

# 클라우드 컴퓨팅

〈Azure를 이용한 스트리밍 서비스 제공〉

- 개발 결과 보고서 -

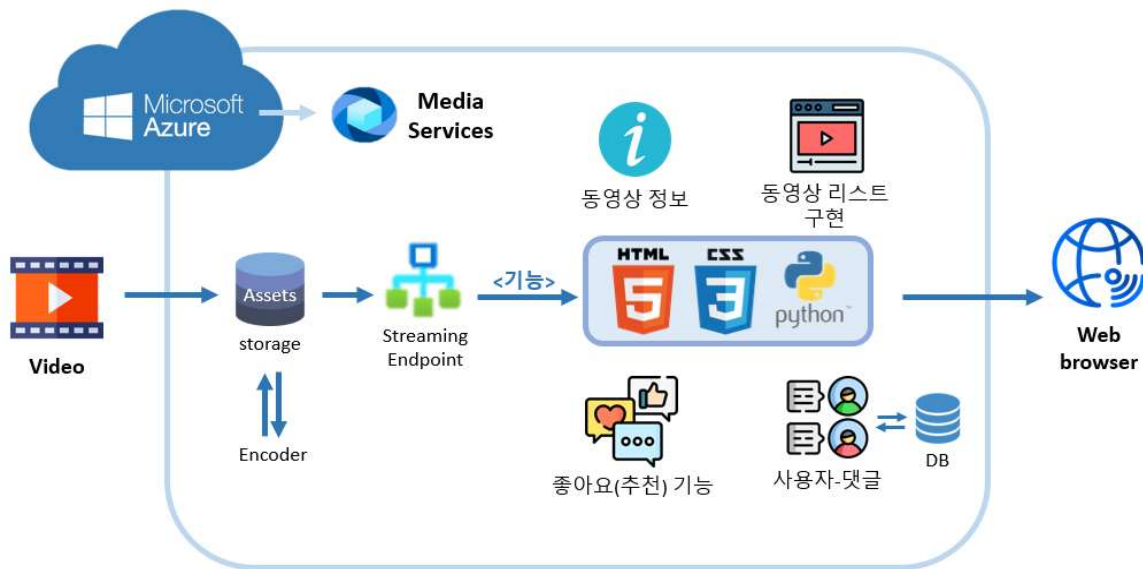
과목명	클라우드 컴퓨팅
담당	이대원 교수님
제출일	2022년 12월 21일
전공	컴퓨터공학과
학번	2020305039
이름	유하영

## 1. 서론

개발 계획서에서 작성한 내용을 바탕으로 Azure을 이용해 스트리밍 서비스를 제작하였다. 개발 계획서에 언급한 기능은 모두 제작하였으며 추가적인 기능은 없다.

## 2. 시스템 구조도

시스템 구조도는 개발 계획서에 작성한 내용과 동일하다.



