



Media Web App Term Project

MediaWebAppTeam5

김하늘, 김유신, 나호영, 심태형

TABLE OF CONTENTS

01

역할 분담

02

트러블 슈팅

가장 인상적이었던 문제 해결 과정

03

교훈

04

프로젝트 시연



01

강민
권영
역할



김하늘
hamuneulbo

김하늘

서강대학교 심리학 / 컴퓨터공학
Github: @hamuneulbo

팀장

Front-end 담당

WebOS Design

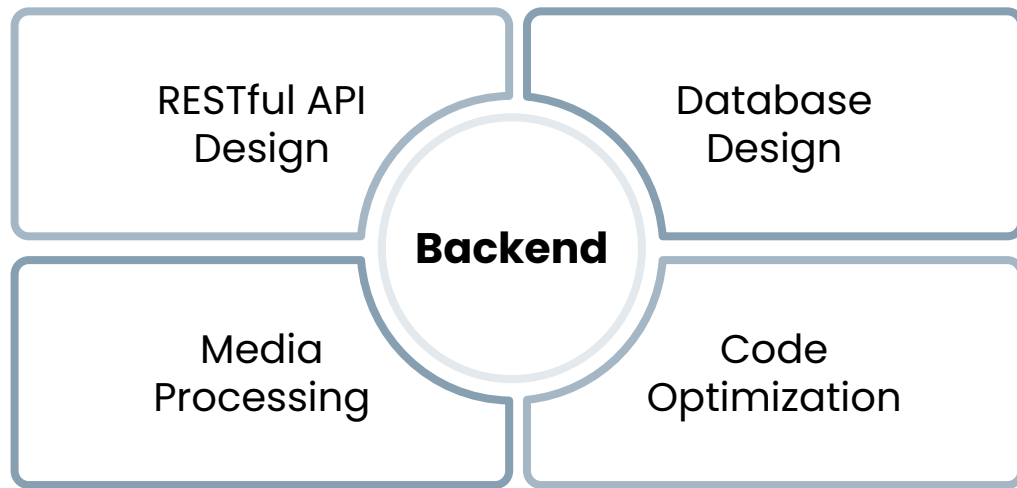


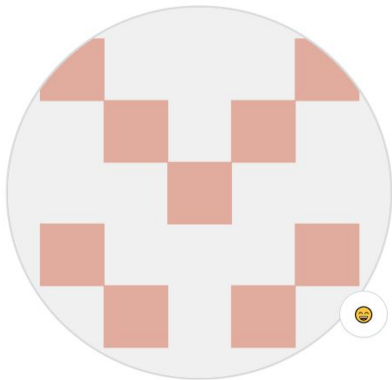
YUSHIN KIM
Yush1nk1m

김유신

서강대학교 심리학 / 컴퓨터공학 / 인공지능

Github: [@Yush1nk1m](#)



