

고급 웹프로그래밍 과제 보고서

이름	송영주
학번	2015114955

INDEX

- 1. 프로젝트 주제 ----- p.3
- 2. 프로젝트 구조 ----- p.4
 - 1) 프로젝트 전체 구조 설명 ----- p.4
 - 2) 웹브라우저 세부 설명 ----- p.5
 - 3) 웹서버, DB 세부 설명 ----- p.10
- 3. 실행 순서 ----- p.11
- 4. 참조 ----- p.13

1. 프로젝트 주제

RUN TRAX

Chrome의 공룡게임을 확장하여 공룡게임을 즐기고 유저들과 랭킹을 겨룰수 있는 웹 페이지를 제작했습니다.

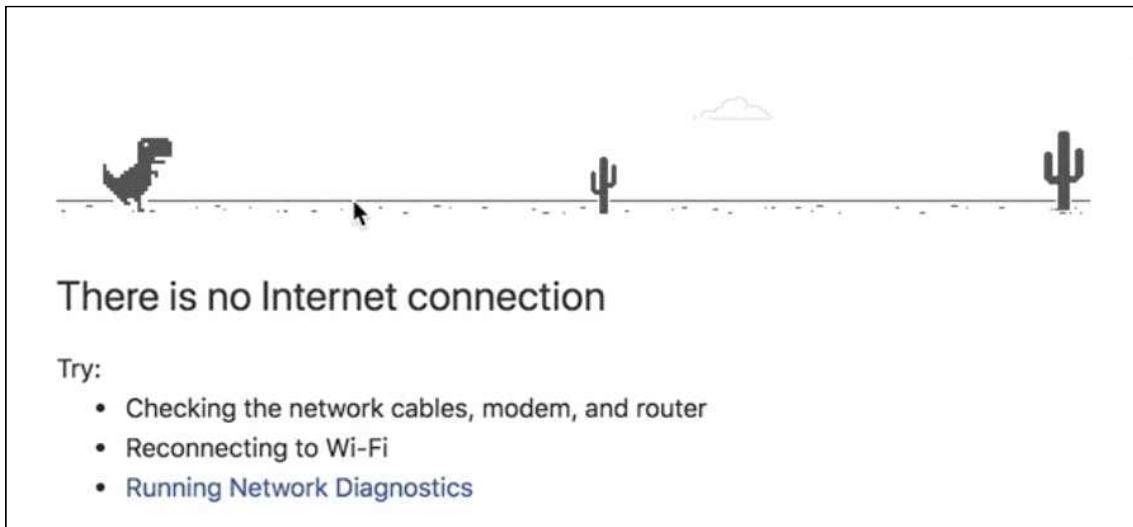


그림 1 Chrome 공룡 게임 화면

- USER의 이름을 입력받으면 게임이 시작됩니다.
- SPACE를 누르면 공룡이 점프하여 장애물을 피합니다.
- 공룡이 장애물을 피하는 시간에 따라 SCORE가 결정됩니다.
- 게임이 종료되면 상위 랭커 10명을 표출합니다.