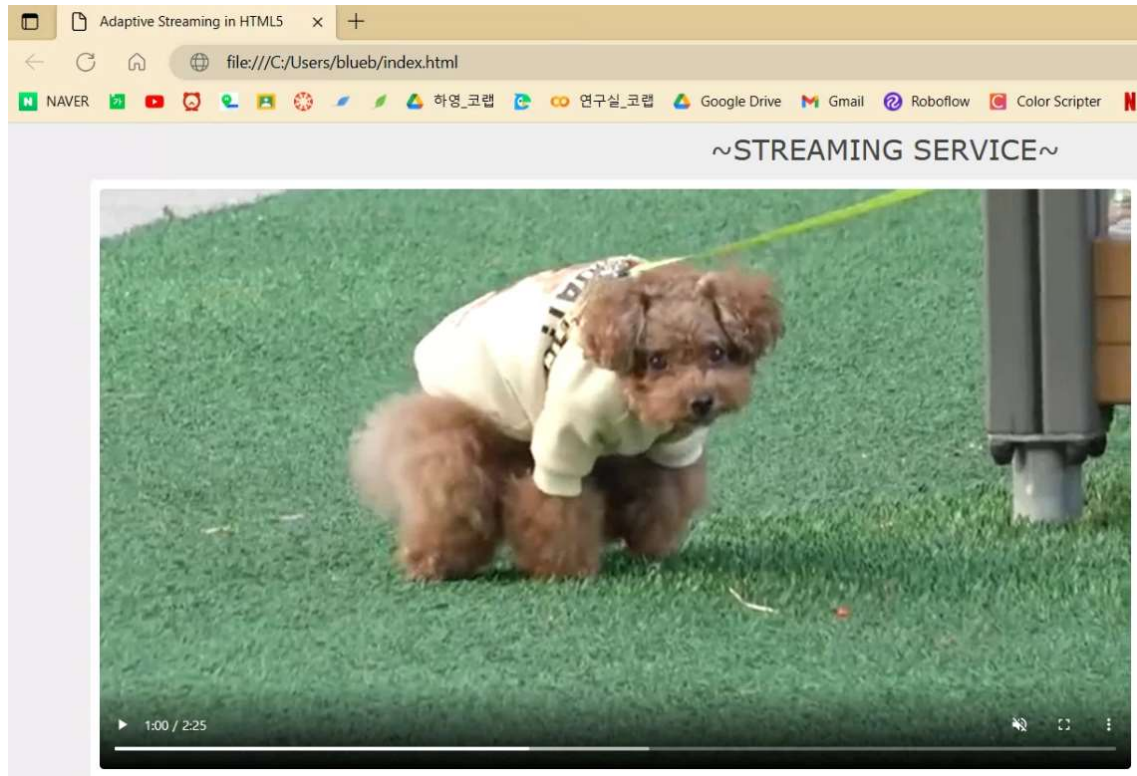
<그림 1. 시스템 구조도>

동영상은 Azure에서 제공하는 스트리밍 서비스 기능을 이용해 Azure asset에 업로드하고 서버에서 동작시킨다. 추가된 기능을 구현하기 위해서는 html과 javascript를 바탕으로 작성하였으며, 기능을 한눈에 알아보기위해서 css를 일부 사용하였다. 동영상 리스트를 나열하고 선택한 영상을 큰 화면에 띄우는 기능과 동영상 정보를 나타내는 기능, 기획서에서 추가로 동영상 정보 기능에 추가로 언급했던 해시태그 기능과 좋아요 기능을 구현하였다. 그리고 사용자 닉네임과 함께 댓글을 다는 기능도 Azure에서 제공하는 DB를 이용해 제작하였다.

### 3. 제공 기능 별 증빙 및 동작 스크린샷

개발한 기능은 계획서에서 언급했듯 필수적으로 개발해야 하는 스트리밍 서비스에 4가지의 추가기능을 더해 총 5개의 기능을 제작하였다.

#### 1) 스트리밍 서비스 (필수)

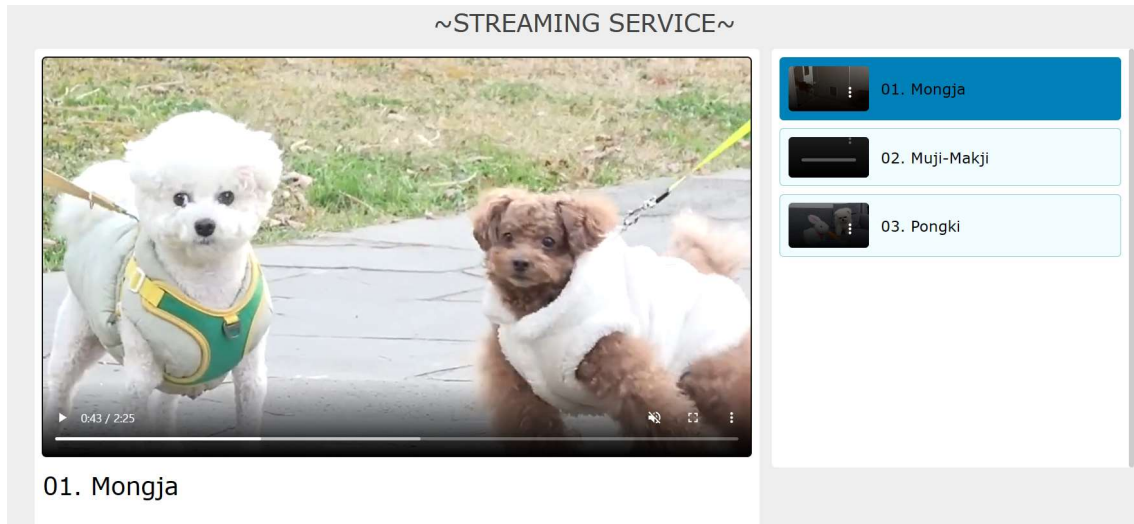


〈그림 2. 비디오 스트리밍 동작〉

웹 환경에서 영상을 동작하기 위해 Azure의 asset storage에 제공하고자 하는 영상을 업로드하고 streaming endpoint를 지정해 준다. html 상에서 DASH url을 연결하여 동작해주면, 위와 같이 웹 환경에서 비디오 스트리밍이 정상적으로 동작하는 것을 확인할 수 있다. 재생과 정지, 배속, 전체화면, 음량조절 및 음소거, 화면 속 화면 등 기본적인 기능은 제공되고 있다.

