

칼로리 가이드

서비스 안내



Calorie Guide

정확하고 편리한
인공지능 칼로리 관리 서비스

B6890005 김건우

B6890007 김덕용

B6890017 김현우

B6890023 박병익

목차

주제 선정 이유

1. 코로나19 건강현황
2. 주제 선정 이유
3. 기존앱의 한계점
4. 프로젝트 목적

구조

1. Tools
2. Teachable machine
3. 시스템 흐름

주요 기능

1. 사진 인식
2. 음식 추가
3. 일일목표

시연 영상

1. 시연 영상
2. Q&A

주제 선정 이유



1.1 코로나19 건강현황

1.2 주제 선정 이유

1.3 기존앱의 한계점

1.4 프로젝트 목적

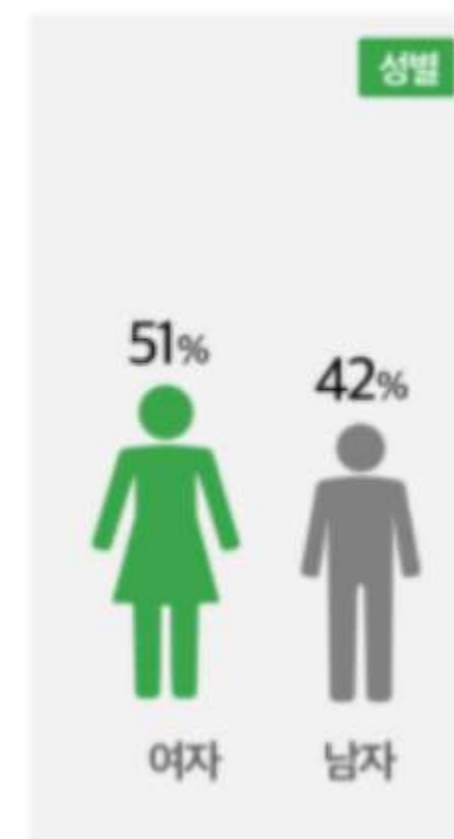
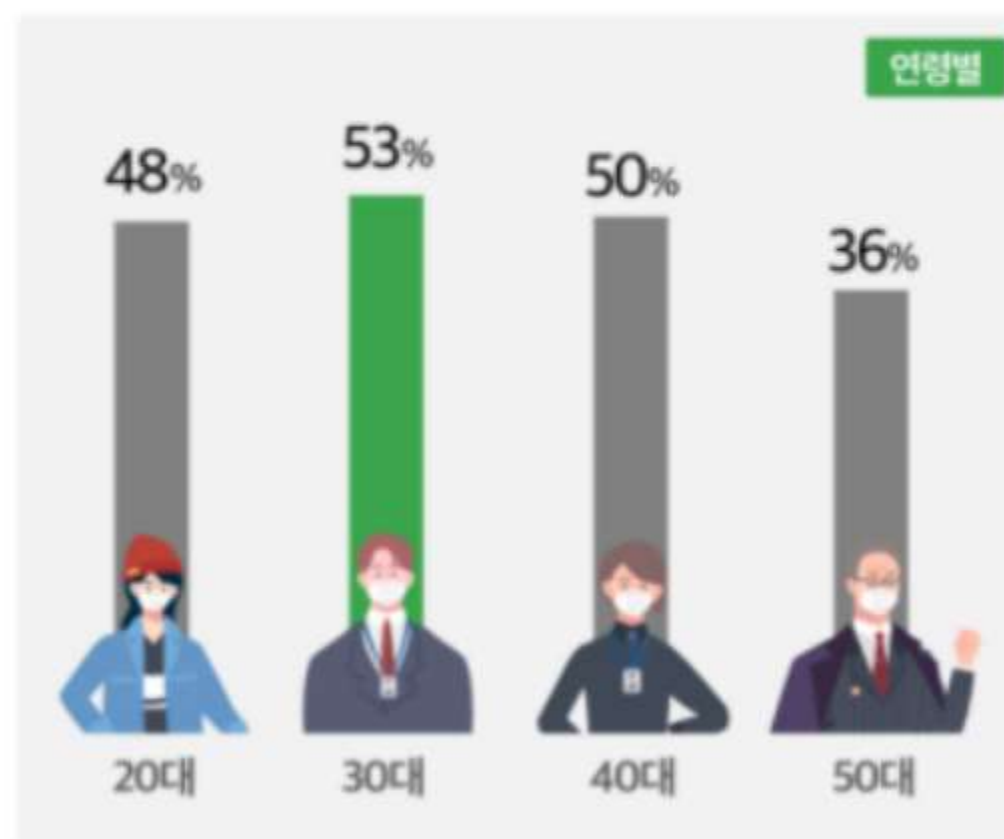
1.1 코로나19 건강현황

국민 10명 중 4명, 코로나19 속 몸무게 3kg 이상 증가

▶ 코로나19 이전 대비,
체중 변화가 있는가?



