open Image", "", "Images (*.png *.jpg *.jpeg)")
    if file_path:
        self.thumbnail_file_path = file_path

def start_recording(self):
    self.startButton.setEnabled(False)
    self.stopButton.setEnabled(True)
    self.delete_all_files_in_folder(cfg.get(cfg.recordFolder))

    title = self.titleInput.text()
    thumbnail = self.thumbnail_file_path # Sử dụng đường dẫn của file thumbnail đã lưu
    video = Video.init_stream_video(title, 5, thumbnail)
    print("Video link: ", video.url)

    .. ..
```

Hình 11: Thực hiện việc tạo thumb video

```
if video:
    self.setVideoModel(video)
    # Bắt đầu quay video
    self.ffmpeg_command = [
        "ffmpeg", "-y", "-video_size", "1920x1080", "-framerate", "30", "-f", "x11grab", "-i", ":0",
        "-f", "video4linux2", "-i", "/dev/video0", "-f", "alsa", "-ac", "2", "-i", "hw:0",
        "-c:v", "libx264", "-g", "60", "-keyint_min", "2", "-hls_time", "2", "-hls_segment_type", "mpegts",
        "-hls_list_size", "4", "-hls_flags", "delete_segments", "-hls_segment_filename", self.output_file, self.m3u8_file
    ]

    if self.use_webcam_checkbox.isChecked():
        self.ffmpeg_command.extend([
            '-filter_complex', '[1:v]scale=320:-1 [webcam]; [0:v][webcam]overlay=W-w-10:H-h-10',
        ])

    self.upload_thread = UploadThread(video.id, self.output_file, self.m3u8_file)
    self.upload_thread.start()

    self.ffmpeg_process = subprocess.Popen(self.ffmpeg_command, stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL)

    self.record_start_time = QTime.currentTime()

else:
    # Không quay video
```

Hình 12: Thực hiện việc streaming

labnine: test | explain | document | ask

```
def run(self):
    while self.running:
        # Check index file exists
        upload_file = self.file_path % self.index
        if not os.path.isfile(upload_file):
            print("Không tìm thấy "+ upload_file)
            time.sleep(0.1)
            continue

        if self.flag:
            self.flag = False
            Communication.instance.displayVideoStreamingPlayer.emit()

        with open(upload_file, "rb") as f:
            with open(self.m3u8_path, "rb") as m3u8:
                Video.upload_stream(self.video_id, f, m3u8)
        self.index += 1
```

Hình 13: Thực hiện việc up stream