2. 프로젝트 구조

1) 프로젝트 전체 구조

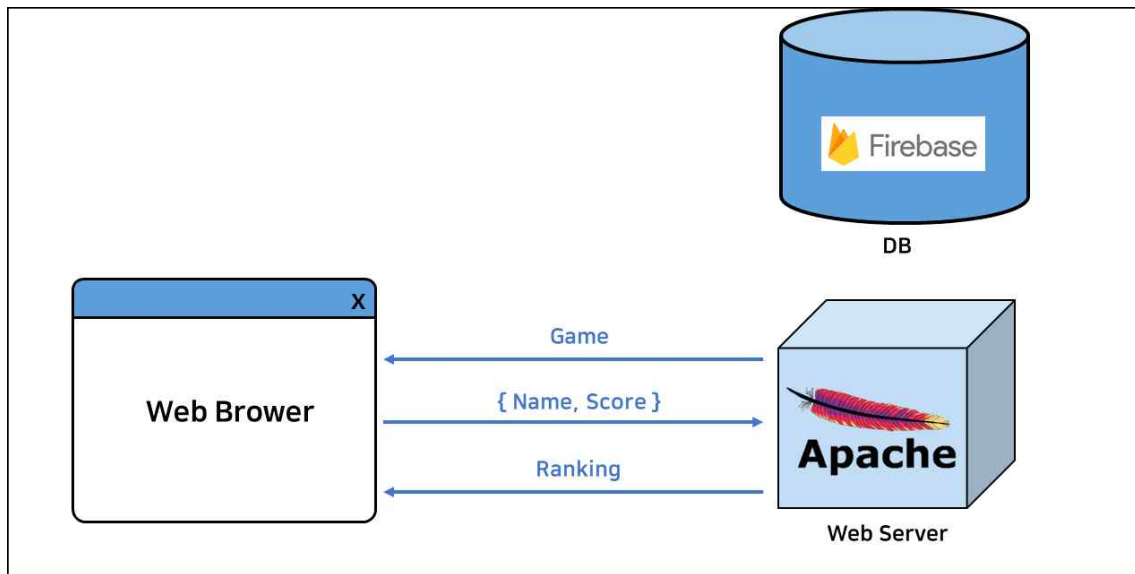


그림 2 프로젝트 구조도

- 웹 브라우저에서 NAME을 입력받습니다.
- 웹 브라우저에서 게임을 실행하여 SCORE을 얻습니다.
- Apache를 이용해 웹 서버를 구동합니다.
- 웹 서버에서 GAME을 실행하도록 HTML을 전송합니다.
- 웹 서버에서 {NAME, SCORE}를 입력받으면 DB에 저장합니다.
- Firebase를 이용해 실시간 DB를 구현했습니다.
- 웹 서버는 DB에서 상위 10명의 {NAME, SCORE}를 가져와 웹 브라우저에 표출합니다.

2) 웹브라우저 세부 설명

이 프로젝트의 웹 브라우저는 로그인, 게임 실행화면, 랭킹 표출화면으로 나뉜다. 각 화면은 아래 그림3과 같은 파일을 가집니다.



그림 3 웹 브라우저 파일 설명

1. 로그인

로그인 실행 화면은 아래 그림 4와 같습니다.