▶ 체중이 증가했는가? (연령/성별)



1.1 코로나19 건강현황

전반적인 신체 활동량 줄어 운동량 감소 · 영상 시청 시간 증가

▶ 코로나19 발생 전후, 운동 빈도 수



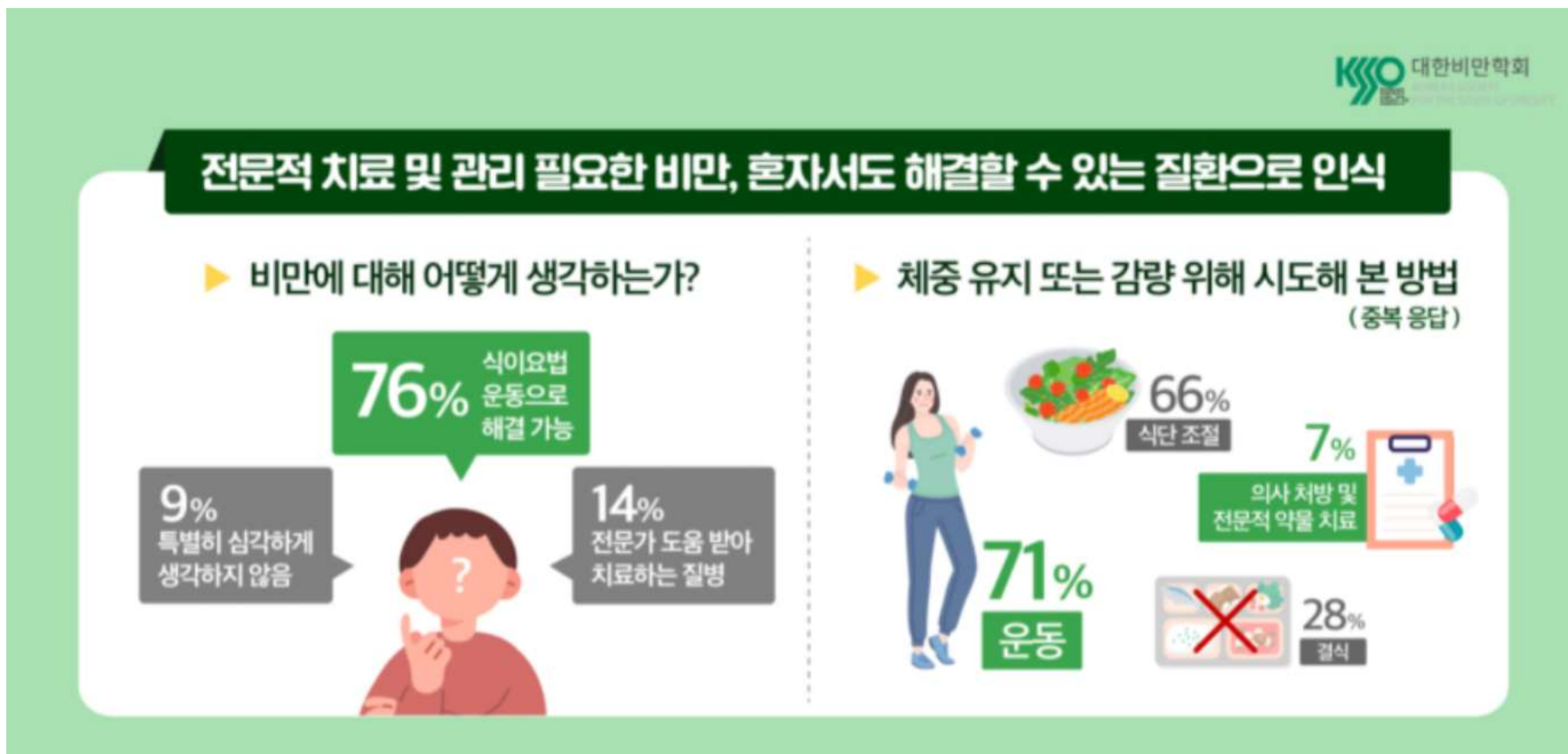
▶ 코로나19 발생 전후, 평균 영상 시청 시간



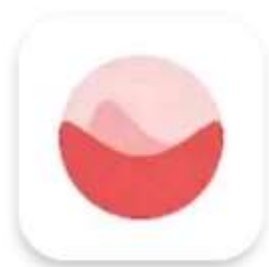
*전국 만 20세 이상 성인 남녀 1,000명, 2021년 3월 29~30일