## 2) 동영상 리스트(목록) 표시

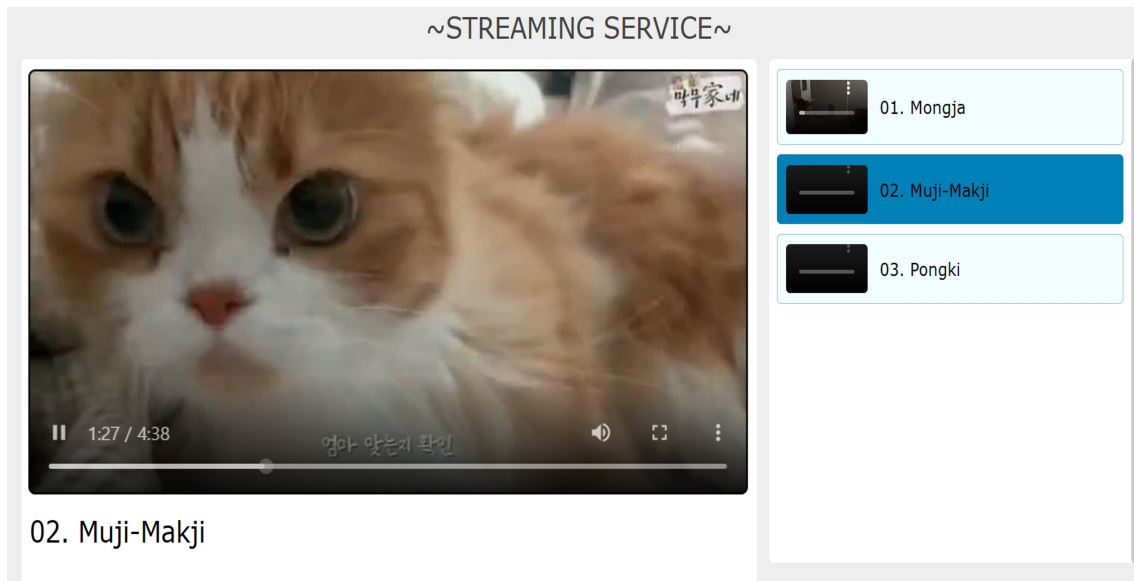
서비스에서 어떤 영상이 제공되는지 한눈에 확인하기 위해 동영상 목록을 표시하는 기능을 넣었다. 해당 서비스는 총 3개의 영상을 제공한다.



<그림 3. 동영상 목록 표시(예시 1)>

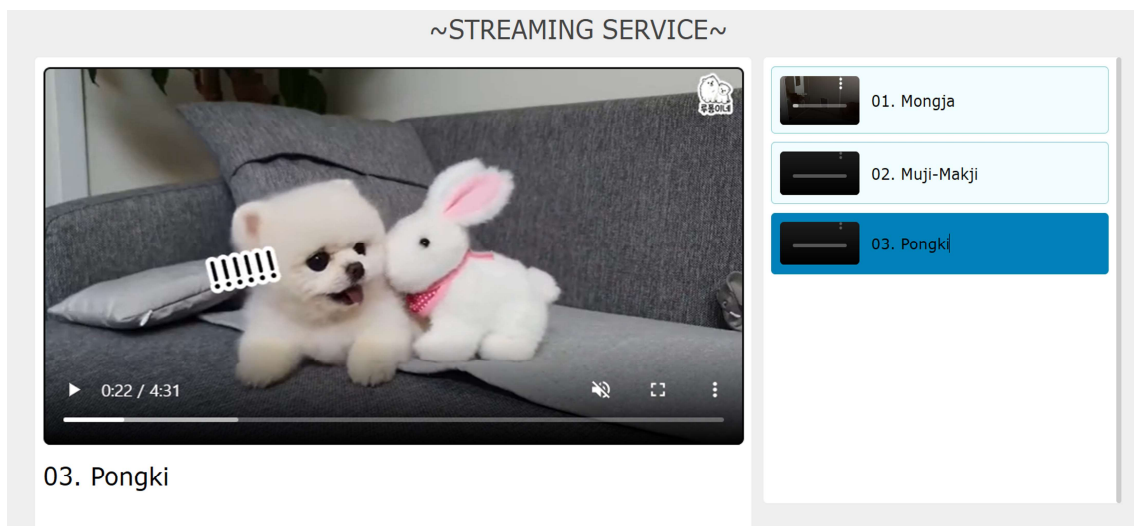
<그림 3. 동영상 목록 표시(예시 1)>의 화면은 초기화면이자, 현재 실행되고 있는 영상이다. 좌측에는 실행되고 있는 영상을 큰 화면으로 볼 수 있도록 200%로 확대하여 나타내었고 영상 하단에는 어떤 영상인지 알 수 있도록 제목을 크게 표시하였다. 우측에는 storage에 저장된, 즉, 제공되는 동영상의 리스트를 나타내며 총 3개의 영상을 제공한다. 현재 첫 번째 영상이 재생되고 있기에 리스트의 첫 부분에 음영처리가 되어있다.