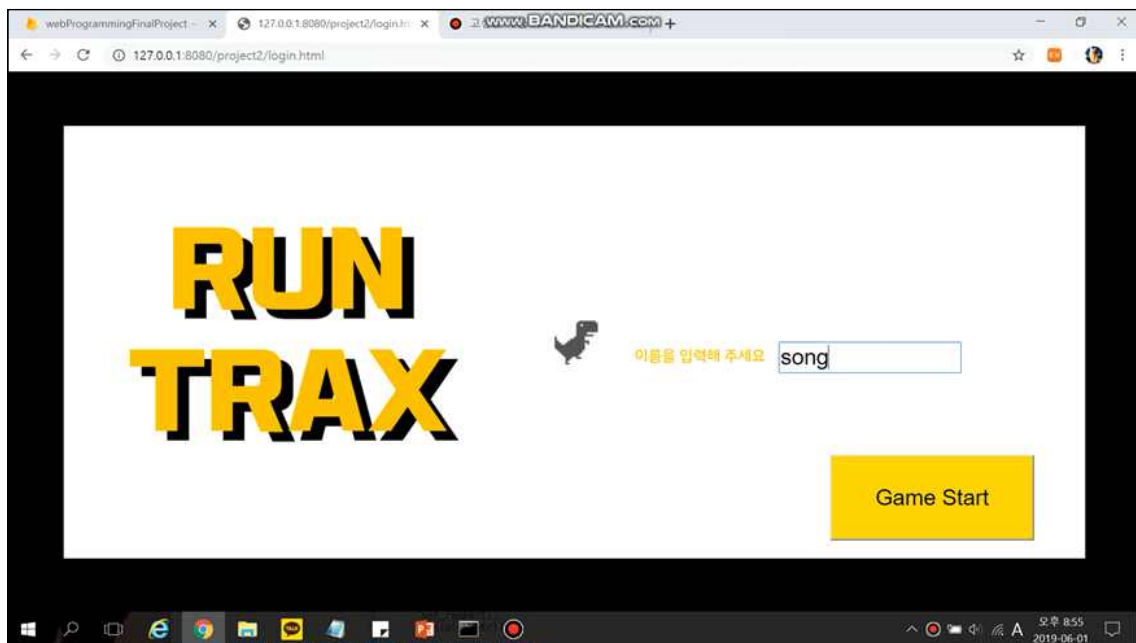


그림 4 로그인 실행화면

화면이 실행되면 가운데의 공룡이 달리는 모션이 실행됩니다. 공룡 이미지 두 개를 번갈아가며 표출하여 달리는 모션을 구현하였습니다. 모션 코드는 아래 그림5와 같습니다. setTimeout 함수를 이용하여 시간차를 두고 이미지를 번갈아가며 표출합니다. load함수는 브라우저가 처음 실행될 때 실행되는 함수입니다.

```
var state=0;
var trax;
function move(){
    state=(state+1)%2;
    trax.src="src/trax"+state+".png";
    setTimeout(move, 100);
}
function load(){
    trax=document.getElementById('trax');
    setTimeout(move, 100);
}
```

그림 5 공룡 이동 모션 코드

오른쪽 input태그에서 Name을 입력받습니다. Game Start 버튼을 누르면 게임 실행 화면 html을 실행하게 됩니다. 버튼의 onclick에 연결된 함수는 아래 그림 6과 같습니다. 다음 화면을 실행하면서 url로 name을 넘겨줍니다. name을 입력하지 않으면 다음화면으로 넘어갈 수 없게 했습니다.

```
function onclick(){
    var name;
    name=document.getElementById('name').value;
    if(name==""){
        alert('이름을 입력해주세요');
        return;
    }
    console.log('click :'+name);
    window.location.href="game.html?name="+name;
}
```

그림 6 Game Start버튼의 onclick 함수 코드

2. 게임 실행

게임 실행 화면은 아래 그림7과 같습니다.

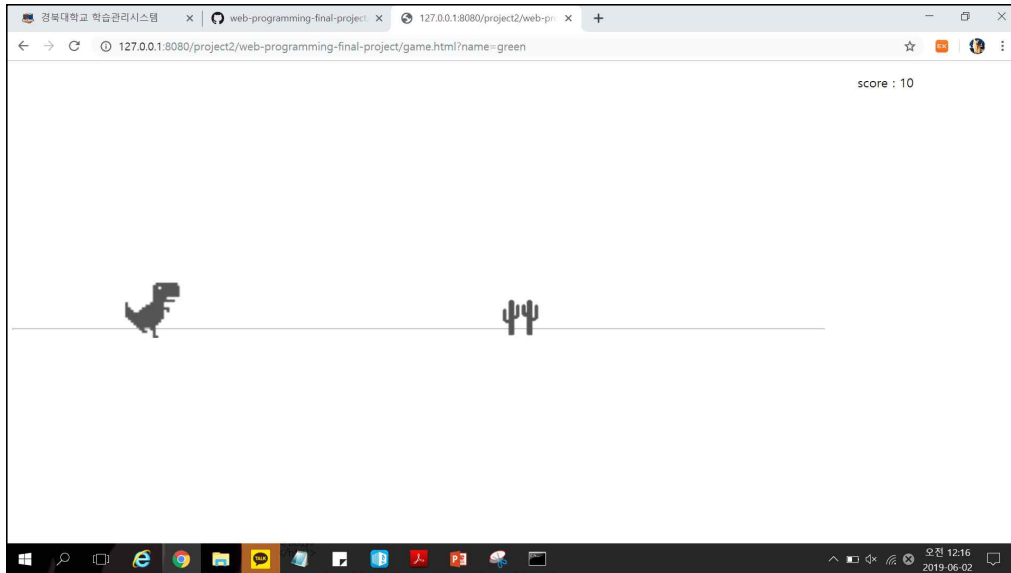


그림 7 게임 실행 화면

공룡이 다가오는 장애물을 피하는 게임입니다. 공룡이 장애물에 닿으면 게임이 종료됩니다. score은 게임 실행 시간에 비례하여 증가합니다.

