

TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN



PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Xây Dựng Ứng Dụng

Phát Triển Phần Mềm Play Video Theo Hình Thức Streaming

Thành viên nhóm:

Diệp Bảo Thanh Phong - 3119410304
Trần Đức Thanh - 3121410451
Nguyễn Trần Đắc Tài - 3118410382

Giáo viên hướng dẫn:

ThS. Từ Lãng Phiêu

TP. HỒ CHÍ MINH, THÁNG 5/2024

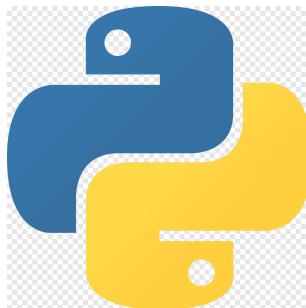
Mục lục

1	Giới Thiệu Chung	2
1.1	Python	2
1.2	FastAPI	2
1.2.1	Giới thiệu	2
1.2.2	Ưu điểm của FastAPI	2
1.2.3	Ví dụ mã nguồn	3
1.3	Jinja2	3
1.3.1	Giới thiệu	3
1.3.2	Ưu điểm của Jinja2	3
1.3.3	Ví dụ mã nguồn	3
2	Phân Tích Đề Tài	4
2.1	Lý do lựa chọn đề tài	4
2.2	Mục Tiêu	4
2.3	Phạm vi nghiên cứu	5
2.4	Ứng dụng	5
3	Nguyên lý hoạt động	6
3.1	Kiến trúc hệ thống	6
3.2	Quy trình hoạt động	6
3.2.1	Tải lên video	6
3.2.2	Quản lý video	6
3.2.3	Xem video trực tuyến	7
4	Phần lập trình	8
5	Ưu và nhược điểm của đồ án	13
5.1	Ưu điểm	13
5.2	Nhược điểm	13

1 Giới Thiệu Chung

1.1 Python

Python là một ngôn ngữ lập trình mạnh mẽ, dễ học và đa mục đích. Nó được tạo ra bởi Guido van Rossum và được phát hành lần đầu vào năm 1991. Python có cú pháp đơn giản và rõ ràng, giúp người dùng dễ dàng đọc, viết và hiểu code.



Hình 1: Biểu tượng của ngôn ngữ lập trình Python.

Với Python có thể phát triển ứng dụng web, ứng dụng di động, ứng dụng, trò chơi, công cụ phân tích dữ liệu, trí tuệ nhân tạo và nhiều hơn nữa. Nó có một cộng đồng lớn và phong phú, cung cấp nhiều thư viện và framework hữu ích để hỗ trợ phát triển.

Python cũng là một ngôn ngữ linh hoạt, có khả năng tích hợp và tương tác với các ngôn ngữ khác như C/C++ và Java. Nó có một cú pháp đơn giản và hỗ trợ nhiều phong cách lập trình, bao gồm lập trình hướng đối tượng, lập trình hàm và lập trình thủ tục. Tuy nhiên, Python thường chậm hơn so với các ngôn ngữ biên dịch và có thể gây ra các vấn đề về quản lý bộ nhớ tự động và không tương thích giữa các phiên bản khác nhau.

1.2 FastAPI

1.2.1 Giới thiệu

FastAPI là một framework hiện đại, nhanh và dễ sử dụng để xây dựng các API với Python 3.7+. Nó được thiết kế để tạo ra các API nhanh và có hiệu suất cao, đồng thời cung cấp một trải nghiệm lập trình viên tuyệt vời với các tính năng tự động sinh tài liệu API và hỗ trợ kiểm tra kiểu mạnh mẽ.

1.2.2 Ưu điểm của FastAPI

- Hiệu suất cao:** FastAPI dựa trên Starlette và Pydantic, cho phép xử lý các yêu cầu một cách nhanh chóng và hiệu quả.
- Dễ sử dụng:** Cung cấp cú pháp đơn giản và trực quan, giúp lập trình viên dễ dàng xây dựng các API.
- Tự động sinh tài liệu:** FastAPI tự động sinh tài liệu API từ mã nguồn, giúp dễ dàng kiểm thử và duy trì.
- Kiểm tra kiểu mạnh mẽ:** Sử dụng các kiểu dữ liệu của Python, FastAPI giúp phát hiện lỗi sớm và cải thiện chất lượng mã nguồn.