만일, 리스트의 두 번째 영역을 클릭하게 되면 다음과 같은 화면이 나타난다.



<그림 4. 동영상 목록 표시(예시 2)>

마지막 영상도 마찬가지로 동일하게 동작한다.



<그림 5. 동영상 목록 표시(예시 3)>

이로써 서비스 사용자는 영상의 리스트를 직관적으로 확인할 수 있고, 보고 싶은 영상을 더 큰 화면으로 볼 수 있다.

해당 기능을 구현하기 위한 코드는 다음과 같다.

```

<div class="video-list">
  <div class="vid active">
    <video id="videoplayer1" controls muted auto play>
      <source src=url1
        type='video/mp4; codecs="avc1, mp4a"'>
    </video>
    <h3 class="title">01. Mongja</h3>
  </div>
</div>

```

(초기화면 videoplayer - 코드)

```

<div class="container">
  <div class="main-video">
    <div class="video">
      <video id="videoplayer1" controls muted auto play>
        <source src=url1
          type='video/mp4; codecs="avc1, mp4a"'>
      </video>
      <h3 class="title">01. Mongja</h3>
    </div>
  </div>
</div>

```

(동영상 리스트 클릭시 동작되는 videoplayer - 코드)

```

let listVideo = document.querySelectorAll('.video-list .vid');
let mainVideo = document.querySelector('.main-video video');
let title = document.querySelector('.main-video .title');

listVideo.forEach(video =>{
  video.onclick = () =>{
    listVideo.forEach(vid => vid.classList.remove('active'));
    video.classList.add('active');
    if(video.classList.contains('active')){
      let src = video.children[0].getAttribute('src');
      mainVideo.src = src;

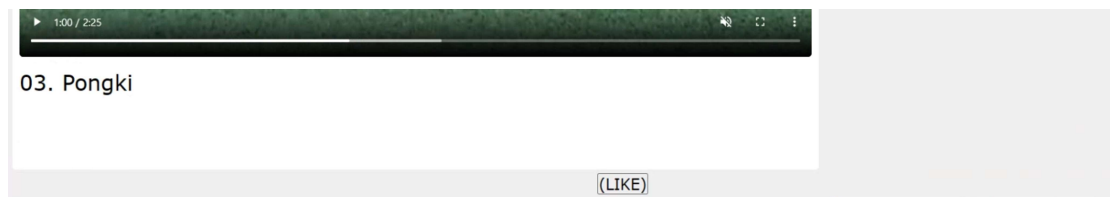
      let text = video.children[1].innerHTML;
      title.innerHTML = text;
    }
  }
});

```

(동영상 리스트 구현 - 코드)

### 3) 동영상 추천 기능

업로드된 동영상의 품질을 판별하기 위해 ‘좋아요’ 기능을 구현하였다. 이 기능으로 실제 동영상의 질을 판단한다던가 알고리즘과 연동이 되지는 않는다. 하지만 좋아요를 눌렀을 때 음영처리와 함께 팝업창이 뜨면서 좋아요가 눌렸음을 확인할 수 있는 단순 추천 버튼 기능을 만들어보았다.



실제 (추천)이 배치된 화면



(추천) 전 형태



(추천) 후 형태

<표 1. 동영상 추천 기능 동작>

영상 하단에는 “(LIKE)” 라는 동영상 추천 기능이 있다. 한 영상에는 한 사람당 하나의 추천만 가능하기 때문에, 이를 반영하여 버튼을 누르게 되면 위와 같이 음영처리 되고 더 이상 버튼을 누를 수 없다. 그리고 버튼을 누름과 동시에 <그림 6. (추천) 시 뜨는 팝업창>와 같은 팝업창이 뜨게 된다.