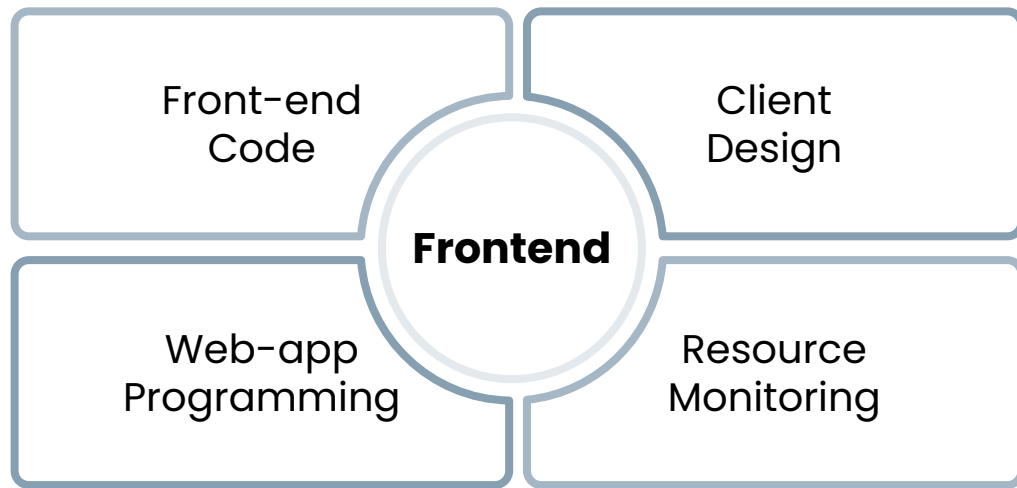


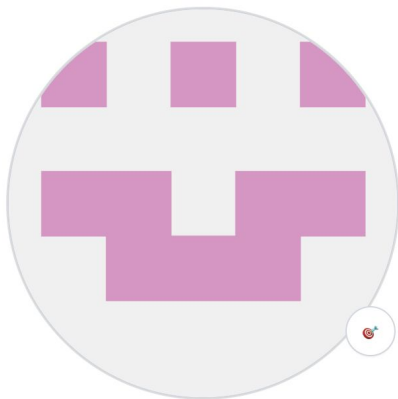
sth
sthsssss

심태형

서강대학교 경제학 / 컴퓨터공학

Github: @sthsssss



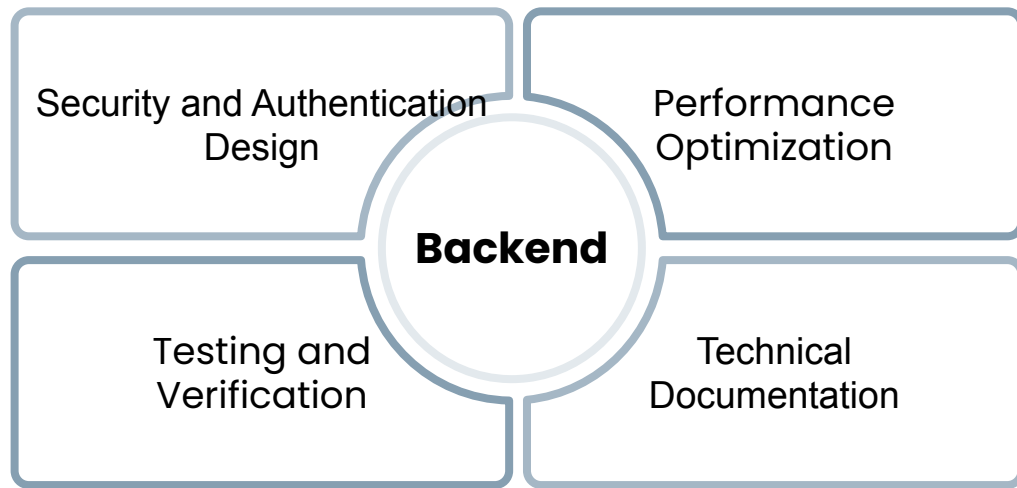


NHY
HOYE0NG

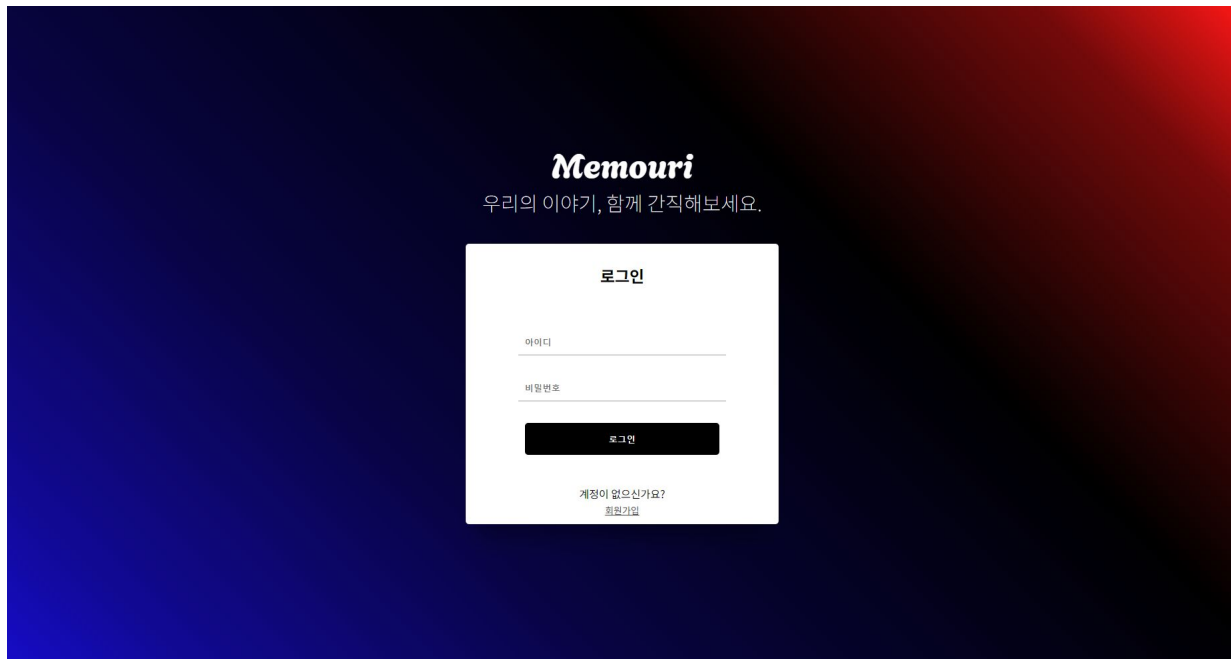
나호영

서강대학교 컴퓨터공학

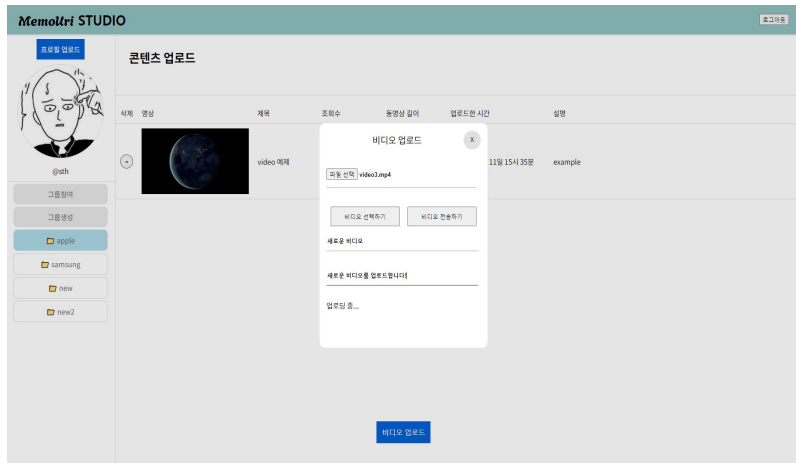
Github: @HOYE0NG



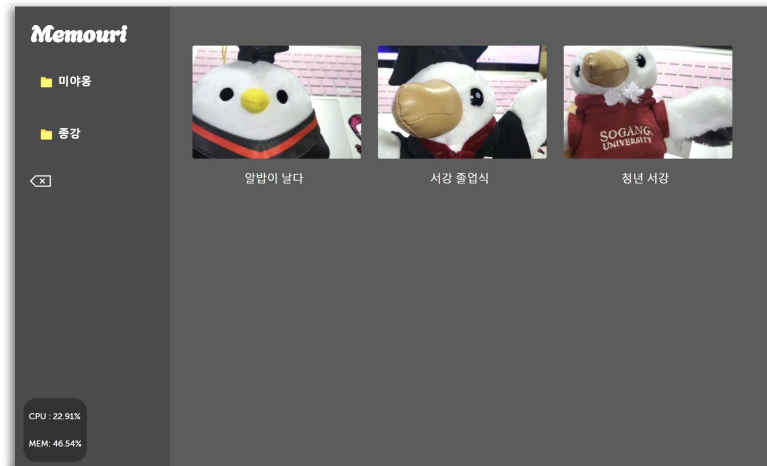
프로젝트 소개 - Frontend



프로젝트 소개



Upload Client



WebOS

프로젝트 소개 - Backend

QR code Login

TV 환경을 고려한 편리한 로그인 기능



프로젝트 소개 - Backend

Group-based interaction

그룹: 사용자 간 상호작용의 기본적인 단위

Personalized experience

이어보기 시점, 핀, 좋아요 등

프로젝트 소개 - Backend

HLS Adaptive Streaming

네트워크 환경을 고려한 신속한 영상 송출

