휴먼 컴퓨터 인터페이스

과제 #1. 기본 공학용 계산기

2018203062

김수빈

2019.04.13.

광운대학교 소프트웨어학부

개요

기능적 요구조건에 대한 구현 완성도요약

기능적요구조건			
기능	종류		구현여부
수식입력	정수, 실수, 복소수	+ - / * % ^ == != > < <= >= i	o
	벡터, 행렬	dot , cross	o
	상수	pi, e	o
	함수	sin, cos, tan exp, log, sqrt	o
결과지원	올바른입력	수식결과값	o
	잘못된 입력	오류메시지	o
변수, 함수 정의	함수	f , g	o
	변수	x, y, z	o
추가기능	각도계산	°(deg) 입력 가능	o
	추가 함수기능	arc, inv, h, abs	o
		!	o

인터페이스 요구조건에 대한 구현 완성도 요약1_기본요구조건

인터페이스 요구조건			
기능		설명	구현여부
팝업메뉴	기능그룹화 및 선택지의 최소화	컨트롤창의 dropdown 메뉴	o
	도움말	팝업형식의 도움말	o
도움말	인터페이스 사용법 설명	팝업창으로 간단한 설명서 작성	o
백 스페이스	입력 수식의 마지막 문자 지우기	'<'버튼	o
벡터, 행렬 입력 간소화	괄호 쌍 입력의 불편함 최소화	활성화 형식의 버튼 입력방식 전환, 행렬메뉴의 추가 입력방식으로 행렬입력을 간편화 함.	o
복사&붙여넣기	일부 영역 선택하여 보관 및 재사용	선택영역 복사/붙여넣기 가능	o
히스토리	과거 입출력 내역의 확인 및 재사용	복사하거나, 수식이 계산되거나, save기능으로 수식을 저장했을 때 위에 있는 히스토리영역에 수식이 하나씩 추가된다.	o

+

인터페이스 요구조건에 대한 구현 완성도 요약2_추가기능

추가	save 기능	계산을 하지 않더라도 기존 수식을 히스토리에 따로 저장할 수 있다.	o
	수식 이어쓰기	앞서 연산결과를 불러들여 그다음 계산에 사용할 수 있다. function일경우에는 이어쓰지 않고 바로 수식계산 가능	o
	백스페이스	문자일 경우 한번에 삭제된다.	o
	이중 operator 처리	연산자 두 개를 연속으로 입력할 경우 나중에 입력한 연산자로 대체 된다.	o
	(x) (x,y)	2변수, 3변수 등 함수의 입력을 빠르게 입력할 수 있도록 구현하였다.	o
	clear 기능	히스토리의 내용을 초기화 할 수 있다.	o
	스크롤	내용 over시 잘리지 않고 스크롤을 내려 확인할 수 있다. 히스토리도 가능하다	o
	cont	히스토리 불러오기 시에 새롭게 불러오거나 뒤에 붙이는 것을 조절하는 버튼	x

오픈소스 라이브러리 의존성 요약

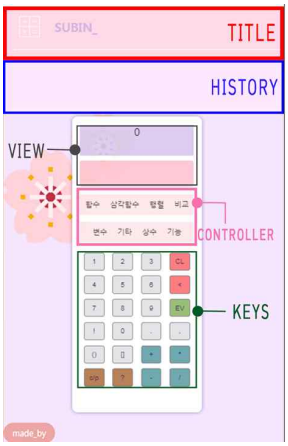
사용 오픈소스

```
<!--외부오픈소스라이브러리-->
<script src="./jquery-3.3.1.js"></script>
<script src='./math.js' type='text/javascript'></script>
```

