

팀 구성원

- 백종록(교육학과, 팀장)
 - 김상윤(산업시스템공학과)
 - 장지욱(경찰행정학부)
-

프로젝트 개요

Youtube 영상 링크와 원하는 시간대를 입력 받아 gif 파일로 변환하여 사용자 컴퓨터로 다운로드 하는 웹사이트 개발

프로젝트 추진 배경

Youtube영상을 다운로드 받는 기능을 제공하는 사이트들은 조금만 검색해도 많이 찾을 수 있지만, 해당 영상을 원하는 시간대만 추출하여 GIF파일, 소위 말하는 '짤'로 변환해주는 서비스는 쉽게 찾아보기 어렵고, 유튜브에서 영상을 다운받거나 로컬에 존재하는 동영상 파일을 gif파일로 변경해주는 여러 오픈소스 소프트웨어들이 존재하지만 프로그래밍 경험이 없는 일반인이 CLI기반의 소프트웨어를 사용하기에는 다소 진입장벽이 있음. 또한 최근 Youtube 쇼츠 등의 짧은 길이의 영상이 인기를 끌고 있는 만큼, 영상의 원하는 부분만 소장하고자 하는 사람들의 요구사항이 존재함.

프로젝트 변경사항

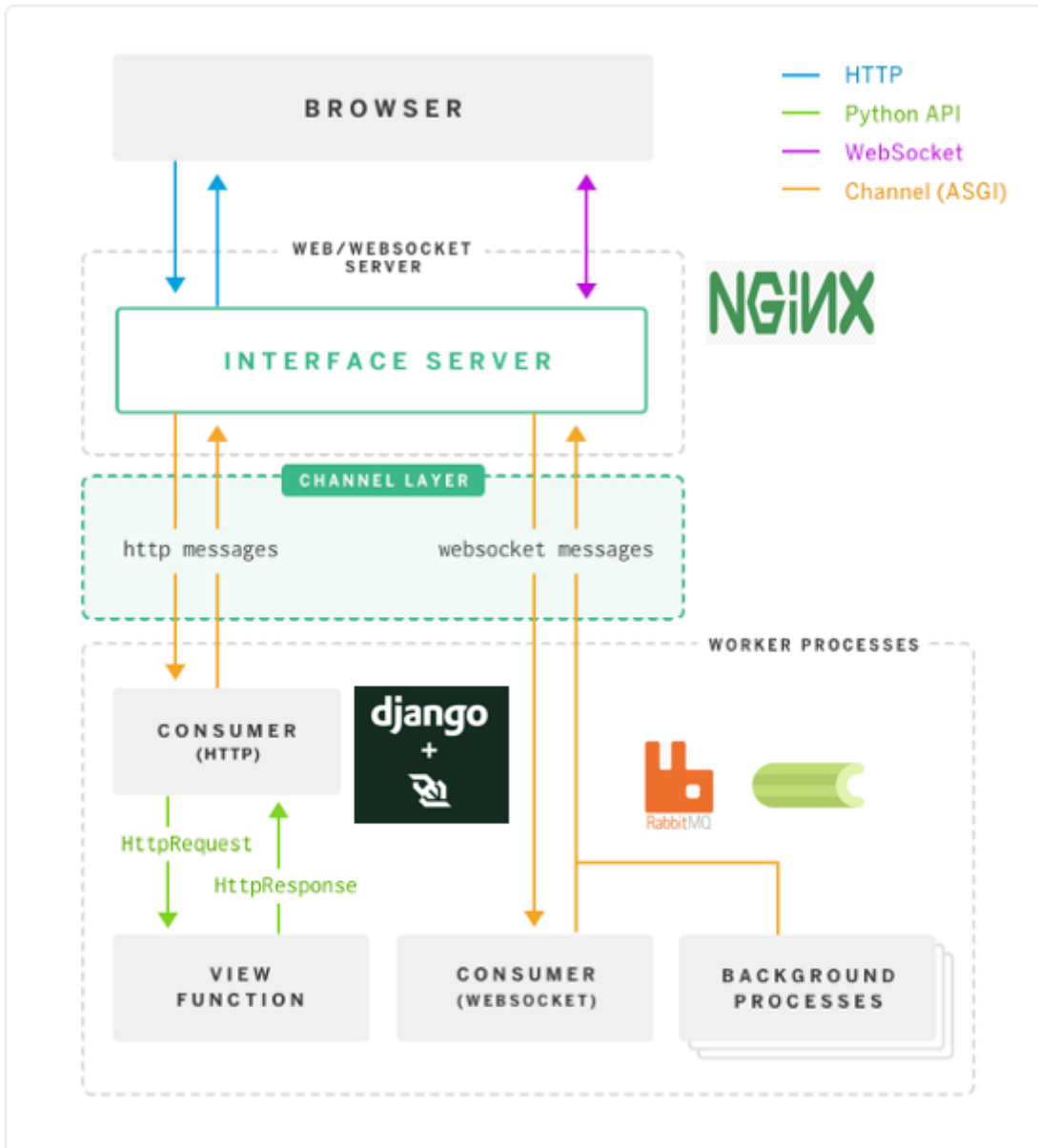
중간 발표까지는 Get, Post 방식을 이용하여 통신을 진행하였다. Get post 방식의 경우 지속적으로 Get 요청을 하여야 Client에게 server가 응답을 할 수 있다. 응답하더라도 Youtube 다운로드가 다 되지 않는다면 HTML 파일 전체를 응답해야 하므로 부담이 컸다. 즉 지속적인 client이 GET 요청, 서버의 페이지 전체 응답이 효율적이지 않았다.