1.2.3 Ví dụ mã nguồn

Dưới đây là một ví dụ về cách sử dụng FastAPI để tạo một API đơn giản:

Listing 1: Ví dụ FastAPI

```
1 from fastapi import FastAPI
2
3 app = FastAPI()
4
5 @app.get("/")
6 async def read_root():
7     return {"Hello": "World"}
8
9 @app.get("/items/{item_id}")
10 async def read_item(item_id: int, q: str = None):
11     return {"item_id": item_id, "q": q}
```

1.3 Jinja2

1.3.1 Giới thiệu

Jinja2 là một engine template mạnh mẽ và linh hoạt được sử dụng rộng rãi trong các ứng dụng web để tạo ra các trang HTML động. Nó được thiết kế để tách biệt logic của ứng dụng với giao diện người dùng, giúp mã nguồn dễ bảo trì và mở rộng.

1.3.2 Ưu điểm của Jinja2

- **Dễ sử dụng:** Cung cấp cú pháp đơn giản và trực quan, dễ học và sử dụng.
- **Linh hoạt:** Hỗ trợ nhiều tính năng mạnh mẽ như vòng lặp, điều kiện, macro và kế thừa template.
- **Hiệu suất cao:** Được tối ưu hóa để xử lý nhanh chóng và hiệu quả các template.

1.3.3 Ví dụ mã nguồn

Dưới đây là một ví dụ về cách sử dụng Jinja2 để tạo một trang HTML động:

Listing 2: Ví dụ Jinja2

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>{{ title }}</title>
5 </head>
6 <body>
7     <h1>{{ header }}</h1>
8     <ul>
9         {% for item in items %}
10            <li>{{ item }}</li>
11        {% endfor %}
12    </ul>
```



13 </body>
14 </html>

2 Phân Tích Đề Tài

2.1 Lý do lựa chọn đề tài.

1. Nhu Cầu Thực Tế và Phổ Biến

Với sự phát triển mạnh mẽ của internet và các thiết bị di động, việc xem video trực tuyến đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày. Các nền tảng như YouTube, Netflix và các dịch vụ phát trực tuyến khác đã chứng minh rằng nhu cầu phát video trực tuyến là rất lớn.

2. Tính Ứng Dụng Cao

Một hệ thống phát video trực tuyến có thể được áp dụng trong nhiều lĩnh vực như giáo dục (các khóa học trực tuyến), giải trí (phim, âm nhạc), truyền thông (tin tức, sự kiện trực tiếp), và doanh nghiệp (hội nghị trực tuyến, đào tạo nhân viên).

3. Thách Thức Kỹ Thuật Thú Vị

Phát triển một hệ thống phát video trực tuyến yêu cầu kiến thức sâu rộng về nhiều công nghệ khác nhau như mạng máy tính, xử lý video, giao thức truyền dữ liệu, tối ưu hóa hiệu suất và bảo mật. Điều này mang lại cơ hội học hỏi và phát triển kỹ năng cho các thành viên trong nhóm.

4. Cơ Hội Cải Tiết và Sáng Tạo

Mặc dù đã có nhiều nền tảng phát video trực tuyến, nhưng vẫn còn nhiều không gian cho sự cải tiến và sáng tạo. Nhóm có thể tìm cách cải thiện trải nghiệm người dùng, tối ưu hóa hiệu suất truyền tải, hoặc thêm các tính năng mới như tương tác trực tiếp, đề xuất nội dung thông minh, và hỗ trợ đa nền tảng.

2.2 Mục Tiêu

1. Xây Dựng Hệ Thống Phát Video Trực Tuyến

Phát triển một ứng dụng web cho phép người dùng tải lên, quản lý và xem video trực tuyến với chất lượng cao và độ trễ thấp.

2. Tối Ưu Hóa Hiệu Suất và Độ Tin Cậy

Sử dụng các kỹ thuật như chia sẻ nội dung theo phân đoạn, nén video, và sử dụng mạng phân phối nội dung (CDN) để đảm bảo video được truyền tải mượt mà và chất lượng cao ngay cả khi có nhiều người dùng truy cập đồng thời.

3. Đảm Bảo Tính Tương Thích và Trải Nghiệm Người Dùng Tốt

Thiết kế giao diện người dùng thân thiện, dễ sử dụng và tương thích với nhiều thiết bị và trình duyệt khác nhau. Hỗ trợ các tính năng như tạm dừng, tua nhanh, tua lại, và chuyển đổi chất lượng video tùy theo băng thông của người dùng.



4. Bảo Mật và Quản Lý Bản Quyền

Tích hợp các biện pháp bảo mật như mã hóa dữ liệu, xác thực người dùng, và quản lý bản quyền nội dung để bảo vệ quyền lợi của người dùng và nhà cung cấp nội dung.

5. Phát Triển Hệ Thống Quản Lý Nội Dung (CMS)

Cung cấp một hệ thống quản lý nội dung cho phép người dùng quản lý các video đã tải lên, theo dõi lượt xem, và thu thập phản hồi từ người xem. Hệ thống này cũng sẽ hỗ trợ quản lý danh mục video, thêm thông tin mô tả, thẻ và danh sách phát.

2.3 Phạm vi nghiên cứu

- Các công nghệ web hiện đại:** Nghiên cứu và sử dụng các công nghệ web như FastAPI, Jinja2, HTML5, CSS3 và JavaScript để xây dựng ứng dụng.
- Phát triển API RESTful:** Xây dựng các API để hỗ trợ việc tải lên, quản lý và phát video trực tuyến.
- Kỹ thuật streaming video:** Sử dụng các phương pháp và kỹ thuật streaming video để đảm bảo việc phát video mượt mà và hiệu quả.
- Lưu trữ và quản lý file:** Nghiên cứu các phương pháp lưu trữ và quản lý file video và hình ảnh một cách hiệu quả.
- Bảo mật:** Đảm bảo an toàn cho việc tải lên và truy cập video, bao gồm quản lý quyền truy cập và bảo vệ dữ liệu.

2.4 Ứng dụng

- Học tập và giảng dạy trực tuyến:** Ứng dụng này có thể được sử dụng để phát triển các nền tảng học tập và giảng dạy trực tuyến, nơi giáo viên có thể tải lên các video bài giảng và học sinh có thể xem trực tuyến.