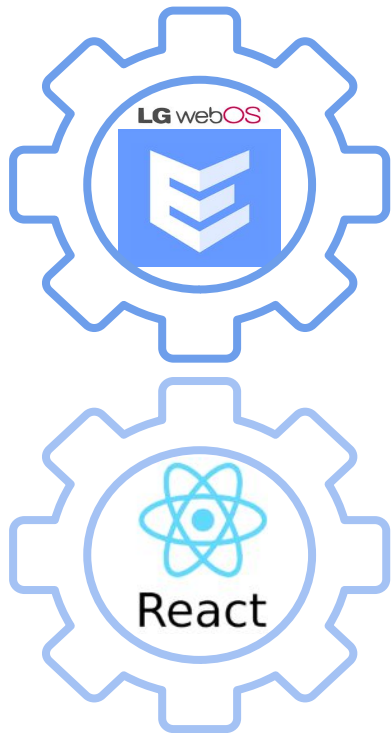
Dynamic thumbnail switch

최근 이어 본 지점에 대한 단서 제공

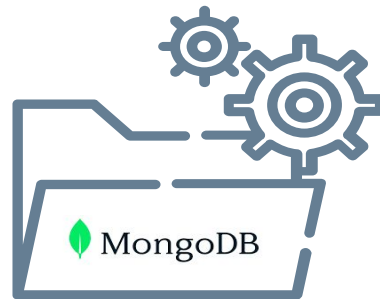
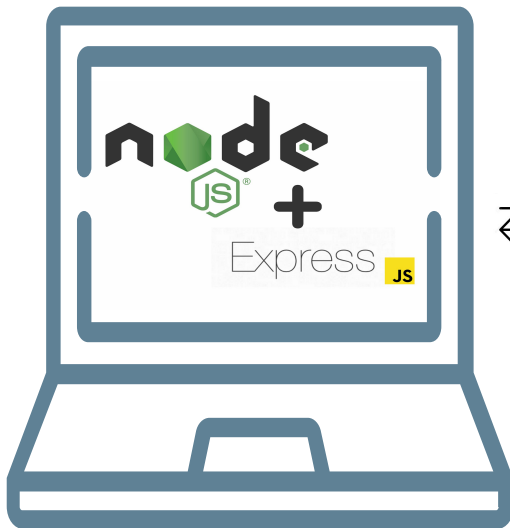
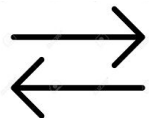
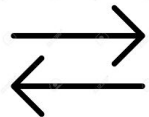


프로젝트 기술 스택

Front-end



Back-end



Common



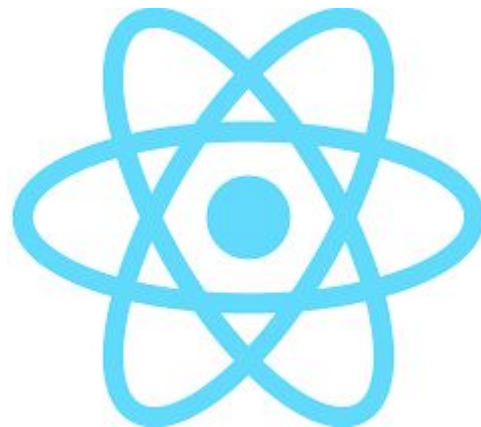
02

트러블 슈팅

가장 인상적이었던 문제 해결 과정

LS2 API & React 문법들..

Luna
Service
API



CSS

CSS



sandstone/ThemeDecorator

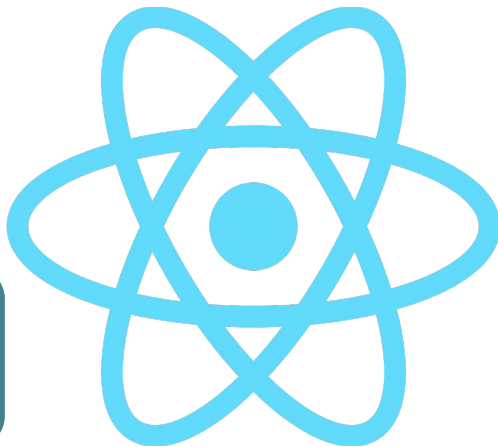


Exports the [sandstone/ThemeDecorator.ThemeDecorator](#) HOC

```
import ThemeDecorator from '@enact/sandstone/ThemeDecorator';
```

[import usage](#)