이후 비효율적인 통신 방식을 변경하기 위해서 HTTP/GET, HTTP/POST 방식이 아니라 websocket을 이용하기로 하였다. 처음 홈페이지 URL에 접속하면 HTTP/GET 요청으로 페이지를 받아오고 websocket연결을 한다. 이후 통신은 websocket을 이용하여 진행한다. websocket은 client에게 요청이 들어오지 않더라도 server가 client에게 정보를 보내줄 수 있다. client가 Youtube download 요청을 하면 celery를 통해 비동기적으로 다운로드가 진행되고, 일정 시간마다 server가 client에게 상태를 보내준다. 다운로드가 완료되면 완료 메시지를 client에게 보내고, 메시지를 받는 즉시 client가 Gif 파일을 요청하고 server는 Filerespnse로 보내준다.

- UI 변경 (모두 최고 화질로 RETURN해주기에 화질 선택창 제거 및 git,youtube 링크 추가)
- UI 변경에 따른 forms.py, model.py 변경
- Ubuntu에서 Daphne, uwsgi, Nginx를 Demone으로 돌리기 위한 ini, service 파일 작성
- Websocket 연결을 위한 sever측면의 asgi.py, socket.py, routing.py, settings.py 파일 수정 및 작성
- Get/Post 방식을 담당하는 Views.py 수정

- Websocket 연결, 메세지 수신, 메세지 발신, 상태확인 등 Websocket 통신을 위한 Javascript 코드 작성
- 기존에는 convert 버튼을 한 번 누르면 disable이 되었는데, websocket으로 다운로드 정보를 받아 다운로드 중이면 disable, 종료가 되면 able이 되도록 변경
- Ubuntu 환경에서 FFMEG 설치 및 설정 및 ws 통신 포트 설정

시스템 구조



web server는 NGINX를 사용하였습니다.
websocket 통신은 위해 Daphne를 사용하였습니다.
Youtube 다운을 비동기로 하기 위해 celery를 사용하였습니다.
celery의 메세지 브로커로는 Rabbitmq를 사용하였습니다.

동작방식

1. user가 url로 들어와서 HTTP Get 요청을 하면 Nginx가 Uwsgi로 연결해주고 Django해서 응답을 해줍니다.
2. user가 값을 입력하면, 입력한 값이 적절한지 Javascript가 확인하고 적절하면 websocket으로 내용을 보내고 아니면 alert 창을 보낸다.
3. ws:// 으로 들어오면 Nginx가 Daphne로 연결하여 Django websocket과 연결 될 수 있게 해줍니다.
4. Django는 입력값을 바탕으로 celery로 Youtube 다운로드를 비동기로 처리합니다.
5. user는 일정 시간마다 websocket으로 다운로드 되고 있는 상태를 요청하고 Django는 진행 상태를 user에게 알려줍니다.
6. 다운로드가 완료되면 gif file을 return 합니다.