*대한비만학회 '코로나19 시대 국민 체중 관리 현황 및 비만 인식 조사'

1.2 주제 선정 이유



1.3 기존앱의 한계점



스프린트 Sprint - 자동 식단 기록앱. 다...
스프린트
인앱 구매

열기

Problem

칼로리, 3대 영양소 시각화 부족



밀리그램 - 식단, 운동, 신체 기록 앱
Kilo Inc.

열기

Problem

카메라 인식기능 부재로 불편함

1.4 프로젝트 목적



1. 하루 섭취 성분 시각화
2. 식단관리의 어려움 해소
3. 트레이너들의 회원 관리 부담 해소

구조



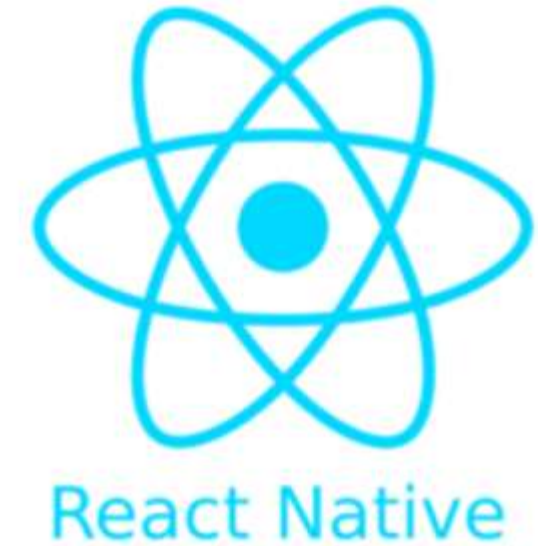
2.1 Tools

2.2 Teachable machine

2.3 시스템 흐름



Teachable
Machine



Teachable Machine

Train a computer to recognize your own images, sounds, & poses.

A fast, easy way to create machine learning models for your sites, apps, and more – no expertise or coding required.

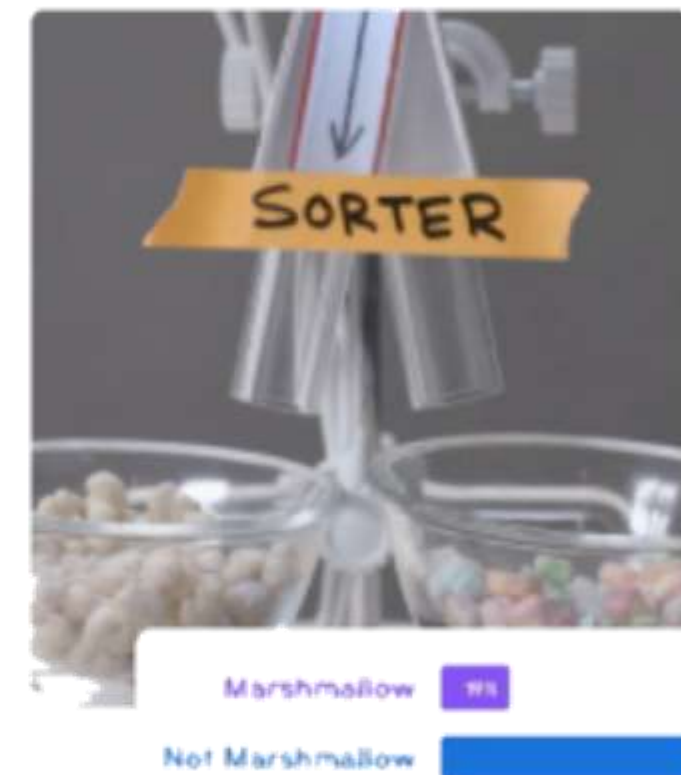
Get Started



[About](#)




[FAQ](#)

[Get Started](#)