**이 페이지에는**

이 동영상을 좋아합니다.

확인

<그림 6. (추천) 시 뜨는 팝업창>



아래는 추천 기능을 구현하기 위한 코드이다.

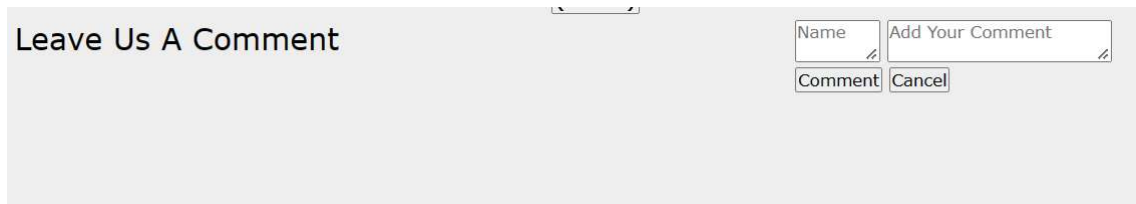
```
function click() {  
    const target = document.getElementById('target_btn');  
    alert("이 동영상을 좋아합니다.");  
    target.disabled = true;  
}
```

(추천 기능 버튼 - 코드)

#### 4) 사용자-댓글 기능

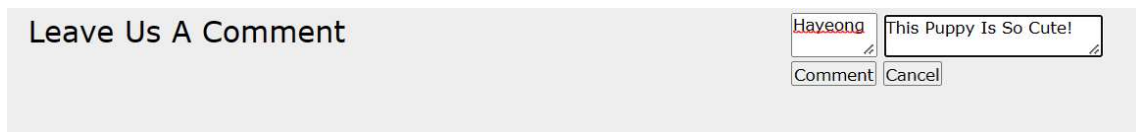
개발 계획서에 작성했듯, 사용자가 로그인하고 댓글을 다는 기능은 구현하기 어렵다고 판단했기에 이전의 경험을 되살려 사용자가 즉석에서 닉네임을 만들고 댓글을 다는 기능을 구현하였다.

아무런 댓글을 작성하지 않았을 때의 초기화면은 다음과 같다.



The image shows a web form titled "Leave Us A Comment". On the right side, there are two input fields: "Name" and "Add Your Comment". Below the "Name" field is a "Comment" button, and below the "Add Your Comment" field is a "Cancel" button. The form is currently empty, with no text entered in the input fields.

<그림 7. 사용자-댓글 기능(초기화면)>

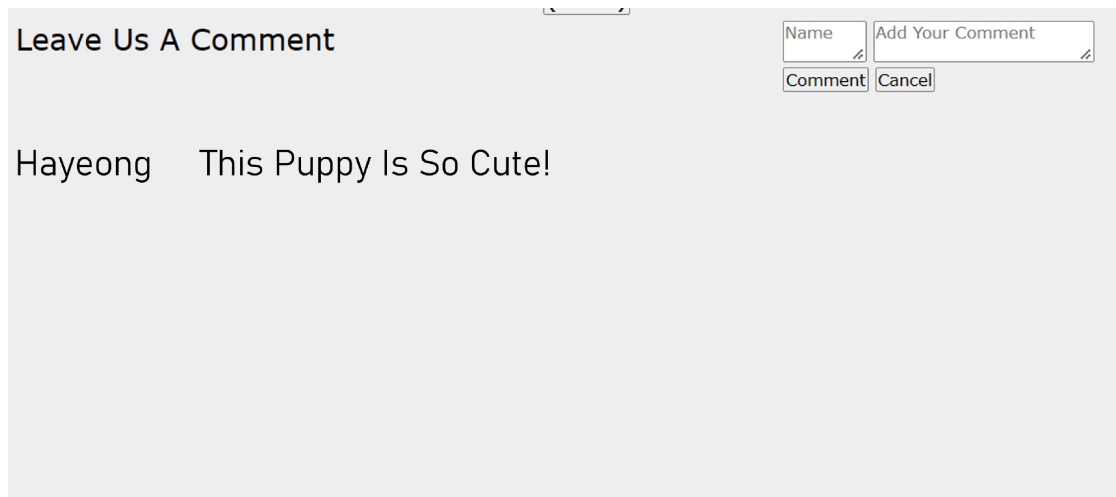


The image shows the same "Leave Us A Comment" form as in the previous image, but now it contains input. The "Name" field has the text "Haveong" entered, and the "Add Your Comment" field has the text "This Puppy Is So Cute!" entered. The "Comment" and "Cancel" buttons remain below the respective fields.