라우팅



Browser Router

- HTML5의 History API 이용
- 검색 엔진 최적화에 적합
- 웹에서만 동작

Hash Router

- 주소에 # 붙음
- 검색 엔진 최적화 사용 안함
- **WebOS** 에서 동작

기획 이슈

애자일 모델



워터폴 모델

- 비효율적 기반 코드 위 코딩

-> 리팩토링 시간 소요

- 절차적 프로그래밍

- 분석 > 설계 > 구현 > 테스트

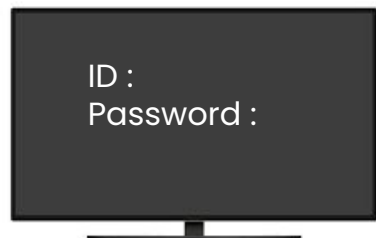
체계적인 파일 구조,

상속 컴포넌트의 정의 필요성

나호영, QRCode Login 기능 구현



나호영, QRCode Login 기능 구현

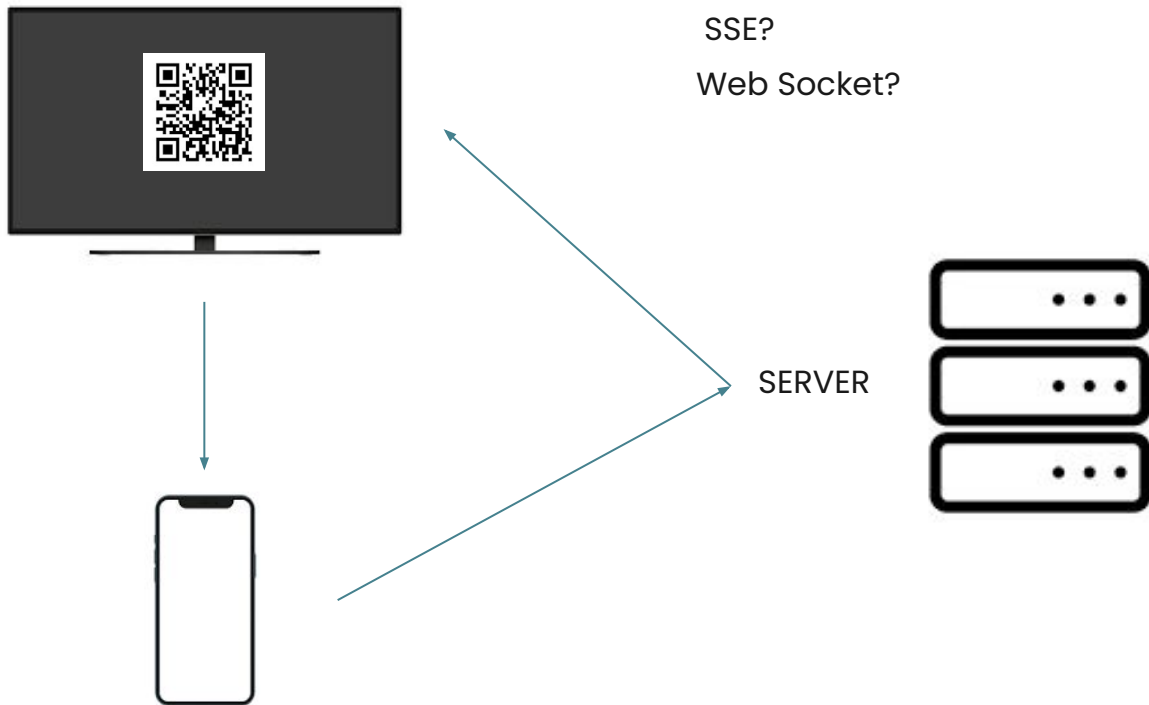


로그인 링크
접속

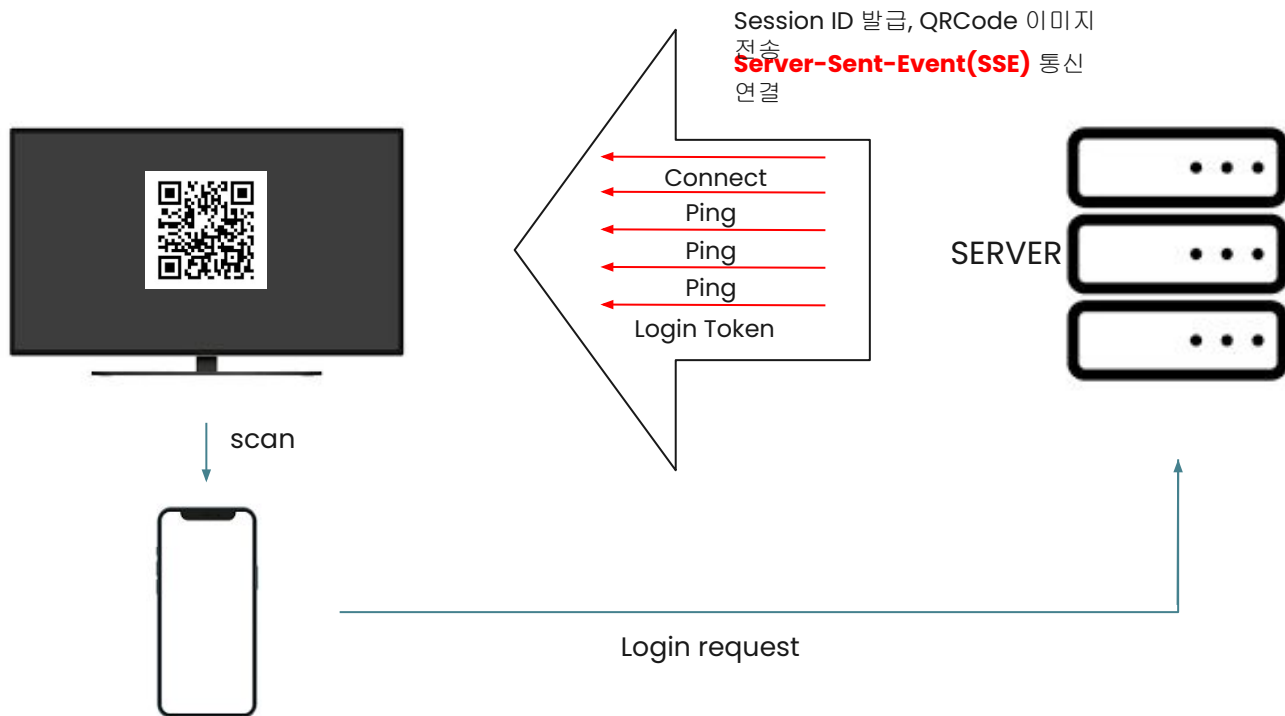
로그인 성공



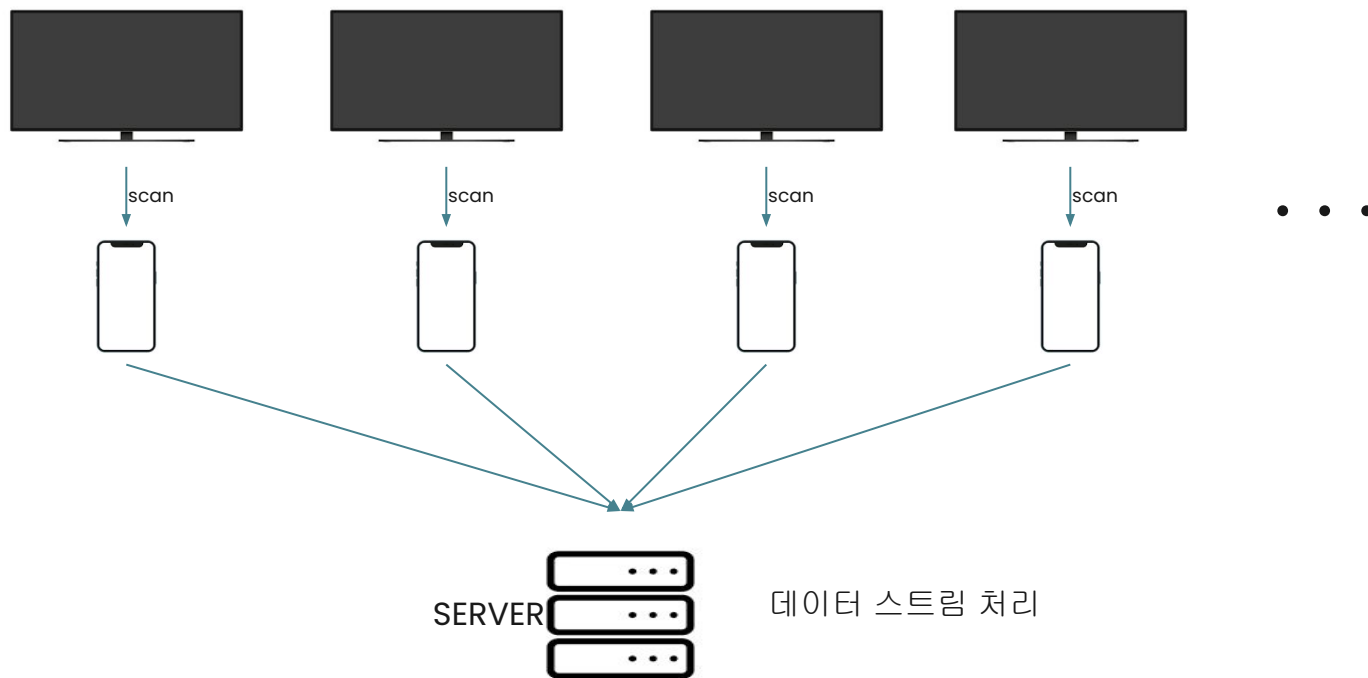