2.2 Teachable machine

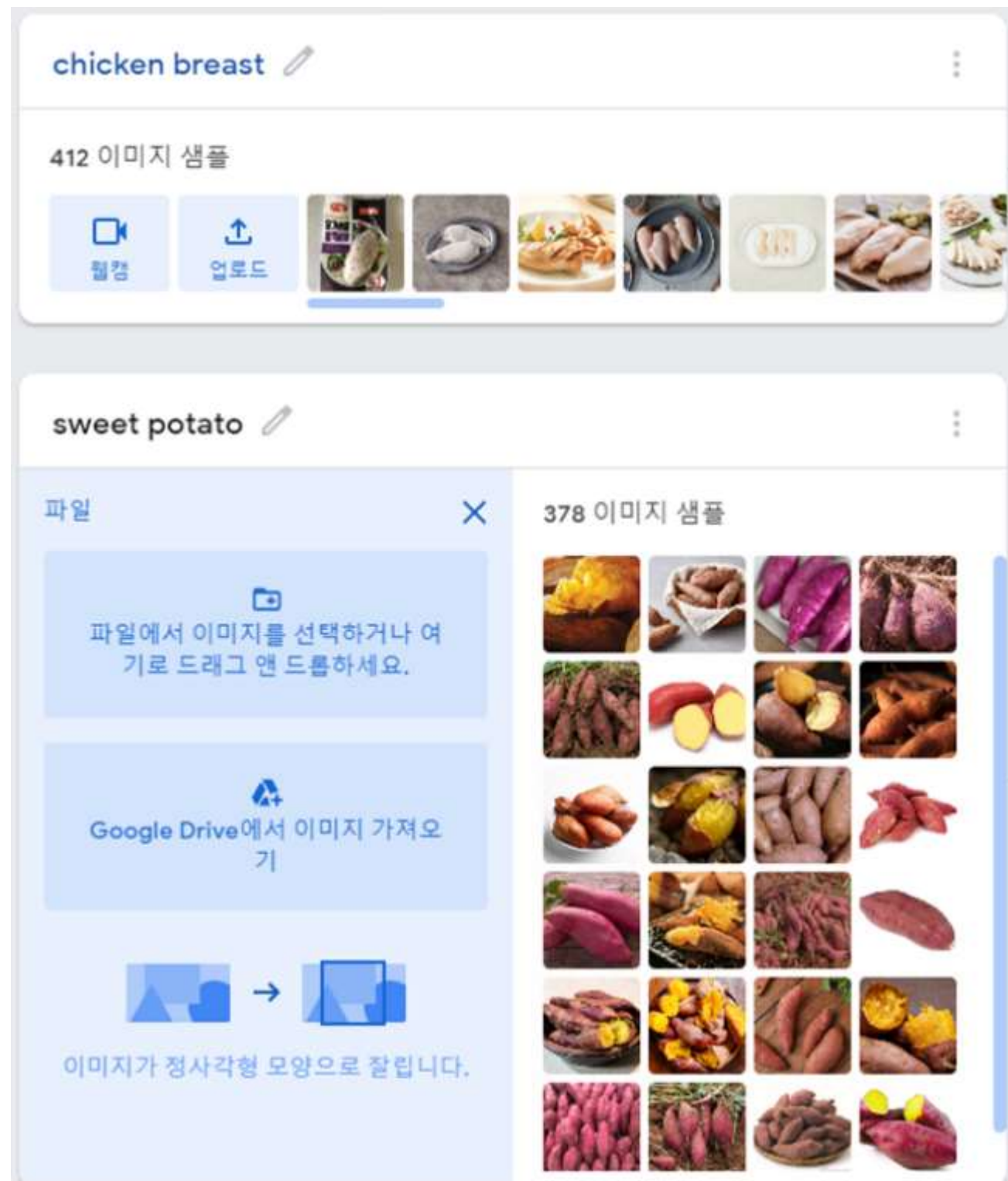
장점

-  실시간 측정상황 파악 가능
-  높은 유지보수성
-  다양한 플랫폼으로 제작 가능

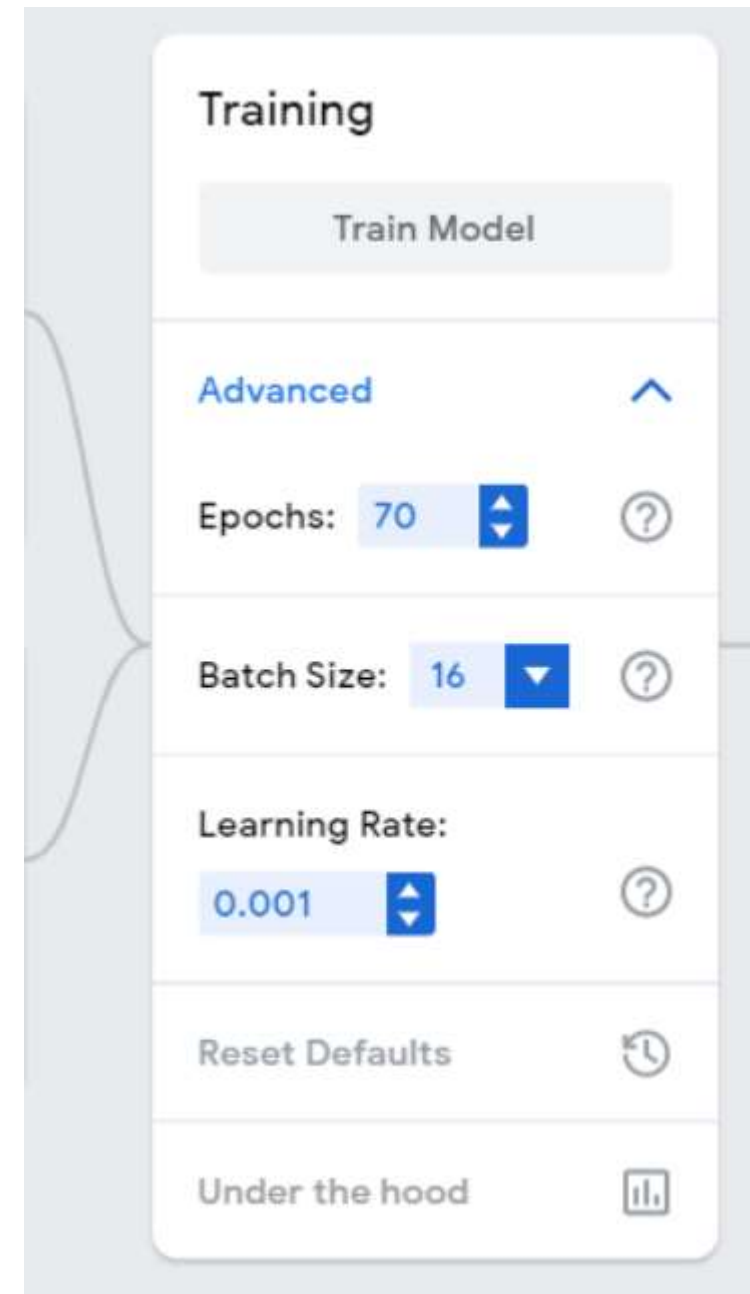


2.2 Teachable machine

Calorie Guide



음식 종류별 300~400개의 이미지 데이터를 수집



모델 트레이닝

Epochs : 70

Batch Size : 16

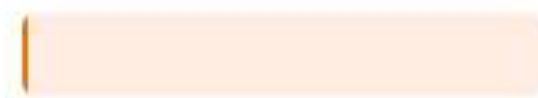
Learning Rate : 0.001

2.2 Teachable machine



Output

chick...
breast



sweet
potato



cucu...



오이



Output

chick...
breast



sweet
potato



cucu...