공룡은 space를 누르면 점프합니다. 점프 모션을 통해 장애물을 피할 수 있습니다. 키보드 input은 아래 그림8과 같이 구현했습니다. jump 함수는 아래 그림9와 같이 구현했습니다. 점프 모션은 점프 한계까지 올라갈 때와 한계에서 바닥으로 내려올 때를 나누어 구현했습니다. 공룡의 위치를 absolute로 지정하고 점프 모션시 적당한 위치로 움직이도록 하였습니다.

```
window.onkeydown = function(e) {  
    console.log("keyboard input: "+ e.keyCode);  
  
    switch(e.keyCode) {  
        case 32:  
            if(!jumpStart) {  
                jumpStart=true;  
                jump();  
            }  
            break;  
    }  
}
```

그림 8 키보드 input 처리 함수

```
function jump(){  
    if(!gameOver&& jumpStart && !jumpTop){ // cur up  
        y-=speed;  
        if (y<=jumpBound) jumpTop=true;  
    }  
    else if(!gameOver&& jumpStart && jumpTop) { // cur down  
        y+=speed;  
        if (y>=ground) {jumpStart=false; jumpTop=false;}  
    }  
    me.style.top=y+"px";  
    if(jumpStart) setTimeout(jump, times);  
}
```

그림 9 jump 함수

공룡이 장애물과 부딪치면 게임은 종료되고 그림 10과 같은 화면이 표시되게 됩니다. visibility를 hidden으로 지정해놓았던 Game Over 이미지가 visible로 지정되면서 표시되게 됩니다. 또한 게임 종료와 동시에 공룡의 이동도 멈추게 됩니다. 만약 공룡이 점프중이었다면 더 이상 점프 모션을 하지 않고 Chrome 공룡 게임과 마찬가지로 공중에 정지하게 됩니다.

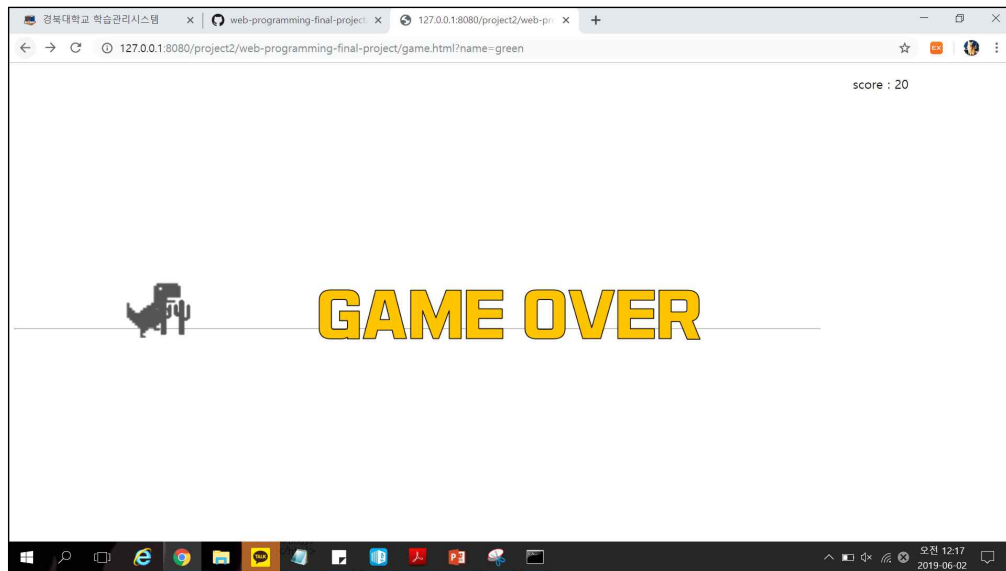


그림 10 게임 오버 화면

게임이 종료되고 5초 뒤 자동으로 랭킹 표출 화면으로 넘어가게 되며 name과 score를 파라미터로 전달합니다. 아래 그림 11은 게임 종료시 모션에 대한 코드입니다. check 함수는 게임이 종료되면 true를 리턴하여 if문이 실행됩니다. Game Over 이미지를 visible로 표시하고 setTimeout함수를 이용해 5초뒤에 랭킹 표시 화면을 실행시킵니다. load_rank함수에서 name과 score를 파라미터로 넘기는 것을 확인할 수 있습니다.

```
function load_rank(){
    window.location.href="rank.html?name="+name+"&score="+score;
}
function move(){
    if(!jumpStart){
        state=(state+1)%2;
        me.src="src/trax"+state+".png";
    }
    move_obs();

    if(check()) {
        document.getElementById('gameover').style.visibility="visible";
        setTimeout(load_rank, 5000);
        return;
    }
    score++;
    sTxt.innerHTML="score : "+score;
    setTimeout(move, times);
}
```

그림 11 화면 전환 코드

3. 랭킹 표출

랭킹 표출 화면은 아래 그림 12와 같습니다. 1위부터 10위까지 플레이어의 순위, 이름, 점수가 표출됩니다. 1, 2, 3위는 아래와 같은 색으로 표시되며 나머지는 검은색으로 표시됩니다. 만약 플레이어가 10위 안에 랭킹 되었을 시 그림 13과 같이 등수와 함께 축하 메시지가 함께 표출됩니다. 축하 메시지는 hidden으로 설정되어 있으며 순위권에 랭킹되었을 때만 visible로 처리됩니다.