통신 방식, 기술 선택에 대한 고민



통신 방식, 기술 선택에 대한 해결 방안



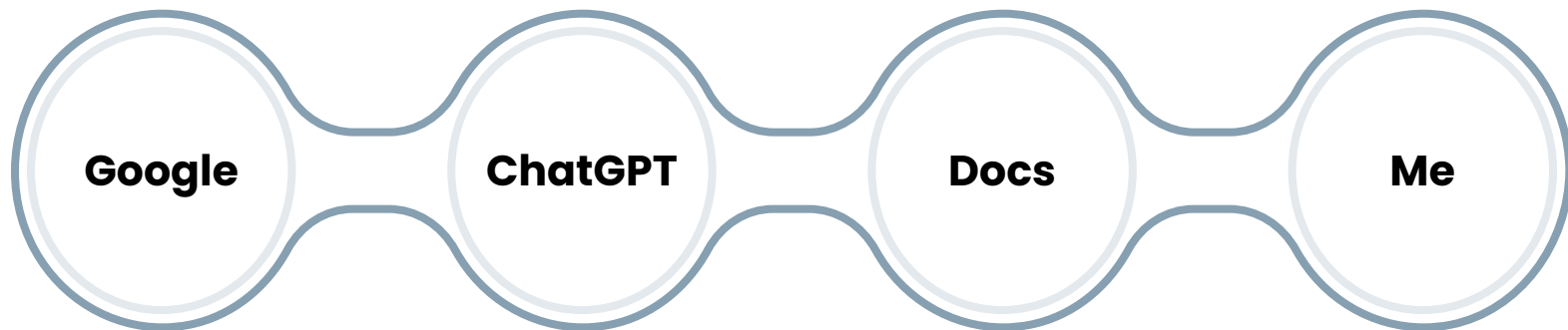
다중 클라이언트 동기화 문제와 해결 방안



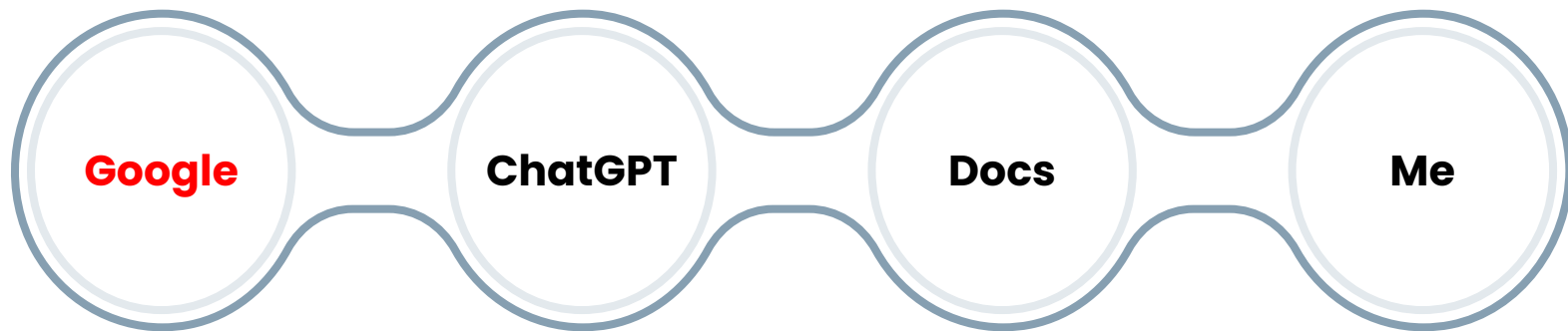
QRCode Login 기능 구현 - Lesson

- 엔지니어로서의 핵심은 사용자 경험을 개선하는 것
- 아이디어를 생각해내는 것보다 그것을 구현하는 것이 중요
- 다양한 기술적인 고민...
- 개발자로서의 성장

김유신, **Adaptive Streaming** 기능 구현



Adaptive Streaming 기능 구현 - Google



Adaptive Streaming 기능 구현 - Google



- 필요한 기술을 빠르게 찾을 수 있다.
- 대부분의 코드는 작동이 보장된다.
- 기능 구현에 필요한 로직을 찾기 어렵다.