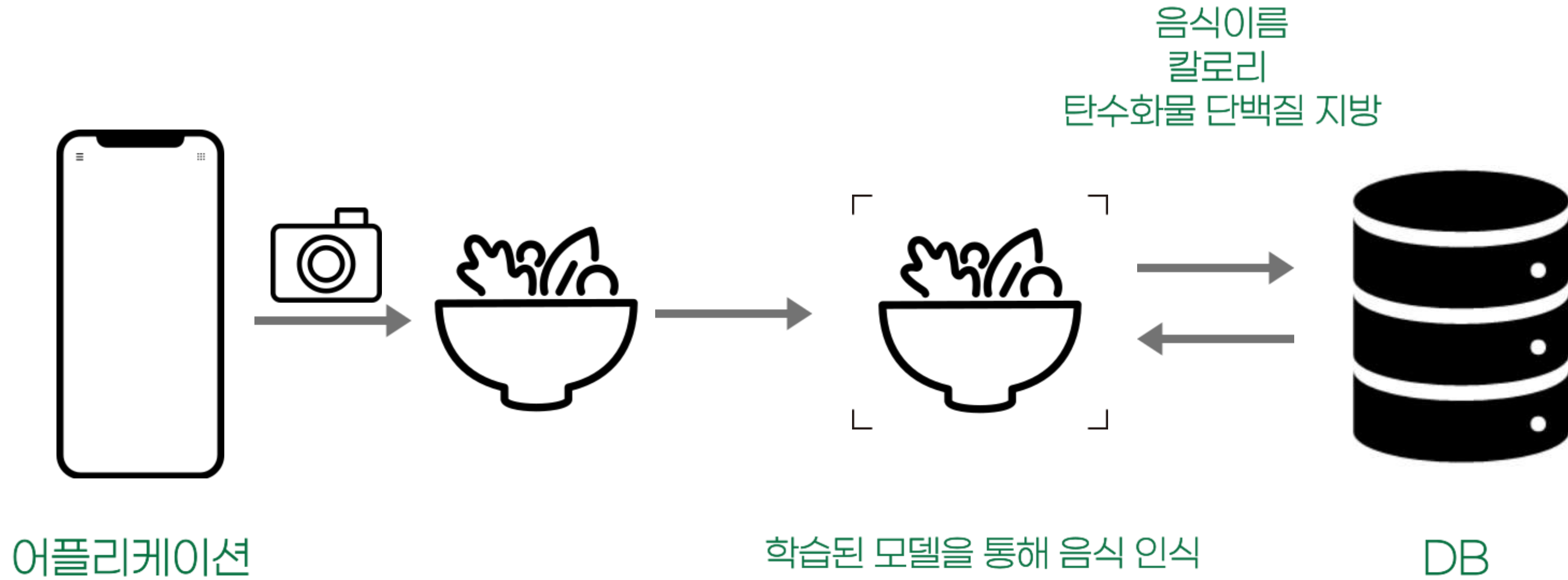


닭가슴살

정확도

약 99%의 높은 정확도를 보여준다

2.3 시스템 흐름



주요 기능



3.1 사진 인식

3.2 음식추가

3.3 일일목표

3.1 사진인식

- 머신러닝의 지도학습을 이용한 학습모델
- 찍은 사진이 뭔지 예측해서 식별하는 학습 모델은 지도학습의 분류 방법 사용
 - >독립변수 : 학습한 사진들
 - >종속변수 : 음식 이름

```
function readURL(input) {
  if (input.files && input.files[0]) {
    var reader = new FileReader();

    reader.onload = function (e) {