<그림 8. 사용자-댓글 기능(댓글 작성 화면)>

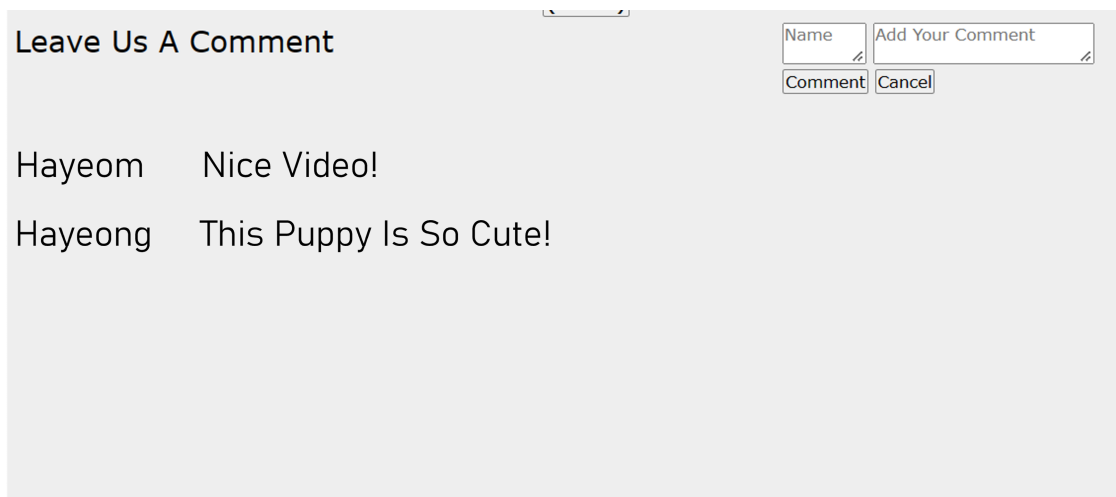


닉네임과 댓글을 모두 작성한 후 Comment 버튼을 누르면 <그림 9. 사용자-댓글 기능 (댓글작성 후 화면)>과 같이 닉네임과 함께 작성한 내용이 그대로 반영되어 출력된다.



The screenshot shows a web form titled "Leave Us A Comment". At the top right, there are two input fields: "Name" and "Add Your Comment", each with a small icon indicating a required field. Below these fields are two buttons: "Comment" and "Cancel". The main area of the form displays the submitted comment: "Hayeong This Puppy Is So Cute!".

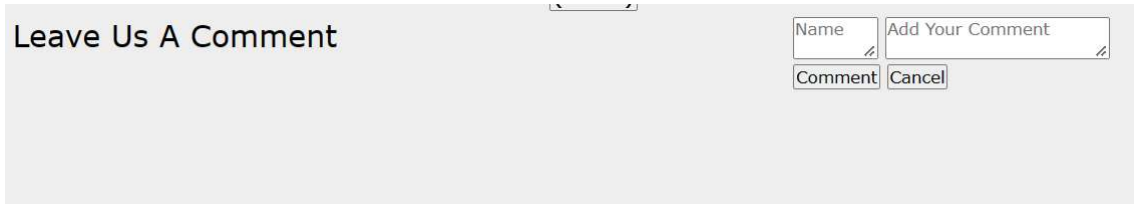
<그림 9. 사용자-댓글 기능(댓글작성 후 화면)>



The screenshot shows the same "Leave Us A Comment" form as in the previous image. The submitted comments are now listed in reverse chronological order: "Hayeom Nice Video!" at the top, followed by "Hayeong This Puppy Is So Cute!" at the bottom.

<그림 10. 사용자-댓글 기능(2번째 댓글작성 후 화면)>

만일, 또 다른 닉네임으로 다른 내용의 댓글을 달게 되면 <그림 10. 사용자-댓글 기능 (2번째 댓글작성 후 화면)>와 같이 역순으로 댓글이 달리게 된다.



<그림 11. 사용자-댓글 기능(cancel 화면)>

만약 댓글을 작성하다 초기화하고자 할 때에는 Cancel 버튼을 클릭하면 된다.

위의 사용자-댓글 기능은 Microsoft Azure에서 제공하는 Azure Cosmos DB를 사용해 구현하였다. DB와 Azure에서 제공하는 함수 앱을 연동하여 javascript로 사용자에게 대한 데이터베이스를 저장해주었다.