#HLS #FFmpeg

nodejs 실시간 스트리밍

동영상 이미지 뉴스 쇼핑 도서 지도 항공편 금융

About 17,300 results (0.23 seconds)

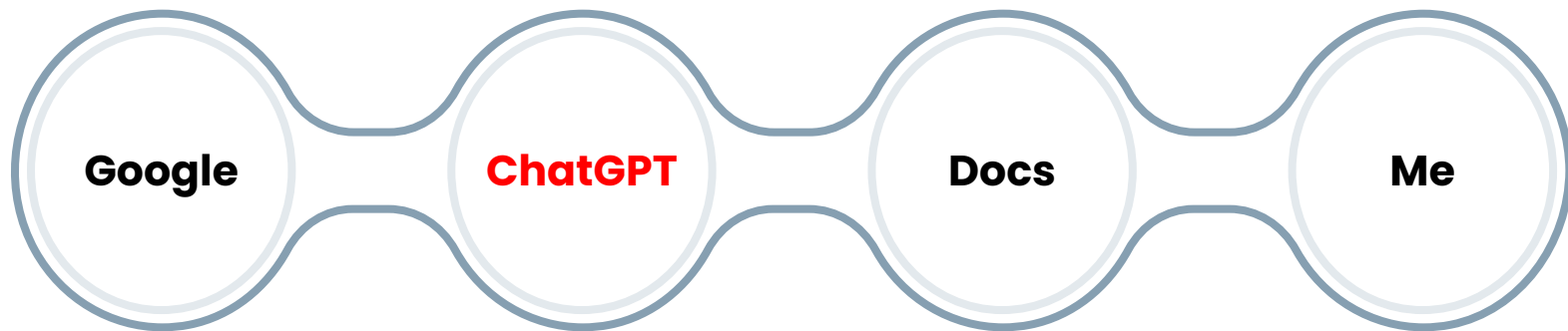
gitbooks.io
https://javafa.gitbooks.io/content/chapter11
11. Streaming 서비스 · node.js 서버구축하기 - javafa
현재 인터넷에서 서비스 되고 있거나 오픈소스로 지원하는 **streaming** 라이브러리가 많이 있지만 **node.js** 는 아주 적은량의 코딩만으로 이런 **streaming** 서비스를 쉽게 ...

Tistory
https://madchick.tistory.com > ...
node.js로 스트리밍 서버 구축하기
Jan 26, 2022 — **node.js**를 이용하여 간단한 비디오 스트리밍 서버를 코딩하는 방법입니다. 상용 비디오 스트리밍 서버의 모습은 아니지만, 큰 그림에서 스트리밍 서버 ...

Microsoft Learn
https://learn.microsoft.com > Learn > 자습서
Node.js 및 TypeScript를 사용하여 Media Services로 라이브 ...
Jul 17, 2023 — **Node.js**, **TypeScript** 및 **OBS Studio**를 사용하여 라이브 이벤트를 스트리밍하는 방법을 알아봅니다.
사전 요구 사항 · 샘플 다운로드 및 구성

Medium · Alvin
70+ likes · 3 years ago
Node.js, Express, HLS로 동영상 스트리밍하기 | by Alvin
May 5, 2020 — 말그대로 HTTP 프로토콜을 사용하는 실시간 스트리밍 방식입니다. HLS는 스트리밍에 .m3u8, .ts 형식의 파일을 사용합니다. .m3u8: 스트리밍할 동영상을 ...

Adaptive Streaming 기능 구현 - ChatGPT



Adaptive Streaming 기능 구현 - ChatGPT



- 기능 구현에 필요한 로직을 생성할 수 있다.
- 오류 발생 ⇒ 이성의 끈 없음 ⇒ 감정적인
쿼리 제공 ⇒ 더 심각한 오류 발생 ⇒ ...

#fluent-ffmpeg
#hls-parser

You

그러면 videos/myId 디렉터리에 영상들이 저장되어 있다고 합시다. 정확히는 videos/myId/144p에는 144p 영상의 세그먼트들이, videos/myId/240p에는 240p 영상의 세그먼트들이 저장되어 있습니다. 이렇게 144p, 240p, 360p, 480p, 720p, 1080p 영상들의 세그먼트들이 존재할 때 하나의 플레이리스트에 해상도, 비트 전송률별 플레이리스트 정보를 저장하는 코드를 작성해보세요



ChatGPT

해상도 및 비트레이트에 따라 Adaptive Streaming을 위한 m3u8 플레이리스트를 생성하는 코드를 작성해보겠습니다. 이 코드는 주어진 해상도와 비트레이트에 따라 플레이리스트 정보를 생성하고, 해당 정보를 m3u8 파일에 저장하는 기능을 가지고 있습니다.

javascript

Copy code

```
const fs = require('fs');
const path = require('path');

// 예시로 사용할 해상도 및 비트레이트 정보
const resolutions = [
  { name: '144p', width: 256, height: 144, bitrate: 192 },
  { name: '240p', width: 426, height: 240, bitrate: 384 },
  { name: '360p', width: 640, height: 360, bitrate: 768 },
  { name: '480p', width: 854, height: 480, bitrate: 1024 },
  { name: '720p', width: 1280, height: 720, bitrate: 2048 },
  { name: '1080p', width: 1920, height: 1080, bitrate: 4096 }
];

// 플레이리스트 생성 함수
```

Adaptive Streaming 기능 구현 - ChatGPT



- 기능 구현에 필요한 로직을 생성할 수 있다.
- 오류 발생 ⇒ 이성의 끈 없음 ⇒ 감정적인
쿼리 제공 ⇒ 더 심각한 오류 발생 ⇒ ...

#fluent-ffmpeg
#hls-parser

You

위에서 마스터 플레이리스트에 variant 추가하는 코드를 보여주셨는데, 오류가 나네요, variants라는 속성은 존재하지 않는다고 합니다. 다시 한번 확인해주세요

ChatGPT

제가 이전에 제공한 코드에 오류가 있었습니다. `hls-parser` 라이브러리에서는 `MasterPlaylist` 객체의 `variants` 속성이 아니라 `playlists` 속성을 사용하여 하위 플레이리스트를 추가해야 합니다. 각 하위 플레이리스트는 `VariantStream` 객체로 만들어져야 합니다.