- jquery.js
- math.js

본문

사용자 인터페이스의 구성 요소 및 사용 방법



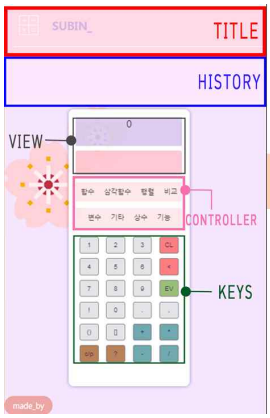
TITLE

HISTORY

VIEW

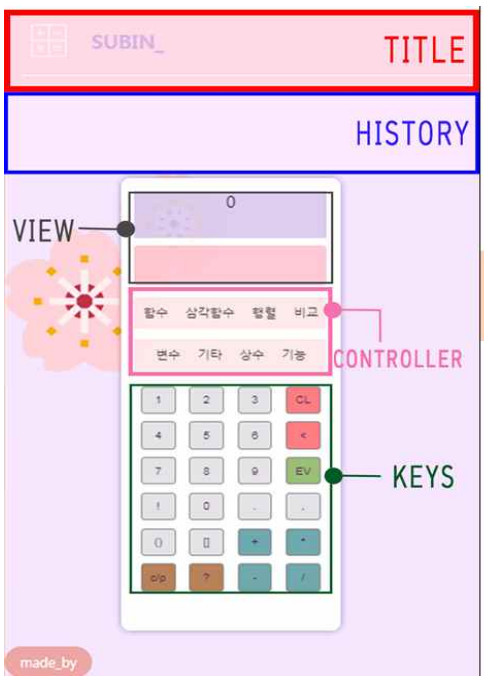
CONTROLLER

KEYS



TITLE

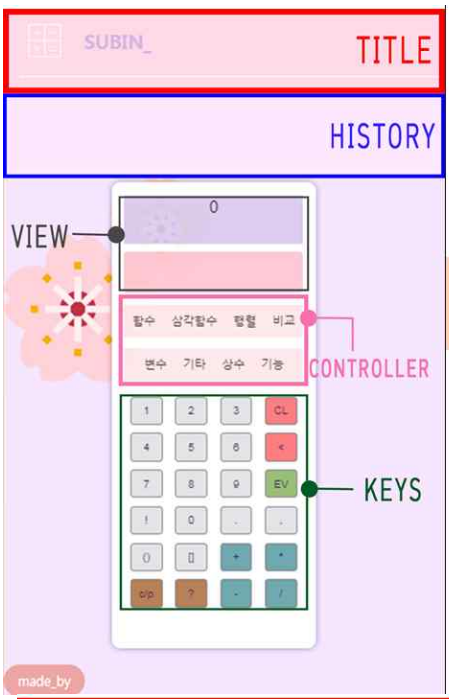
타이틀



HISTORY

히스토리

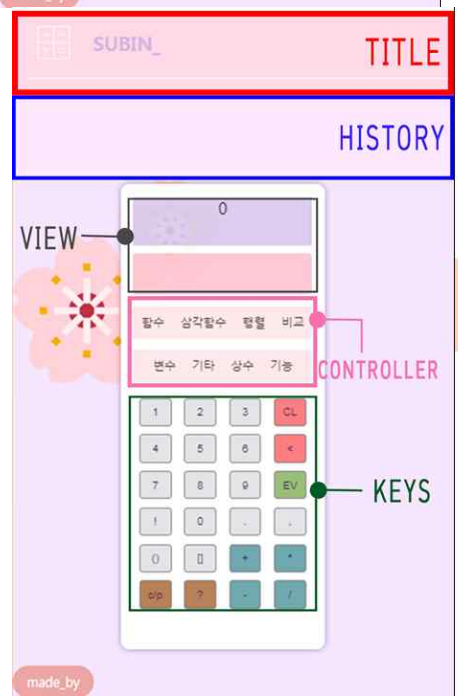
계산결과 및 SAVE data 를 저장한다.



VIEW

수식입력 및 결과창

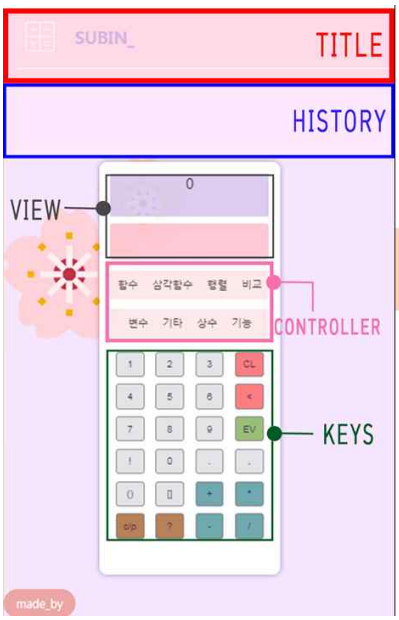
수식 입력과 출력을 보여주는 영역이다.