      $('.image-upload-wrap').hide();

      $('.file-upload-image').attr('src', e.target.result);
      $('.file-upload-content').show();

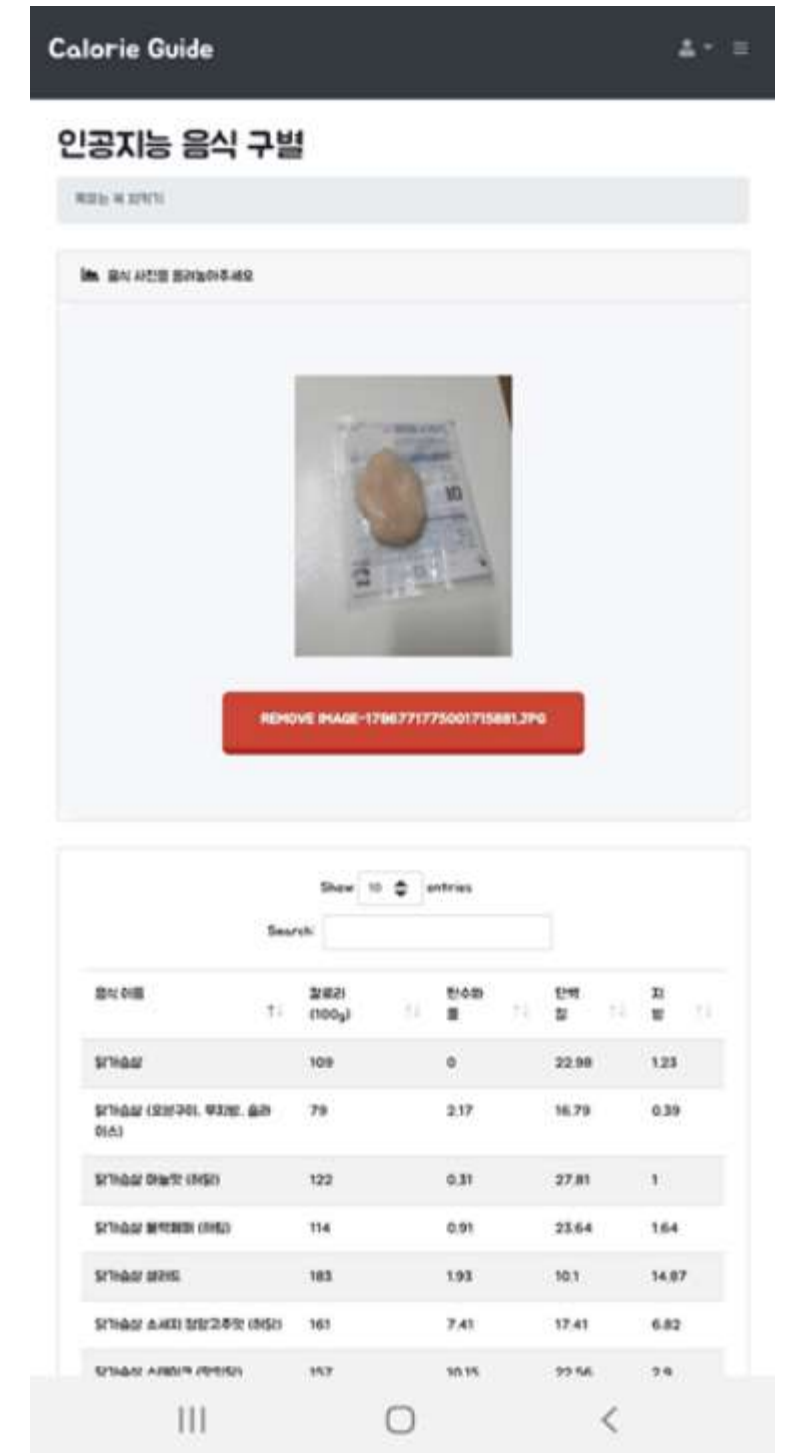
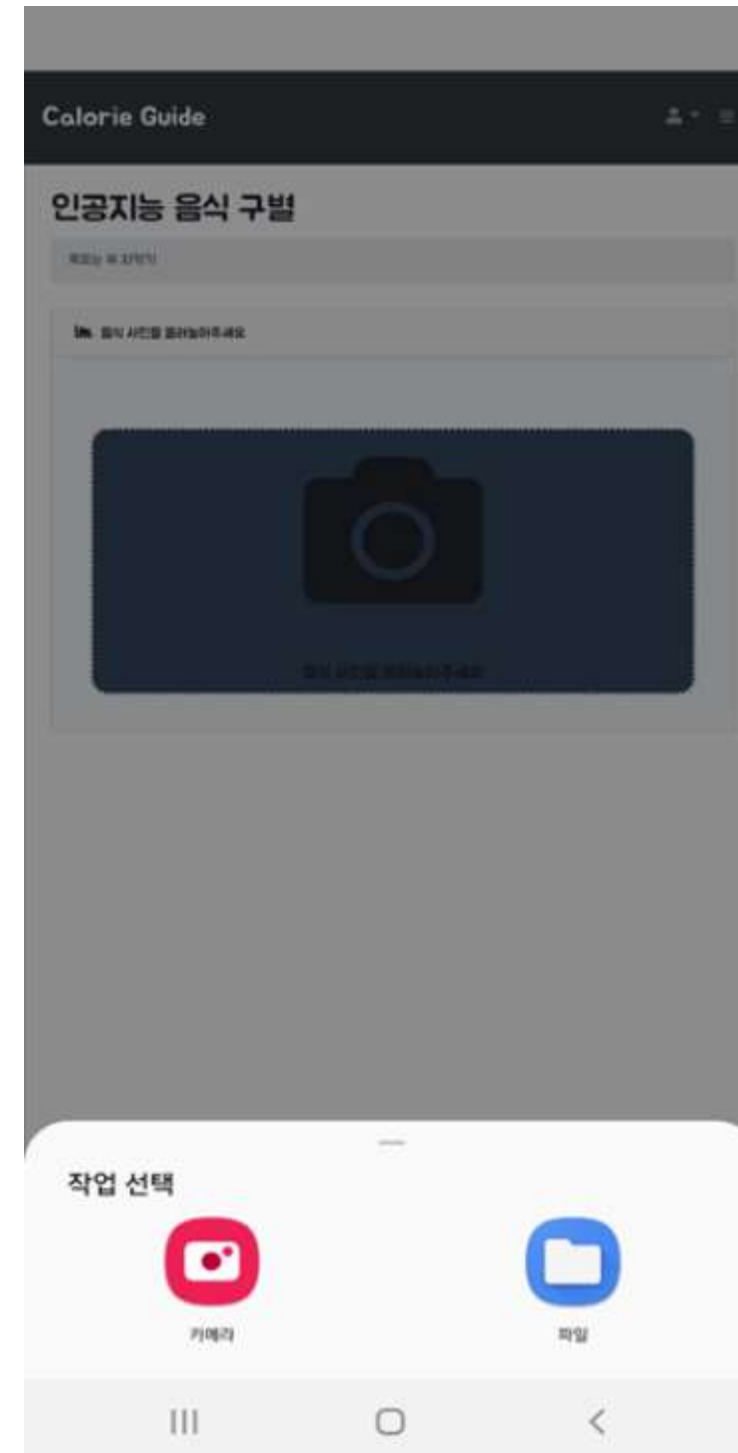
      $('.image-title').html(input.files[0].name);
    };

    reader.readAsDataURL(input.files[0]);

    init();
  } else {
    removeUpload();
  }
}
```