입력값은 1초 이상 6초 미만입니다.

이 범위를 넘어갈 경우 안내 문구가 등장합니다.

간략한 코드로 보는 동작 방식

1. URL로 들어오면 Get 요청에 대한 응답

```
# urls.py URL 확인 후 올바른 view로 가도록 지시
...
urlpatterns = [
    path('',views.MainView.as_view(), name = 'home'),
    ...
# Views.py
...
class MainView(View):
    template_name = 'gif_to_mp4/index.html'

    def get(self, request):
        form = URLform()
        ctx = {'form':form}
        return render(request, self.template_name, ctx)
    ...
```

2. User에서 웹소켓 연결

```
# 웹소켓 연결
const GifSocket = new WebSocket(
    'ws://'
```

```

+ window.location.host + ":8000"
+ '/'
);
# Youtube 링크 및 정보 server에 보내는 코드
function clickSubmit(this1){
    var url = document.querySelector('#youtube_link').value;
    var s_m = document.querySelector('#start_minute').value;
    var s_s = document.querySelector('#start_second').value;
    var e_m = document.querySelector('#end_minute').value;
    var e_s = document.querySelector('#end_second').value;
    var diff = 60 * e_m + e_s - (60 * s_m + s_s)
    console.log(diff)
    if (5 < diff){
        alert("최대 변환 길이는 5초 입니다")
        document.getElementById("message").textContent = "최대 변환 길이는 5초 입니다.";
    }else if (diff < 0){
        alert("1초 이상의 값을 입력해 주세요.")
        document.getElementById("message").textContent = "1초 이상의 값을 입력해 주세
요.";
    }else if (diff < 6 && 0 < diff){
        spinner.style.visibility = 'visible';
        document.getElementById("submitButton").disabled = true;
        console.log("form 보냄 socekt도 보냄")
        GifSocket.send(data = JSON.stringify({
            'status' : '',
            'youtube_link' : url,
            'start_minute': s_m,
            'start_second' : s_s,
            'end_minute' : e_m,
            'end_second' : e_s
        )))
        // this1.form.submit(); 그냥 socket으로 처리
        document.getElementById("message").textContent = "동영상을 다운로드 중 입니다.
새로고침을 하지 말아주세요";
        this1.form.reset();
    }
    else{
        alert("올바른 값을 입력해 주세요")
        document.getElementById("message").textContent = "올바른 값을 입력해 주세요";
    }
}
}

```

3. Server에서 websocket 응답 처리

```

# routing.py 일부 소켓 path에 따른 연결 정보 처리
websocket_urlpatterns = [
    re_path(r'', sockets.GifConsumer.as_asgi()),
# sockets.py 일부 메세지 받으면 다운로드 처리
def receive(self, text_data):
    data_json = json.loads(text_data)

```

```

if not data_json['status']:
    print(data_json['status'])
    url = data_json["youtube_link"]
    start_min = int(data_json["start_minute"])
    start_sec = int(data_json["start_second"])
    end_min = int(data_json["end_minute"])
    end_sec = int(data_json["end_second"])
    # print(start_min,type(start_min))
    ss = f"00:{start_min:02}:{start_sec:02}.00"
    to = f"00:{end_min:02}:{end_sec:02}.00"
    dic = {'url' : url, "ss" : ss, "to" : to}
    data = json.dumps(dic, indent = 4)
    self.t = downloand_video.delay(data) # celery
    self.send(text_data=json.dumps({
        'message': "Doing"
    }))
    ...