본문

```
1 {  
2   "name": "Hayeong",  
3   "comment": "dd"  
4 }
```

<그림 12. 사용자-댓글 기능(DB 연동 코드)>

다음과 같이 코드를 작성하여 웹 내에서 댓글을 달게 되면 DB와 연동되어, 사용자의 name과 comment 정보가 함께 DB에 저장되게 된다.

## 5) 동영상 정보

사용자는 시청하고 있는 영상의 정보를 알 권리가 있다. “1) 스트리밍 서비스”에서 현재 실행되고 있는 영상의 제목이 출력되므로 추가적으로 영상에 대한 날짜와 해시태그 기능을 삽입하였다.



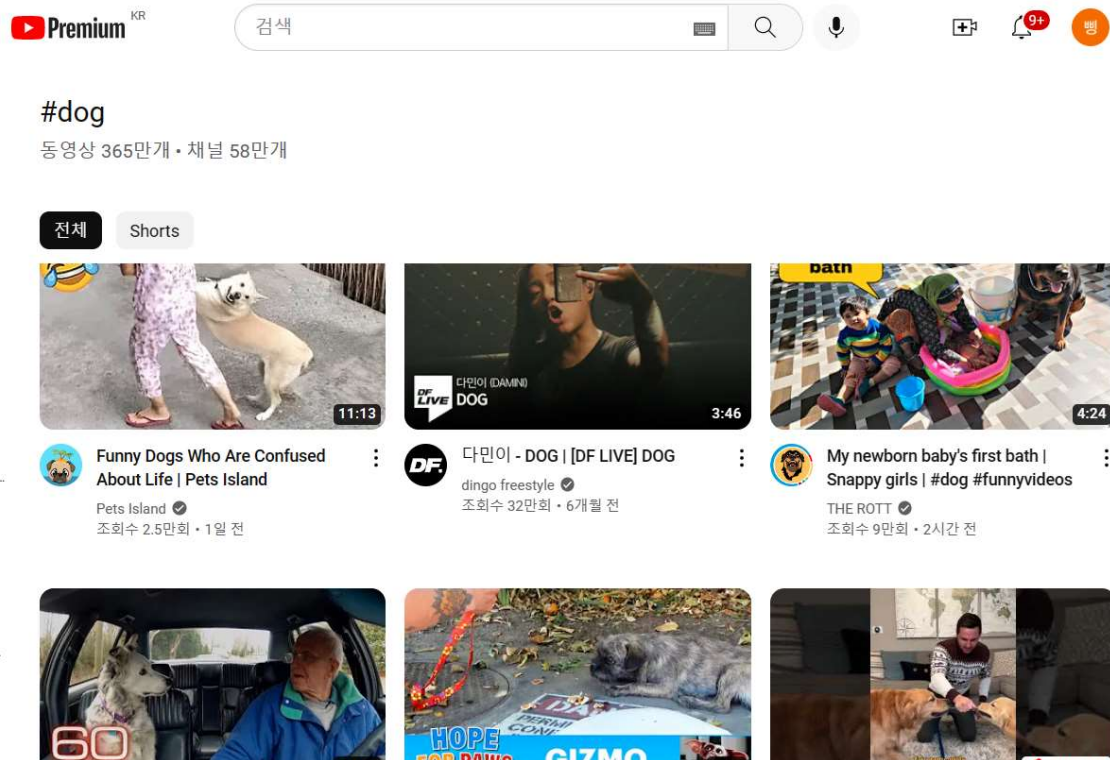
## 01. Mongja

2021.10.19

[#강아지](#) [#몽자](#) [#Dog](#)

<그림 13. 동영상 정보(예시 1)>

큰 글씨의 동영상 제목 하단에 해당 영상이 업로드된 날짜가 출력된다. 업로드 날짜는 임의로 설정해주었다. 날짜 아래에는 해시태그가 달려있다. 해시태그는 영상을 집약하는 정보이다. <그림 13. 동영상 정보>은 강아지 영상이므로 ‘#강아지’, ‘#Dog’가 달렸고, 위 영상의 강아지는 유명한 유튜버 강아지이기에 자신의 이름을 딴 ‘#몽자’ 라는 해시태그가 추가적으로 달렸다. 만약 ‘#Dog’를 클릭하게 되면 해당 해시태그와 연동된 사이트로 이동하게 된다.