CONTROLLER

컨트롤러

여러가지 기능들을 지원하는 창이다



KEYS

키

일반적인 키들을 모아둔 창

CL 입력한 전체 수식 초기화

< Back Space : 한글자씩 지움

EV 계산완료

? 도움말 창 open

c/p 선택한 수식 copy & paste

일반 상태

일반 상태의 () 혹은 [] 버튼 을 누르면 활성화 상태가 된다.

활성화 상태에서 수식을 입력하면 괄호안에 작성이 된다.

수식 입력을 마치고 다시 누르면 일반상태로 돌아간다.

활성화 상태

특징적인 상호작용 방식들에 대한 세부 구현 방법

1) 벡터, 행렬 입력 간소화 _ 괄호의 활성화 및 전체영역 괄호

```
else if ($(this).text() == '[]') {
    id=5;
    $('#square').addClass('atv');
    origin = displayValue;
    par = "";
}
else if ($(this).text() == '()') {
    id=3;
    $('#round').addClass('atv');
    origin = displayValue;
    par = "";
}
```

```
else if ($(this).hasClass('atv')) {
    $(this).removeClass('atv');
    par = "";
}
```

- (좌)버튼 활성화 및 (우)해제 코드

활성화 :
[ ]또는 ( ) 글씨를 가진 버튼이 눌리면, [ ]또는 ( )에 atv 클래스를 추가해준 뒤,
현재까지 입력된 수식을 origin 변수에 넣는다. par에는 앞으로 입력될 수식이 들어간다.
해제 :
atv 클래스를 갖게 된 [ ]나 ( )가 눌리게 된다면 atv class를 없애준다.

```
else if ($('.par').hasClass('atv') === true) {
    if (origin[0] == '0') {
        origin = "";
    }
    par = par + $(this).text();
    displayValue = displayValue + $(this).text();
    console.log(id);
    if(id==3){
        displayValue = origin + '(' + par + ')';
    }
    else{
        displayValue = origin + '[' + par + ']';
    }
    $('#result').text(displayValue);
}
```

- 활성화 된 상태에서의 처리 코드

( ) 또는 [ ] 가 atv 클래스를 가지고 있는 동안에는 다음 코드가 실행이 된다.
par은 괄호안에 들어갈 수식들이 들어가는 문자열이다.
id변수는 ( ) 와 [ ]을 구분하기 위해 사용된 변수이다.

```
if ($(this).text() == 'EV') {
    try {
        $('.par').removeClass('atv');
        history.push(displayValue.toString());
```

- EV가 눌리면 자동으로 비활성화 된다

```
else if ($(this).text() == '[ ]') {
    displayValue = '[' + displayValue + ']';
    $('#result').text(displayValue);
}
else if ($(this).text() == '( )') {
    displayValue = '(' + displayValue + ')';
    $('#result').text(displayValue);
}
```

- 전체 영역 괄호

지금까지 입력한 수식 가장 외각에 괄호를 추가해준다.

2) operator 중복

```
else if ($(this).hasClass("operator") === true)//중복된 operator는 나중에 입력한 operator로 대체됨
{
    //입력되어 있는 마지막 값이 +-/* 이라면 다음을 실행
    switch (displayValue[displayValue.length - 1]) {
        case '+':
        case '-':
        case '/':
        case '*':
            let tmp = "";
            for (let i = 0; i < displayValue.length - 1; i++) {
                tmp += displayValue[i];
            }
            displayValue = tmp;
            displayValue += $(this).text();
            $('#result').text(displayValue);
            break;
        default:
            displayValue += $(this).text();
            $('#inputText').val(displayValue);
            $('#result').text(displayValue);
            break;
    }
}
```

- 이중 operator을 방지하기 위한 코드

연산자의 종류 + - / * 들이 이중으로 들어갈 경우, 나중에 입력된 operator로 대체된다. 새롭게 들어온 경우가 연산자인 경우, displayValue에 저장되어있던 변수를 임시 문자열에 저장한다 그리고 맨 마지막 문자를 새롭게 들어온 operator로 대체한다.