다음은 올바르게 수정된 코드입니다:

```
javascript Copy code

const { MasterPlaylist, VariantStream } = require("hls-parser");

// MasterPlaylist 생성
const masterPlaylist = new MasterPlaylist();

// 144p 플레이리스트 생성
const playlist144p = new VariantStream({
  uri: "videos/myId/144p.m3u8",
  // 해상도와 비트 전송률 정보를 추가할 수 있습니다.
  resolution: { width: 256, height: 144 },
  bandwidth: 100000,
});
```

Adaptive Streaming 기능 구현 - ChatGPT



- 기능 구현에 필요한 로직을 생성할 수 있다.
- 오류 발생 ⇒ 이성의 끈 잃음 ⇒ 감정적인
쿼리 제공 ⇒ 더 심각한 오류 발생 ⇒ ...

#fluent-ffmpeg
#hls-parser



You

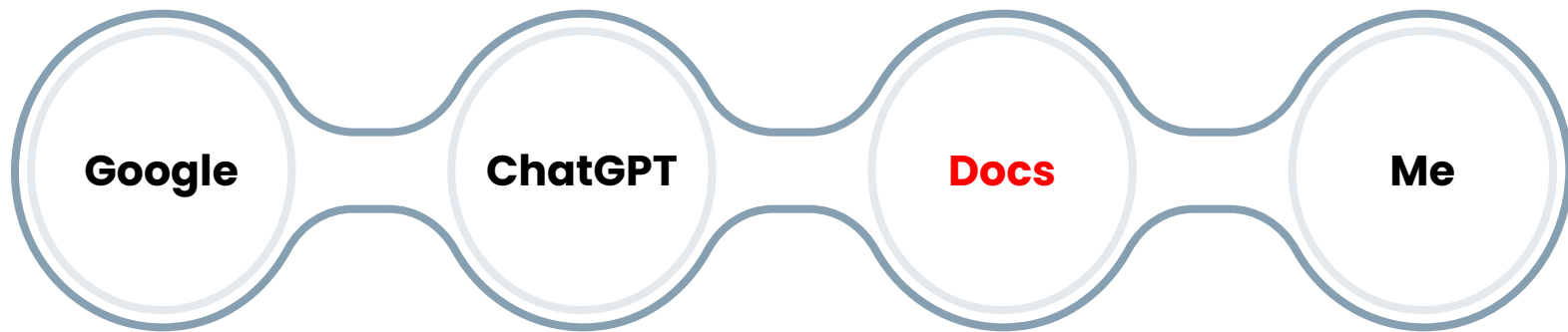
해봤는데 안 되잖아



ChatGPT

죄송합니다. 코드에서 실수가 있었습니다. `hls-parser` 라이브러리에서는 `variants`가 아니라 `playlists` 배열을 사용해야 합니다. 다음은 올바르게 수정된 코드입니다:

Adaptive Streaming 기능 구현 - Docs



Adaptive Streaming 기능 구현 - Docs



“결국, 유능한 개발자가 되기 위해선
프로그래밍 언어가 아닌
영어를 잘해야 한다.”

#fluent-ffmpeg
#hls-parser

≡ README.md

Video frame size options

The following methods enable resizing the output video frame size. They all work together to generate the appropriate video filters.

size(size): set output frame size

Aliases: `videoSize()`, `withSize()`.

This method sets the output frame size. The `size` argument may have one of the following formats:

- `640x480`: set a fixed output frame size. Unless `autopad()` is called, this may result in the video being stretched or squeezed to fit the requested size.
- `640x?`: set a fixed width and compute height automatically. If `aspect()` is also called, it is used to compute video height; otherwise it is computed so that the input aspect ratio is preserved.
- `?x480`: set a fixed height and compute width automatically. If `aspect()` is also called, it is used to compute video width; otherwise it is computed so that the input aspect ratio is preserved.
- `50%`: rescale both width and height to the given percentage. Aspect ratio is always preserved.

Note that for compatibility with some codecs, computed dimensions are always rounded down to multiples of 2.

```
ffmpeg('/path/to/file.avi').size('640x480');  
ffmpeg('/path/to/file.avi').size('640x?');  
ffmpeg('/path/to/file.avi').size('640x?').aspect('4:3');  
ffmpeg('/path/to/file.avi').size('50%');
```

aspect(aspect): set output frame aspect ratio

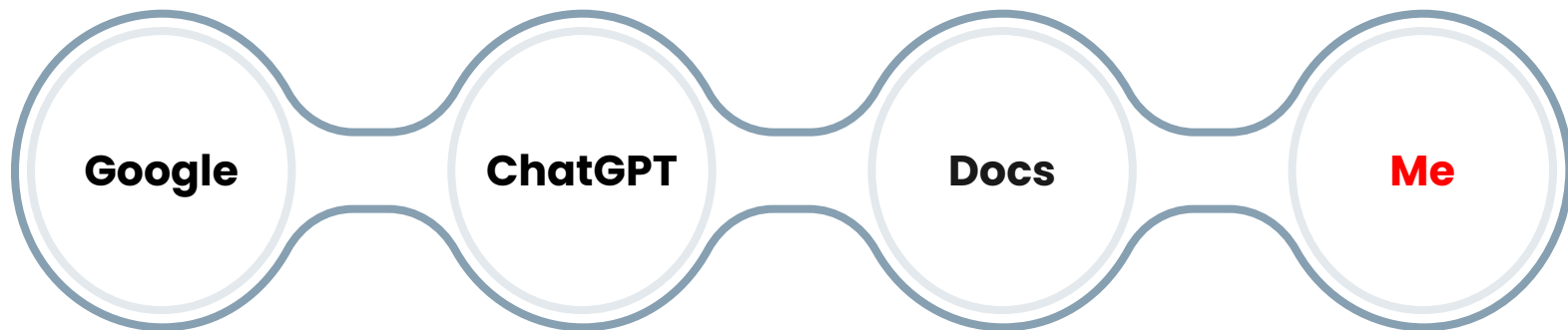
Aliases: `withAspect()`, `withAspectRatio()`, `setAspect()`, `setAspectRatio()`, `aspectRatio()`.

This method enforces a specific output aspect ratio. The `aspect` argument may either be a number or a `X:Y` string.