```

4.Youtube download celery를 통한 비동기 처리

```

# task.py
...
@shared_task()
def downloand_video(data):
    try:
        data = json.loads(data)
        url, ss ,to = data['url'], data['ss'],data['to']

        if os.path.exists('video.mp4'):
            os.remove('video.mp4')

        if os.path.exists('video.gif'):
            os.remove('video.gif')

        ydl_opts = {
            'format': "best",
            'videoformat' : "mp4",
            'outtmpl' : "video.mp4",
            'external_downloader': 'ffmpeg',
            'external_downloader_args': ["-ss", ss, "-to", to],
        }

        with youtube_dl.YoutubeDL(ydl_opts) as ydl:
            ydl.download([url])
            title = "video"
            clip = VideoFileClip(title + '.mp4')
            clip.write_gif(title + '.gif')
            clip.close()
    except:

```

```
raise Ignore()
```

5. client의 다운로드 상태확인 요청과 그 응답

```
function send_message(){
  GifSocket.send(data = JSON.stringify({
    'status' : 'yes'
  }))
}
...
# 메세지 확인하며 상태에 따라 보내는 함수 일부
sleep(6000);
send_message();
console.log("status 물어보는 중")
```

server의 응답

```
if self.t and self.t.ready(): # 작업 끝
    self.send(text_data = json.dumps({
        'message': "Done" # 완료 메세지
    }))
else: # 작업 진행 중
    self.send(text_data = json.dumps({
        'message': "Doing"
    }))
```

6. 다운로드 완료 후 그 응답

client

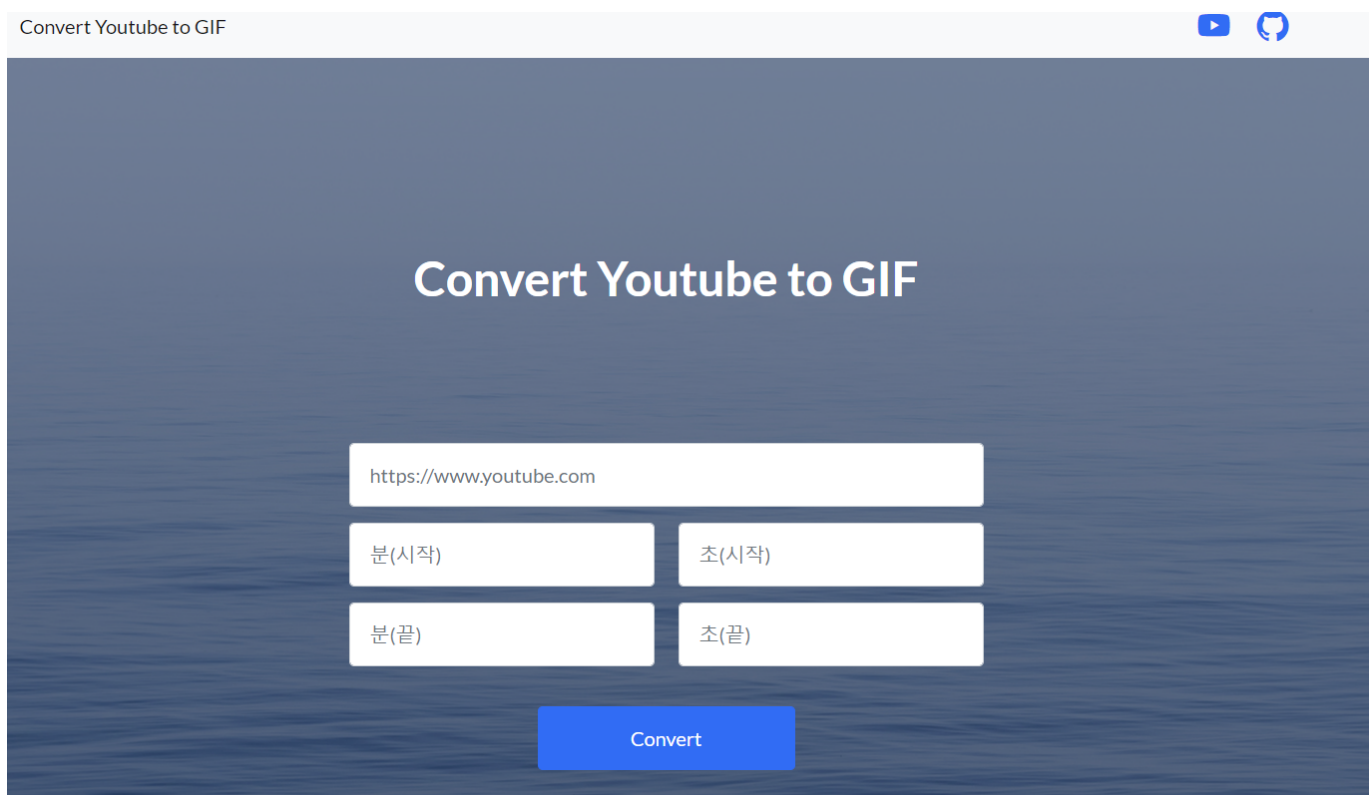
```
if (data.message == 'Done'){
  window.location = 'gif'; # gif 요청
  document.getElementById("submitButton").disabled = false;
  document.getElementById("message").textContent = "동영상이 모두 다운로드 되었습니다!";
  spinner.style.visibility = 'hidden';
```