3) history

```
var history = [];//수식저장
var historyEval = [];//결과저장
```

- 수식과 결과값을 저장할 배열

```
if ($(this).text() == 'EV') {
    try {
        $('.par').removeClass('atv');
        history.push(displayValue.toString());
historyEval.push(displayValue);//히스토리에 계산결과 등록
```

- 배열에 저장하는 코드

EV키가 눌리면 history 배열에 현재 입력된 수식이 저장된다. 계산이 완료된 후의 결과값은 historyEval 배열에 따로 저장이 된다.

```
$('.history').append($('<button/>', {
    class: 'hk',
    text: history[history.length - 1] + ' = ' + historyEval[historyEval.length - 1]
}));
```

- 히스토리 영역에 표시

히스토리 영역에 "수식 = 결과값" 형태의 내용을 가진 버튼이 추가된다.

```
else if ($(this).text()=='clear'){
    $('.history').empty();
}
```

- clear

히스토리 클래스에 있는 모든 내용을 지운다.

```
else if ($(this).text() == 'save') {
    $('.history').append($('<button/>', {
        type: 'Button',
        class: 'key history_key',
        text: $('#result').text()
    }));
}
```

- save

save가 눌린다면 EV가 눌림과 관계없이 히스토리에 수식값을 추가할 수 있다.

```
$('.history').click(function(event){
    let tmp = $(event.target).text();
    // console.log(tmp);
    console.log($(this).hasClass('cont'));//-1
    if($('#latex').hasClass('cont')==(-1)){//cont를 가지고 있지않다면
        // alert("비워진다");
        displayValue = "";//비우고 시작
    }
    displayValue = "";

    if(tmp[0]=='f'||tmp[0]=='g')//함수라면 두번째 = 까지 복사
    {
        for (let i = 0; i < n_indexOf(tmp,"=",2)-1; i++) {
            // console.log(i);
            displayValue = displayValue + tmp[i];//카피한다.
        }
    }
    else if (tmp.indexOf("=")+1){//없으면 -1 있으면 1

        for (let i = 0; i < tmp.indexOf("=") - 1; i++) {
            // console.log(i);
            displayValue = displayValue + tmp[i];//카피한다.
        }
    }
    else{
        displayValue = displayValue + tmp;
    }
    $('#result').text(displayValue);
    return false;
})
```

- 히스토리 불러오기

히스토리의 내용이 눌리면 =가 나오기 전까지 즉 결과값을 제외한 수식값이 displayValue 에 저장되어 화면에 표시된다. 함수의 경우에도 =이 쓰이기 때문에, 2번째로 나오는 = 이전까지의 수식을 불러온다.

4) copy&paste

```
else if ($(this).text() == 'c/p') {
    if(document.getSelection())
    {
        let txt = selectText();
        if (window.getSelection) {
            txt = window.getSelection();
            if (txt.toString() != ""){
                copy = txt.toString();
                $('.history').append($('<button/>', {
                    type: 'Button',
                    class: 'key history_key',
                    text: txt
                }));
            }
            else
            {
                displayValue = displayValue + copy;
                $('#result').text(displayValue);
            }
        }
    }
}
```

-복사 붙여넣기 코드

copy :
만약 선택된 부분이 있다면 그 부분을 copy변수에 넣고 히스토리에 복사된 수식을 넣어준다.
paste :
선택된 부분이 없는 상태로 c/p 버튼을 누른다면, 붙여넣기로 간주하고
copy되어있었던 문장을 displayValue에 붙여서 result에 출력해준다.

5) 한글자씩 지우기

```
else if ($(this).text() == '<') {
    var tmp = "";

    switch (displayValue[displayValue.length - 1]) {//마지막 글자가
        case 'n':
        case 's':
            for (let i = 0; i < displayValue.length - 3; i++) {
                tmp += displayValue[i];
            }
            break;//tan,sin && cos 일 경우 3글자를 지움.

        case 'h':
            for (let i = 0; i < displayValue.length - 4; i++) {
                tmp += displayValue[i];
            }
            break;//h가 붙은 경우. 4글자를 지움.
        default:
            for (let i = 0; i < displayValue.length - 1; i++) {
                tmp += displayValue[i];
            }
            break;//나머지는 그냥 하나 지움.

    }
    displayValue = tmp;
    $('#inputText').val(displayValue);
    $('#result').text(displayValue);
    //이걸 누르면 글자가 하나가 사라짐
}
```

- 벡스페이스 버튼 코드

현재 입력된 글자의 마지막을 인식해서, 문자일 경우 그에 해당하는 글자수를 지워주고, 나머지는 default는 한글자씩 지워지는 코드이다.