- Chia sẻ video cá nhân:** Người dùng có thể tải lên và chia sẻ các video cá nhân với bạn bè hoặc cộng đồng.
- Truyền thông và giải trí:** Các công ty truyền thông có thể sử dụng ứng dụng này để phát trực tiếp các sự kiện, chương trình truyền hình, hoặc phim ảnh.
- Doanh nghiệp và tổ chức:** Các doanh nghiệp và tổ chức có thể sử dụng ứng dụng này để phát triển các nền tảng đào tạo nhân viên trực tuyến hoặc chia sẻ thông tin nội bộ qua video.
- Mạng xã hội video:** Tạo nền tảng mạng xã hội nơi người dùng có thể tải lên, xem và bình luận về các video của nhau.
- Giáo dục trực tuyến:** Hỗ trợ các khóa học trực tuyến, hội thảo và các chương trình giáo dục khác với tính năng tải lên và phát video bài giảng.
- Truyền thông và quảng cáo:** Các công ty truyền thông có thể sử dụng nền tảng này để phát các video quảng cáo, chương trình giải trí hoặc tin tức.
- Chia sẻ video gia đình:** Lưu trữ và chia sẻ các video gia đình với người thân và bạn bè, đảm bảo tính riêng tư và bảo mật.

3 Nguyên lý hoạt động

3.1 Kiến trúc hệ thống

Ứng dụng web được thiết kế với kiến trúc phân lớp bao gồm các thành phần chính:

- **Frontend:** Đoạn mã sử dụng Jinja2Templates để tạo giao diện người dùng (UI). Các trang HTML được tạo ra từ các template Jinja2.
- **Backend:** Đoạn mã sử dụng FastAPI để xử lý các yêu cầu từ người dùng, bao gồm việc tải lên video, quản lý video và phát video trực tuyến.
- **Static File Storage:** Các tệp video và hình ảnh được lưu trữ trong các thư mục tĩnh như "uploaded-videos" và "uploaded-images".

3.2 Quy trình hoạt động

3.2.1 Tải lên video

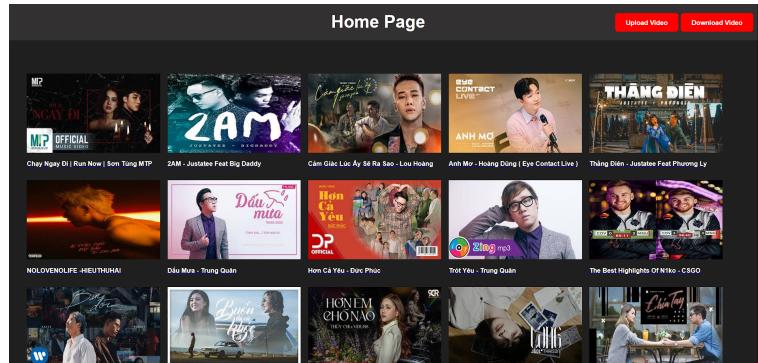
1. Người dùng gửi yêu cầu tải lên video và hình ảnh thông qua một form HTML..

Hình 2

2. Backend nhận các tệp này, lưu trữ chúng trong hệ thống file tĩnh và cập nhật thông tin vào tệp JSON để quản lý dữ liệu video.

3.2.2 Quản lý video

Người dùng có thể truy cập trang chính để xem danh sách các video đã được tải lên. Thông tin về các video được lấy từ tệp JSON và hiển thị trên giao diện người dùng



3.2.3 Xem video trực tuyến

Người dùng có thể chọn và xem video trực tuyến. Backend sử dụng kỹ thuật streaming để phát video từng phần, đảm bảo việc phát video mượt mà và hiệu quả

The screenshot shows the Network tab of a browser's developer tools. It lists several requests for media files, all with a status of 206 (Partial Content) and a type of 'media'. The requests are from the same URL and have a size of 250 kB or 206 B. The initiator is 'Other'.

Name	Status	Type	Initiator	Size	Time
20240514233517	206	media	Other	250 kB	
20240514233517	206	media	Other	250 kB	
20240514233517	206	media	Other	250 kB	
20240514233517	206	media	Other	250 kB	
20240514233517	206	media	Other	250 kB	
20240514233517	206	media	Other	250 kB	
20240514233517	206	media	Other	250 kB	
20240514233517	206	media	Other	206 B	

The screenshot shows a video player interface. At the top, there is a navigation bar with a 'Home Page' button. Below it is a large video thumbnail for the song '2AM - Justatee Feat Big Daddy'. The video thumbnail shows a person in a blue shirt with the text 'FREE HOOVER'. Below the thumbnail, there is a progress bar showing '0:16 / 4:10'. To the right of the video player, there is a sidebar with other video thumbnails and titles, including 'Chạy Ngày Đó | Run Now | Sơn Tùng MTP', '2AM - Justatee Feat Big Daddy', 'Cảm Giác Lúc Ấy Sẽ Ra Sao - Lou Hoàng', and 'ANH MỌI - HOÀNG ĐỨNG (Eye Contact Live)'.



4 Phần lập trình

Nhập thư viện và khởi tạo ứng dụng