〈그림 14. 동영상 정보(해시태그 사이트)〉

연동된 사이트는 유튜브 해시태그 기능을 이용하였다. 〈그림 14. 동영상 정보(해시태그 사이트)〉는 ‘#Dog’ 클릭 시 이동되는 화면이다.

## 02. Muji-Makji

2022.12.13

[#고양이](#) [#Cat](#) [#무지막지](#)

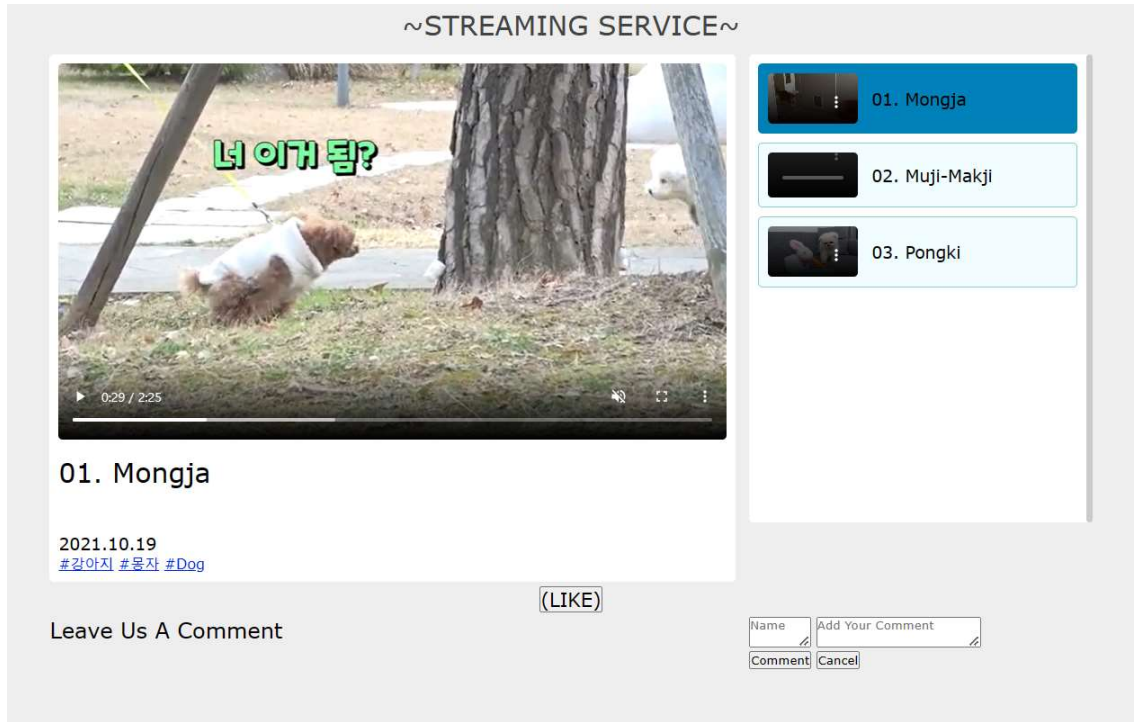
## 03. Pongki

2022.10.5

[#감아지](#) [#Dog](#) [#퐁키](#) [#Fun](#)

〈그림 15. 동영상 정보(예시 2)〉

다른 영상들도 〈그림 15. 동영상 정보(예시 2)〉와 같이 영상의 정보와 해시태그가 삽입되어있다.



〈그림 16. 최종 화면〉

최종적으로 출력되는 스트리밍 서비스 화면은 다음과 같다.

#### 4. 느낀 점

이번 프로젝트 활동을 통해 스트리밍 서비스가 어떤 구조로 동작하는지 알 수 있었고, 직접 구현해봄으로써 클라우드 서비스에 대해 더 잘 이해하게 되었다.

더불어 html과 javascript는 처음 사용해보는 언어라 많은 어려움이 있었고 다양한 basic 문서를 참고하게 되는 것이 불가피했다. 하지만 그동안 많은 언어를 접해본 경험을 바탕으로 개발 초기보다 능숙하게 사용할 수 있게 되었다.

개발 기간이 짧아 구현하고 싶었던 기능을 추가적으로 넣지 못하였고, 계획서에 작성한 대로 제한된 기능만 구현해야 한다는 아쉬움이 있었다. 하지만 직접 스트리밍 서비스를 구현해냈다는 뿌듯함을 동시에 느꼈다. 많은 우여곡절이 있었지만 특별한 경험을 얻어 가게 되는 프로젝트 활동이었다.