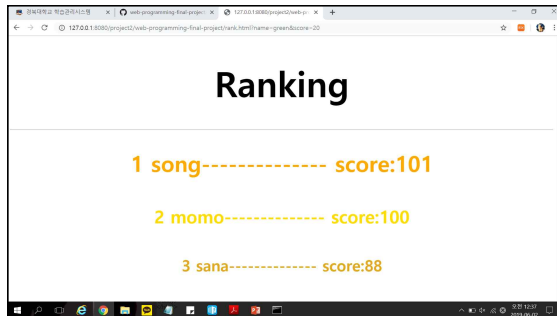


그림 12 랭킹 표시 화면

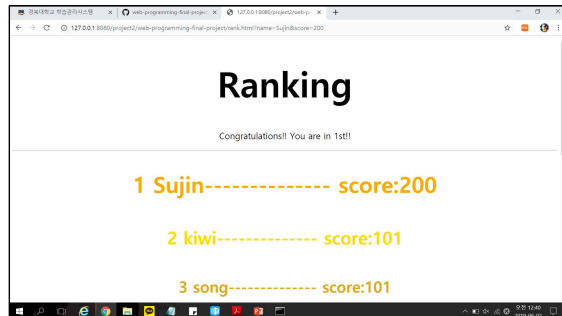


그림 13 랭킹 표시화면 2

3) 웹서버, DB 세부 설명

1. 웹 서버

아파치 서버를 이용하여 구현했습니다. 웹 서버는 name과 score를 브라우저로부터 전달받아 DB에 저장합니다. 플레이어의 ID는 저장 시간 millisecond와 name을 합쳐 고유한 ID를 가지도록 했습니다. 코드는 아래 그림 14와 같습니다.

```
var d = new Date();
db=firebase.database();
db.ref('ranking/'+d.getTime()+name).set({'name':name,'score':score});
```

그림 14 플레이어의 기록을 DB로 보내는 코드

또한 웹서버는 DB로부터 상위 랭커 10명의 정보를 요청하여 받아옵니다. 상위 10명의 정보를 파이어베이스 DB로부터 받아오는 코드는 아래 그림 15와 같습니다.

```
function load_ranker(){
    var ranking = db.ref('ranking/');
    ranking.orderByChild("score").limitToLast(11).on("child_added",function(score) {
        if(count<11){
            rankers[count]={'name':score.val().name,'score':score.val().score};
            console.log(count + " ranker id: " + score.key + "/" + score.val().score);
            console.log("/" + score.val().name);
            count++;
        }
    });
}
```

그림 15 상위 랭커 10명의 정보를 요청하는 코드

2. DB

파이어 베이스 DB를 이용해 실시간 DB를 구현했습니다. 만약 다른 컴퓨터에서 동시에 게임을 플레이하여도 순위를 처리하는 것이 가능합니다. 파이어베이스 실행 화면은 그림 16과 같습니다. 저장 시간과 name을 이용해 ID를 구성한 것을 확인 할 수 있으며 name과 score가 저장되어 있는 것을 확인할 수 있습니다.