Note that calls to `aspect()` are ignored when `size()` has been called with a fixed width and height or a percentage, and also when `size()` has not been called at all.

```
ffmpeg('/path/to/file.avi').size('640x?').aspect('4:3');  
ffmpeg('/path/to/file.avi').size('640x?').aspect(1.33333);
```

Adaptive Streaming 기능 구현 - Me



Adaptive Streaming 기능 구현 - Me



- 가장 디버깅하기 쉬운 코드
= 내가 직접 작성한 코드
- 가장 리팩토링하기 쉬운 코드
= 내가 직접 작성한 코드
- (내 눈에) 가장 깔끔한 코드
= 내가 직접 작성한 코드

```
67 function createSegment(inputPath, outputDir) {  
68   return new Promise((resolve, reject) => {  
69     try {  
70       // 플레이리스트가 저장될 경로  
71       const playlistPath = path.join(outputDir, "playlist.m3u8");  
72  
73       // 세그먼트 및 플레이리스트 생성  
74       ffmpeg(inputPath)  
75         .addOption("-threads ${os.cpus().length}")  
76         .addOption("-start_number", 0)  
77         // 세그먼트의 길이를 강제한 후 동영상 이어보기 시 치명적인 오류가 발생하여 이 옵션을 제거한다 2023-12-04 01:44  
78         // .addOption("-hls_time", segmentDuration) // 한 세그먼트의 길이  
79         // .addOption("-hls_flags split_by_time") // 동영상 시간을 기준으로 분할  
80         .addOption("-hls_list_size", 0) // 저장할 세그먼트 파일 최대 개수(무제한)  
81         .addOption("-f", "hls") // HLS 포맷  
82         .output(playlistPath)  
83         .on("end", () => {  
84           console.log(`${outputDir}에 HLS 세그먼트가 생성되었습니다.`);  
85  
86           resolve();  
87         })  
88         .on("error", (err) => {  
89           console.log(`HLS 세그먼트 생성 중 오류 발생: ${err}`);  
90  
91           reject(err);  
92         })  
93         .run();  
94     } catch (err) {  
95       console.error(err);  
96  
97       reject(err);  
98     }  
99   });  
100 }
```

Adaptive Streaming 기능 구현 - Summary

Life Lesson

요행을 바라지 말자



03

교후

1. Open Source Software



프로젝트 작업 환경도

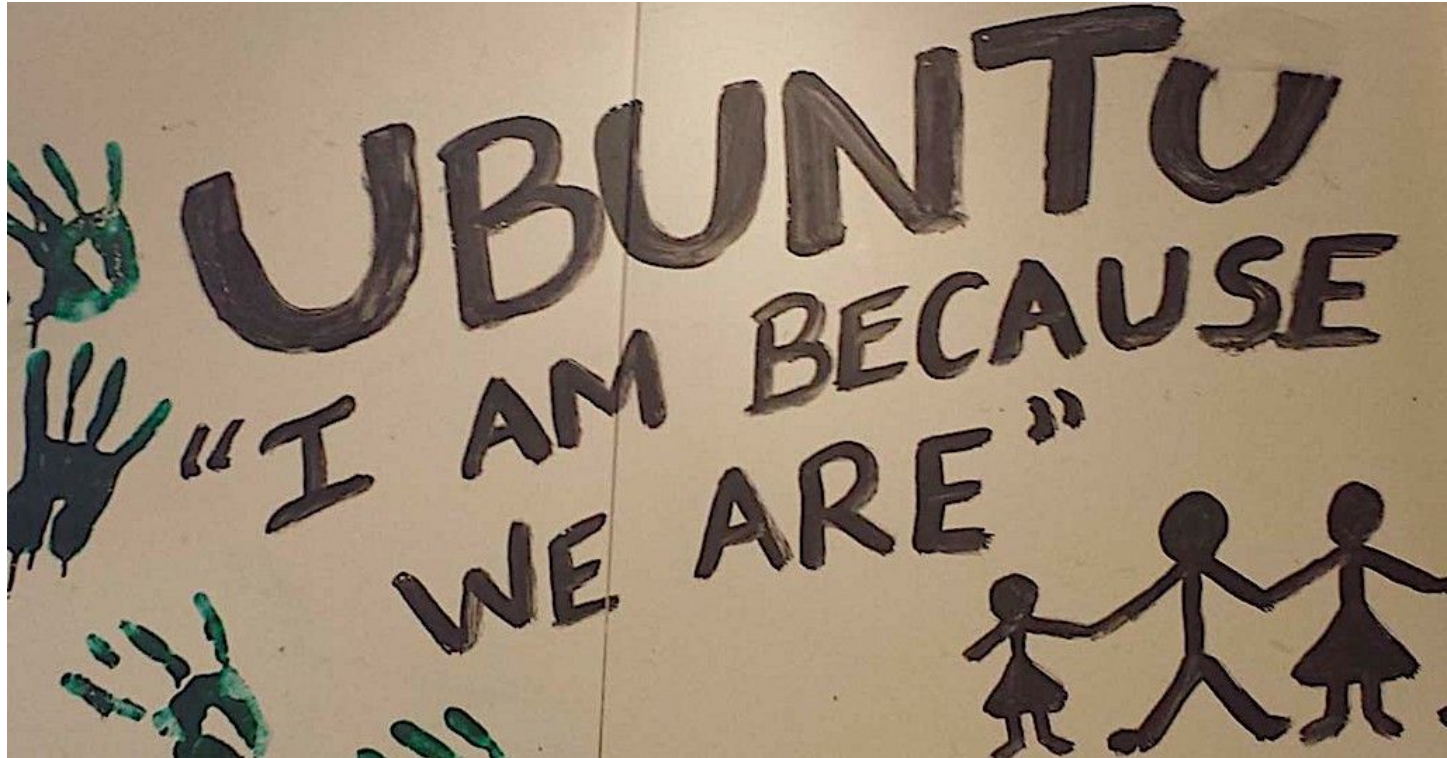


기능 구현의 핵심이 된 도구도



모두 오픈 소스 소프트웨어이다.

2. Teamwork



3. Life Lesson



04

프로젝트 시연

THANKS!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**