```
1  from pathlib import Path
2from fastapi import FastAPI, HTTPException
3from fastapi import Request, Response, Form, Depends, UploadFile, File
4from fastapi.staticfiles import StaticFiles
5from fastapi.responses import HTMLResponse, JSONResponse
6from fastapi import Header
7from fastapi.templating import Jinja2Templates
8import shutil
9import os
10import json
11import requests
12from datetime import datetime
```

- pathlib.Path: Để làm việc với các đường dẫn tệp.
- fastapi: Các thành phần chính của FastAPI để xây dựng API.
- StaticFiles: Để phục vụ các tệp tĩnh như CSS, JS.
- HTMLResponse, JSONResponse: Để trả về các phản hồi HTML và JSON.
- Jinja2Templates: Để kết xuất các mẫu HTML.
- shutil, os: Để thao tác với tệp và hệ điều hành.
- json: Để làm việc với dữ liệu JSON.
- datetime: Để xử lý thời gian và ngày tháng.

Khởi tạo ứng dụng và các cấu hình ban đầu:

```
1app = FastAPI()
2templates = Jinja2Templates(directory="templates")
3app.mount("/static", StaticFiles(directory="static"), name="static")
4app.mount("/uploaded-images", StaticFiles(directory="uploaded-images"),
      name="uploaded-images")
5CHUNK_SIZE = 1024 * 1024
```

- app = FastAPI(): Khởi tạo ứng dụng FastAPI.
- templates = Jinja2Templates(directory="templates"): Cấu hình Jinja2 cho việc kết xuất các mẫu HTML từ thư mục "templates".
- app.mount: Gắn kết các thư mục tĩnh để phục vụ tệp tĩnh.



- CHUNK-SIZE: Định nghĩa kích thước mảnh cho việc truyền phát video (1 MB).

API để hiển thị trang phát trực tiếp

```
1 @app.get("/streaming/{file-id}")
2 async def read_root(request: Request, file-id: str):
3     return templates.TemplateResponse("streaming-form.html", {"request": request,
4                                         "file-id": file-id})
```

- @app.get("/streaming/file-id"): Xác định API GET tại URL "/streaming/file-id".
- async def read-root(request: Request, file-id: str): Định nghĩa hàm xử lý yêu cầu, nhận request và file-id từ URL.
- templates.TemplateResponse("streaming-form.html", "request": request, "file-id": file-id): Kết xuất mẫu HTML "streaming-form.html" và truyền request cùng file-id vào mẫu.

API để truyền phát video

```
1 @app.get("/video/{file-id}")
2 async def video_endpoint(range: str = Header(None), file-id: str = None):
3     with open("file-info.json", "r") as log_file:
4         data = json.load(log_file)
5     for item in data:
6         if item["file-id"] == file-id:
7             video_path = Path(f"./uploaded-videos/{file-id}{item['ext']}")  

8             break
9     else:
10         raise HTTPException(status_code=404, detail="File not found")
11     start, end = range.replace("bytes=", "").split("-")
12     start = int(start)
13     end = int(end) if end else start + CHUNK-SIZE
14     with open(video_path, "rb") as video:
15         video.seek(start)
16         data = video.read(end - start)
17         filesize = str(video_path.stat().st_size)
18         headers = {
19             'Content-Range': f'bytes {str(start)}-{str(end)}/{filesize}',
20             'Accept-Ranges': 'bytes'
21         }
22         return Response(data, status_code=206, headers=headers, media_type="video/mp4")
```



- @app.get("/video/file-id"): Xác định API GET tại URL "/video/file-id".
- async def video-endpoint(range: str = Header(None), file-id: str = None): Định nghĩa hàm xử lý yêu cầu, nhận range từ tiêu đề HTTP và file-id từ URL.
- with open("file-info.json", "r") as log-file: Mở tệp "file-info.json" ở chế độ đọc.
- for item in data: Tìm tệp video có file-id phù hợp trong dữ liệu.
- if item["file-id"] == file-id: Nếu tìm thấy tệp phù hợp, xác định đường dẫn tệp.
- else: Nếu không tìm thấy, trả về lỗi 404.
- start, end = range.replace("bytes=", "").split("-"): Tách phần bắt đầu và kết thúc của phạm vi byte từ tiêu đề HTTP.
- start = int(start): Chuyển đổi phần bắt đầu thành số nguyên.
- end = int(end) if end else start + CHUNK-SIZE: Chuyển đổi phần kết thúc thành số nguyên hoặc tính toán kích thước mảnh nếu không có phần kết thúc.
- with open(video-path, "rb") as video: Mở tệp video ở chế độ nhị phân.
- video.seek(start): Di chuyển con trỏ tệp đến vị trí bắt đầu.
- data = video.read(end - start): Đọc dữ liệu từ vị trí bắt đầu đến vị trí kết thúc.
- filesize = str(video-path.stat().st_size): Lấy kích thước tệp.
- headers = 'Content-Range': f'bytes {str(start)}-{str(end)}/{filesize}', 'Accept-Ranges': 'bytes': Tạo tiêu đề phản hồi HTTP.
- return Response(data, status_code=206, headers=headers, media_type="video/mp4"): Trả về phản hồi với dữ liệu video và tiêu đề HTTP phù hợp.

API để hiển thị biểu mẫu tải lên video