server

```
def gif(request):  
    title = 'video'  
    file_path = os.path.abspath("./")  
    file_name = os.path.basename("./" + title + ".gif")  
    fs = FileSystemStorage(file_path)  
    response = FileResponse(fs.open(file_name, 'rb'),  
                             content_type='image/gif')  
    response['Content-Disposition'] = f'attachment; filename=video.gif'  
    return response
```

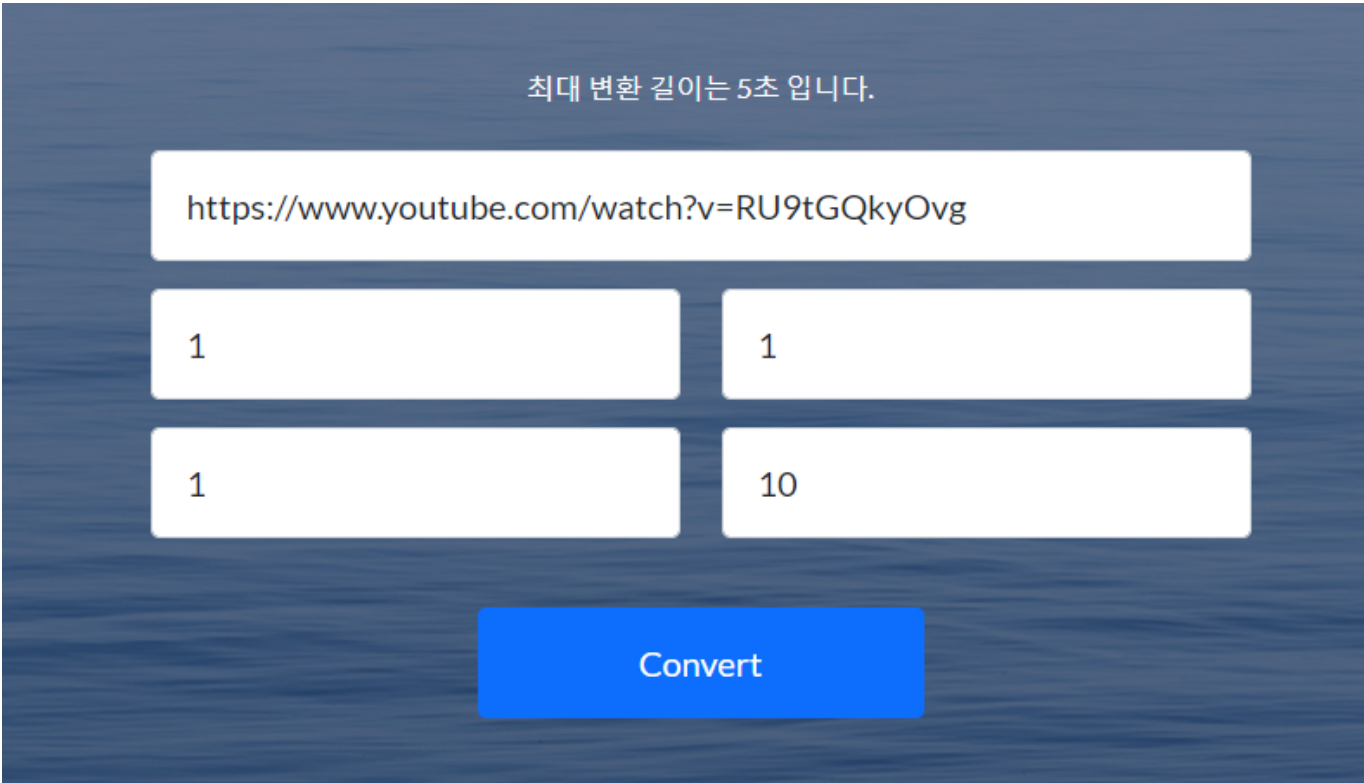
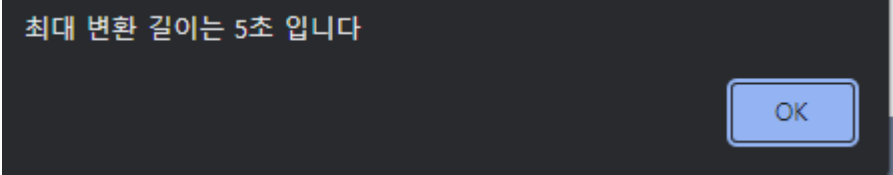
동작화면