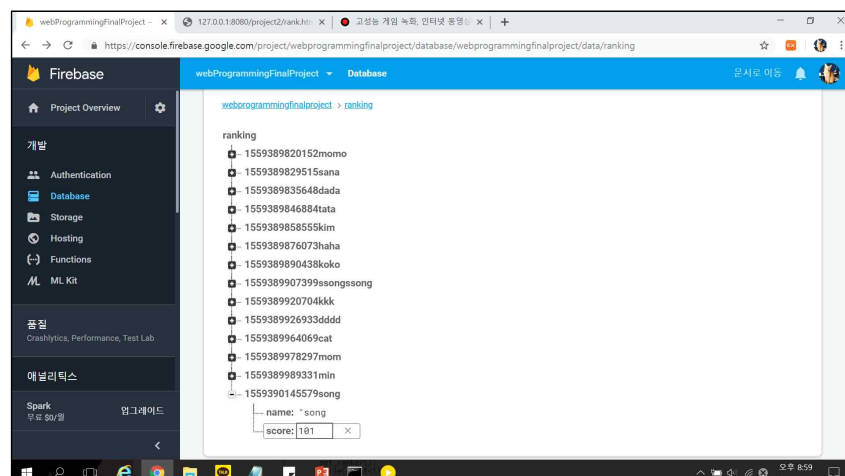
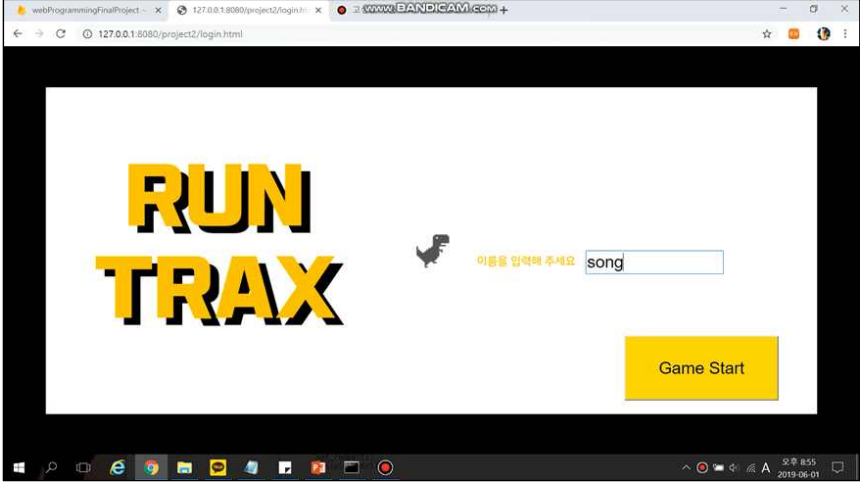
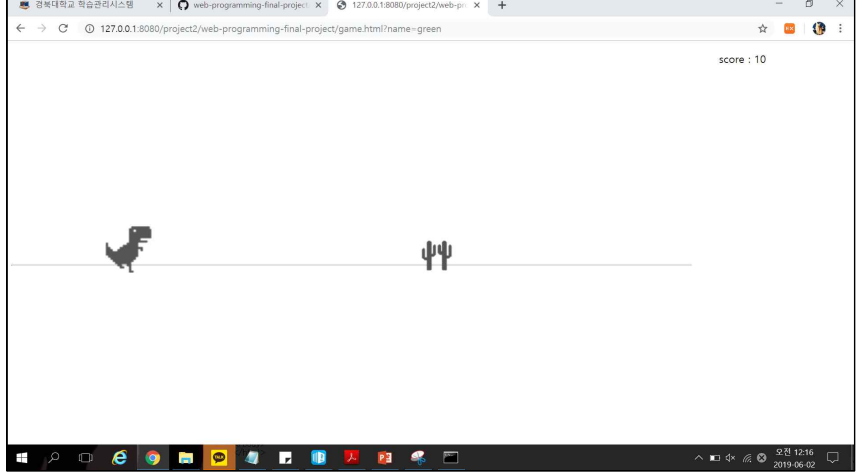
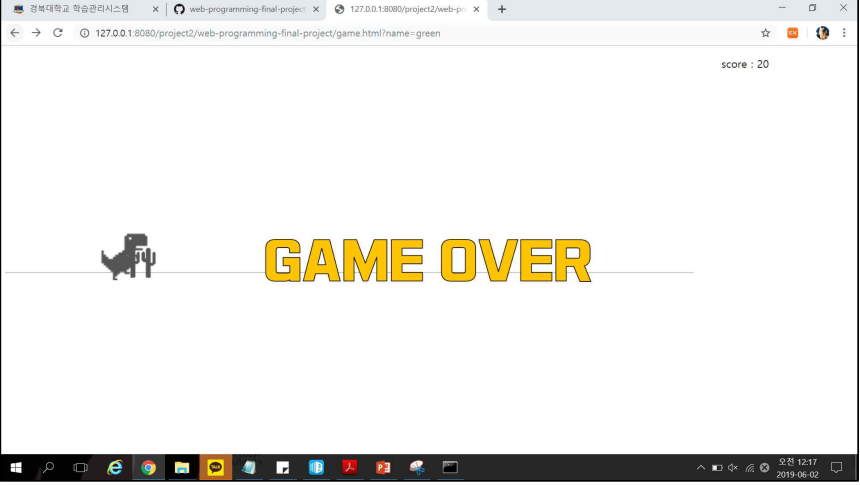
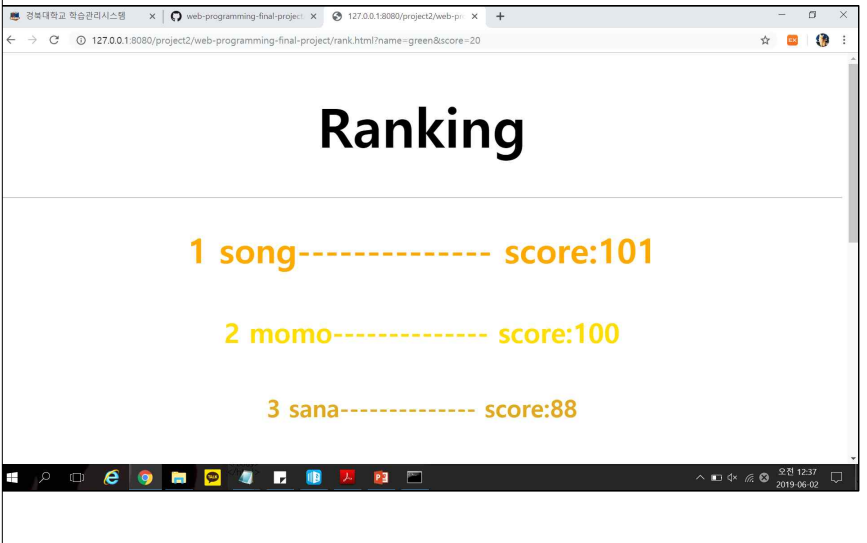
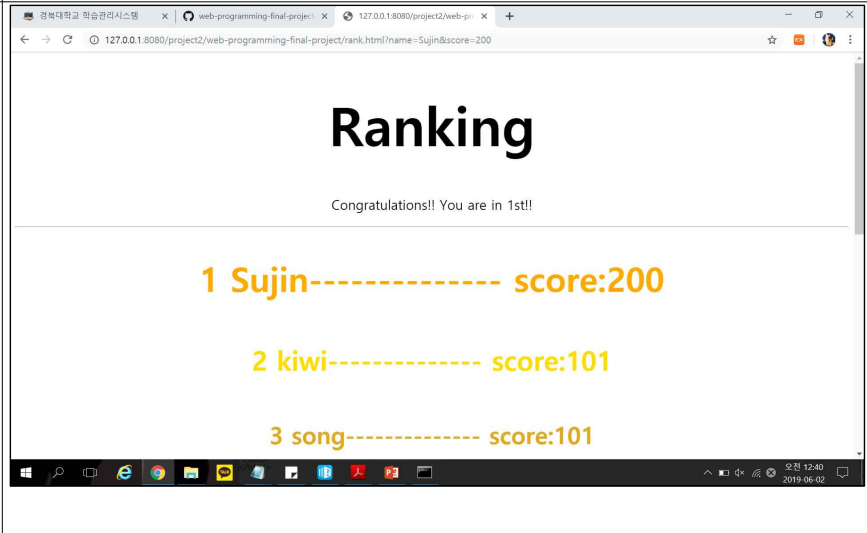


그림 16 파이어베이스 실행 화면

3. 실행 순서

화면 1	
화면 2	
화면 3	

<p>화면 4-1</p>	
<p>화면 4-2</p>	

4. 참조

- 공룡, 장애물 그림 소스 : Chrome 공룡 게임
- html parameter 전달하기 : <https://blog.opid.kr/431>
- firebase 인증 : <https://firebase.google.com/docs/auth/web/google-signin?hl=ko>
- firebase read/write :
<https://firebase.google.com/docs/database/web/read-and-write?hl=ko>
- firebase 조회 : <https://firebase.google.com/docs/database/web/lists-of-data?hl=ko>
- 구현 코드 Github :
<https://github.com/SongYeongJu/web-programming-final-project>
- Firebase 실시간 DB 링크 : <https://webprogrammingfinalproject.firebaseio.com/>