```
1 @app.get('/upload-video', response_class=HTMLResponse)
2 def get-basic-form(request: Request):
3     return templates.TemplateResponse("upload-form.html", {"request": request})
```

- @app.get('/upload-video', response_class=HTMLResponse): Xác định API GET tại URL "/upload-video" và chỉ định phản hồi dạng HTML.
- def get-basic-form(request: Request): Định nghĩa hàm xử lý yêu cầu, nhận request.
- return templates.TemplateResponse("upload-form.html", "request": request): Kết xuất mẫu HTML "upload-form.html" và truyền request vào mẫu.



API để tải lên video

```
1 @app.post("/upload-video")
2 async def upload-video(title: str = Form(...), file: UploadFile = File(...), image:
3     UploadFile = File(...)):
4     # Generate a unique ID for the file based on the current datetime
5     file-id = datetime.now().strftime("%Y%m%d%H%M%S")
6     -, ext = os.path.splitext(file.filename)
7
8     data = {"file-id": file-id, "title": title, "ext": ext}
9     # Read existing data
10    with open("file-info.json", "r") as log-file:
11        existing-data = json.load(log-file)
12
13    # Append new data
14    existing-data.append(data)
15
16    with open("file-info.json", "w") as log-file:
17        json.dump(existing-data, log-file)
18
19    # Check if the file is a video
20    if ext not in ['.mp4', '.avi', '.mov', '.flv', '.wmv']: # Add or remove extensions
21        as needed
22        raise HTTPException(status-code=400, detail="Invalid file type. Please upload a
23        video file.")
24
25    file-name = f"{file-id}{ext}"
26    # Define the path for the video
27    video-path = f"./uploaded-videos/{file-name}"
28
29    # Save the video file
30    with open(video-path, "wb") as buffer:
31        shutil.copyfileobj(file.file, buffer)
32
33    # Save the image file
34    -, image-ext = os.path.splitext(image.filename)
35    image-name = f"{file-id}{image-ext}"
36    image-path = f"./uploaded-images/{image-name}"
37    with open(image-path, "wb") as buffer:
38        shutil.copyfileobj(image.file, buffer)
39
40    # Return a success response
41    return JSONResponse(content={"message": "Video uploaded successfully"},
42                        status-code=200)
```

- @app.post("/upload-video"): Xác định API POST tại URL "/upload-video".
- async def upload-video(title: str = Form(...), file: UploadFile = File(...), image: UploadFile = File(...)): Định nghĩa hàm xử lý yêu cầu, nhận title từ biểu mẫu, file và image từ tệp tải lên.
- file-id = datetime.now().strftime("
- -, ext = os.path.splitext(file.filename): Tách phần mở rộng của tệp video.



- data = "file-id": file-id, "title": title, "ext": ext: Tạo đối tượng dữ liệu chứa ID tệp, tiêu đề và phần mở rộng.
- with open("file-info.json", "r") as log-file: Mở tệp "file-info.json" ở chế độ đọc.
- existing-data = json.load(log-file): Đọc dữ liệu JSON từ tệp.