1. 화면

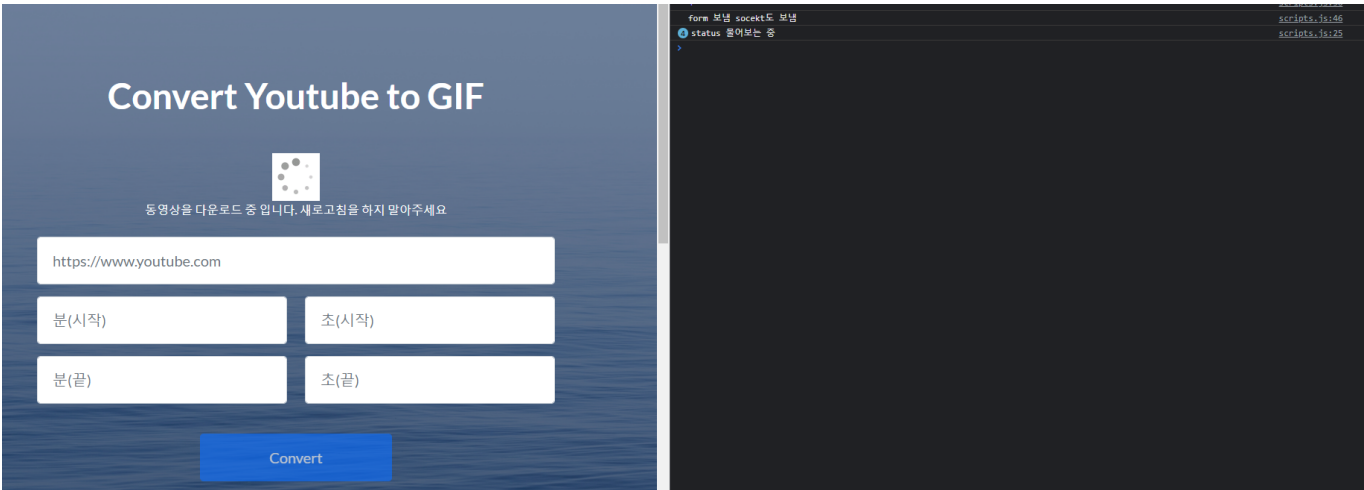


The screenshot shows a web browser window with the title 'Convert Youtube to GIF'. The page has a dark blue background with a white title 'Convert Youtube to GIF' in the center. Below the title, there is a form with a text input field containing 'https://www.youtube.com'. Underneath the input field, there are four smaller input fields arranged in a 2x2 grid: '분(시작)' (minutes start), '초(시작)' (seconds start), '분(끝)' (minutes end), and '초(끝)' (seconds end). At the bottom of the form, there is a blue button labeled 'Convert'.

2. 입력값 초과



3. 다운로드 및 완료



동영상이 모두 다운로드 되었습니다!

<input type="text" value="분(시작)"/>	<input type="text" value="초(시작)"/>
<input type="text" value="분(끝)"/>	<input type="text" value="초(끝)"/>

video (2).gif

License

여기만 추가해 주세요