6) 문자 변환

```
if(displayValue.indexOf("ln") + 1) {//없으면 -1이출력됨
    // console.log(displayValue.indexOf("ln"));
    exchange = displayValue.replace('ln', 'log');
    // console.log(exchange);
}
if (displayValue.indexOf("√") + 1) {
    exchange = displayValue.replace('√', 'sqrt');
    // console.log(exchange);
}
if (displayValue.indexOf("°") + 1) {
    exchange = displayValue.replace('°', 'deg')
    // console.log(exchange);
}
```

- 문자열을 변환해준다.

기존 sqrt 등 영어로 작성되어있던 수식들을 우리가 기존에 잘 알고 있는 기호들로 바꿔준 뒤 내부에서 계산시에는 기존 문자열로 다시 바꾸어주는 작업을 해주었다.

7) 팝업 / 도움말

```
else if ($(this).text() == '?') {
    openHelpModal();
}
```

- 팝업을 지원하는 ' ? ' 버튼을 누르면 도움말 팝업창이 열린다.

```
function openHelpModal() {
    $('.helpModal').removeClass('hidden');
    // $(".function_key").bind("click",keyhandling);
}
function closeHelpModal() {
    $('.helpModal').addClass('hidden');
}
$('.closeHelpModal').click(closeHelpModal);
$('.modal_overlay').click(closeHelpModal);
//function 버튼 클릭시 모달창 on
```

- modal의 open & close

hidden 클래스의 여부로 팝업창을 여닫는다. 닫기버튼이나, 오버레이된 부분을 누르면 팝업창이 닫힌다.

8) 드롭다운메뉴

```
.menubar li ul{
background: rgb(109,109,109);
display:none; /* 평상시에는 드랍메뉴가 안보이게 하기 */
height:auto;
padding:0px;
margin:0px;
border:0px;
position:absolute;
width:200px;
z-index:200;
/*top:1em;
/*left:0;*/
}

.menubar li:hover ul{
display:block; /* 마우스 커서 올리면 드랍메뉴 보이게 하기 */
}
```