- existing-data.append(data): Thêm dữ liệu mới vào dữ liệu hiện có.
- with open("file-info.json", "w") as log-file: Mở tệp "file-info.json" ở chế độ ghi.
- json.dump(existing-data, log-file): Ghi dữ liệu JSON cập nhật vào tệp.
- if ext not in ['.mp4', '.avi', '.mov', '.flv', '.wmv']: Kiểm tra xem tệp có phải là video không.
- raise HTTPException(status-code=400, detail="Invalid file type. Please upload a video file."): Nếu không phải video, trả về lỗi 400.
- file-name = f"file-idext": Tạo tên tệp video.
- video-path = f"./uploaded-videos/file-name": Xác định đường dẫn lưu tệp video.
- with open(video-path, "wb") as buffer: Mở tệp video ở chế độ ghi nhị phân.
- shutil.copyfileobj(file.file, buffer): Lưu tệp video.
- -, image-ext = os.path.splitext(image.filename): Tách phần mở rộng của tệp hình ảnh.
- image-name = f"file-idimage-ext": Tạo tên tệp hình ảnh.
- image-path = f"./uploaded-images/image-name": Xác định đường dẫn lưu tệp hình ảnh.
- with open(image-path, "wb") as buffer: Mở tệp hình ảnh ở chế độ ghi nhị phân.
- shutil.copyfileobj(image.file, buffer): Lưu tệp hình ảnh.
- return JSONResponse(content="message": "Video uploaded successfully", status-code=200): Trả về phản hồi JSON thành công với thông báo.



5 Ưu và nhược điểm của đồ án

5.1 Ưu điểm

- Hiệu suất cao:** Sử dụng FastAPI, một framework nhẹ và nhanh, giúp giảm thời gian phản hồi và tăng khả năng xử lý của ứng dụng.
- Dễ phát triển và mở rộng:** FastAPI hỗ trợ OpenAPI và JSON Schema, giúp dễ dàng mở rộng và bảo trì ứng dụng.
- Giao diện người dùng thân thiện:** Sử dụng Jinja2 để tạo các template HTML, giúp giao diện người dùng trực quan và dễ sử dụng.
- Khả năng phát video mượt mà:** Sử dụng kỹ thuật chunked transfer encoding để phát video, giảm tải cho server và cải thiện trải nghiệm người dùng.
- Bảo mật:** FastAPI hỗ trợ quản lý các tệp tải lên một cách an toàn, kiểm tra loại tệp và kích thước tệp trước khi xử lý.

5.2 Nhược điểm

- Phụ thuộc vào cơ sở hạ tầng:** Việc phát video trực tuyến yêu cầu cơ sở hạ tầng mạng mạnh mẽ và băng thông cao, đặc biệt khi có nhiều người dùng truy cập cùng lúc.
- Khả năng mở rộng hạn chế:** Mặc dù FastAPI nhanh và nhẹ, nhưng khi số lượng người dùng và video tăng lên, việc mở rộng ứng dụng có thể gặp khó khăn nếu không có các biện pháp tối ưu hóa phù hợp.
- Bảo mật và quyền riêng tư:** Việc quản lý video của người dùng yêu cầu các biện pháp bảo mật mạnh mẽ để đảm bảo quyền riêng tư và an toàn dữ liệu, điều này có thể phức tạp và tốn kém.
- Yêu cầu kiến thức chuyên sâu:** Để triển khai và bảo trì một ứng dụng như vậy, yêu cầu kiến thức chuyên sâu về lập trình, mạng, và bảo mật.
- Hạn chế về định dạng video:** Hiện tại, ứng dụng chỉ hỗ trợ một số định dạng video phổ biến. Việc mở rộng để hỗ trợ thêm các định dạng khác có thể phức tạp và cần thêm thời gian phát triển.