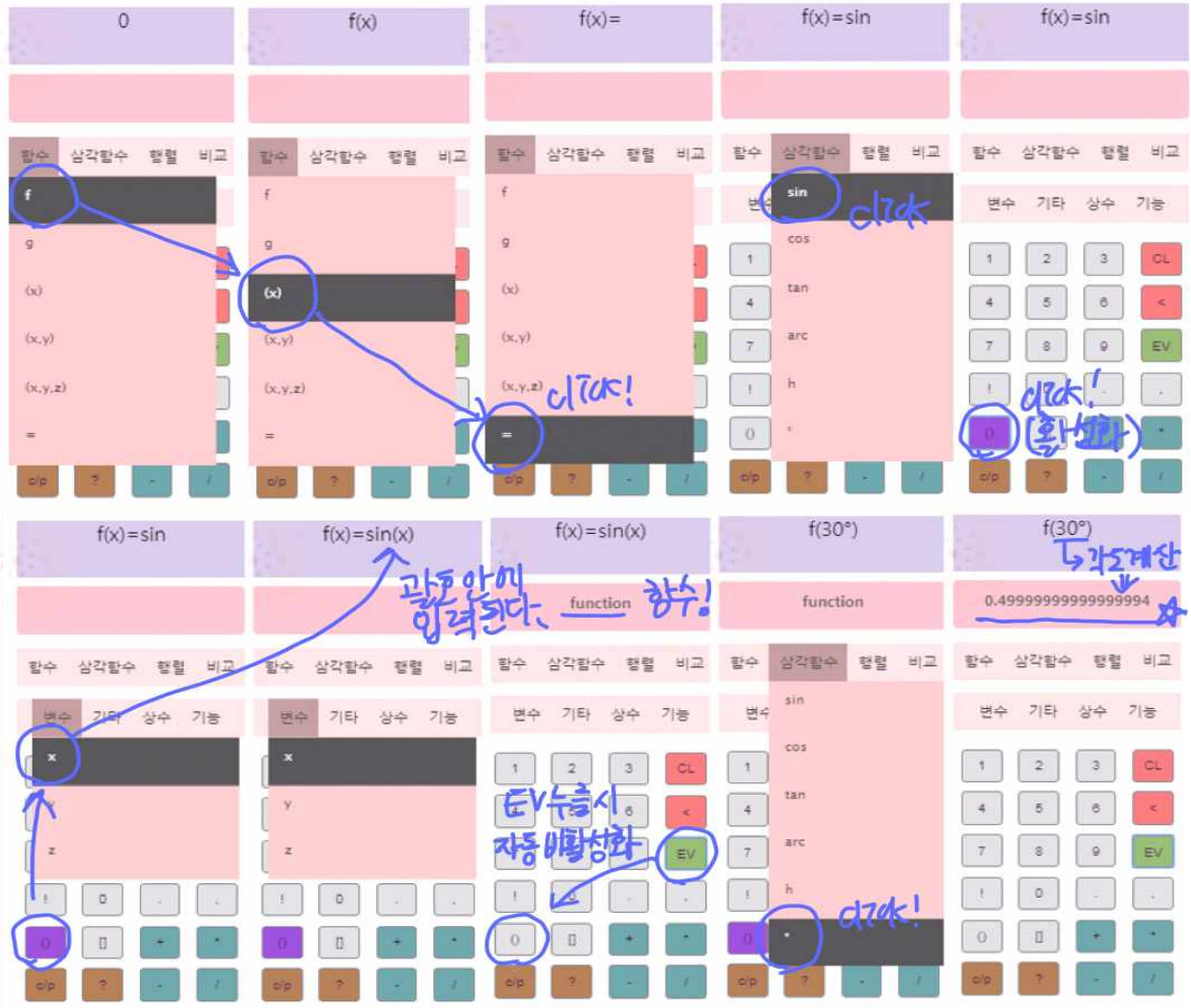
- dropdown 메뉴

ul의 display를 none으로 설정해놓은 후 마우스 커서를 올리면 드랍 메뉴가 보이도록 설정해 주었다.

실제 문제에 대한 사용 예시

1) cross([1,2,4],[2,4,5])=[-6,3,0]

2) sin(30°) = 1/2



3) det([[0,0,1,0], [0,1,0,0], [0,0,0,1], [1,0,0,0]]) = 1

논의

구현 측면에서 성공적인 부분과 실패한 부분

성공

이중 operator 방지 기능과 copy & paste 기능이 가장 성공적으로 구현된 기능인 것 같다.
괄호버튼의 활성화와 비활성화가 성공적으로 되었다.

실패

실패한 부분은 시간이 부족하여 탑재하고 싶었던 cont (히스토리 이어붙이기 기능)
구현을 완성하지 못했다.

사용성 측면에서 긍정적인 측면과 부정적인 측면

긍정

드롭다운 메뉴의 경우 매번 메뉴를 볼때마다 클릭을 해야 하는 불편을 덜어주어서 사용할 때 클릭의 과정을 많이 단축시켜준다.

부정

( )의 기능이 기존 방식과 달라서, 도움말을 보지 않으면 사용할 때 어려움을 겪을 수 있다는 점이 가장 큰 단점인 것 같다.

과제 결과에 대한 전반적인 자체 평가 및 향후 개선 계획

1

이벤트에 대한 이해가 정확히 되지 않은 상태로 과제가 마무리 되었는데,
이것으로 인해 왜 안 되는지 모르겠는 코드들이 많이 존재했다.

특히 구현하려던 cont기능을 만들지 못한 이유도 여기에 있는 것 같다.

제대로 DOM의 구조와 함수들의 기능, 그리고 자바스크립트의 문법들에 대한 개념을
좀 더 공부해야 할 것 같다.

2

레이아웃을 짜는 것 또한 시간이 무척 많이 걸렸다.

좀 더 시각적으로 깔끔하고 이쁜 디자인으로 제작하고 싶었지만 원하는 대로 잘 되지 않아서 결국 기본적인 계산기 모양으로 가게 되었다.

향후에 개선한다면, 레이아웃 부분에서 더 이쁘게 만들어 보고 싶다.

클릭 가능한 YouTube 링크

https://www.youtube.com/watch?v=vbdfoTWVNO4&feature=youtu.be