diaryplus.html

음식 사진 찍어서 올리는 부분
올린 후 init() 호출



3.1 사진인식

```
async function init() {  
    var value = ""  
  
    const modelURL = URL + "model.json";  
    const metadataURL = URL + "metadata.json";  
  
    model = await tmImage.load(modelURL, metadataURL);  
    maxPredictions = model.getTotalClasses();  
  
    for (let i = 0; i < model._metadata.labels.length; i++) {  
        value += model._metadata.labels[i]  
        if(i != model._metadata.labels.length-1) {  
            value += ", "  
        }  
    }  
  
    $("#num1").text(value);  
  
    predict();  
}
```

diaryplus.html

3.1 사진인식

```
287     async function predict() {  
288         var jsonArray = new Array();  
289         // predict can take in an image, video or canvas html element  
290         var image = document.getElementById("food-image")  
291         const prediction = await model.predict(image, false);  
292         for (let i = 0; i < maxPredictions; i++) {  
293               
294             var jsonObj      = new Object();  
295               
296             jsonObj.name      = prediction[i].className;  
297             jsonObj.probability = prediction[i].probability.toFixed(2);  
298               
299             jsonObj = JSON.stringify(jsonObj); //String 형태로 파싱한 객체를 다시 json으로 변환  
300             jsonArray.push(JSON.parse(jsonObj));  
301         }  
302         ajax(jsonArray);  
303     }  
304
```

diaryplus.html

3.2 음식추가

전달된 데이터를 json형태로
요청한 주소인 /diary/list로 뿌림

```
function ajax(jsonArray) {  
    var labelContainername = document.getElementById("label-container-name");  
  
    $.ajax({  
        url: '/diary/list', // 요청 할 주소  
        type: 'POST', // GET, PUT  
        data: JSON.stringify(jsonArray), // 전송할 데이터  
        contentType: "application/json",  
        dataType: 'JSON', // xml, json, script, html  
        //beforeSend: function(jqXHR) { // 서버 요청 전 호출 되는 함수 return false; 일 경우 요청 중단  
        success: function(data) {  
            $("#food-table").dataTable({  
                data: data,  
                destroy: true,  
                columns: [  
                    { data: 'kor_F_name' },  
                    { data: 'F_kcal' },  
                    { data: 'F_tan' },  
                    { data: 'F_dan' },  
                    { data: 'F_gi' },  
                    { data: 'F_name' }  
                ],  
                columnDefs: [{  
                    targets: [5],  
                    searchable: false,  
                    visible: false  
                }]  
            });  
  
            $('food-table').show();  
        }, // 요청 완료 시  
        error: function(data) {  
            console.log("error")  
        }, // 요청 실패.  
        //complete: function(jqXHR) {} // 요청의 실패, 성공과 상관 없이 완료 될 경우 호출  
    });  
}
```


diaryplus.html

```
44 /* POST list page. */
45 router.post('/list', function(req, res, next) {
46   var data = req.body;
47
48   probability = 1;
49
50   data.sort(({ probability: a }, { probability: b }) => Math.abs(a - probability) - Math.abs(b - probability));
51
52   var sql = 'SELECT * FROM FOOD WHERE F_name LIKE ' + conn.escape('%' + data[0].name + '%');
53   conn.query(sql, function (err, rows, fields) {
54     if(err) console.log('geury is not excuted. select fail...\n' + err);
55     else res.send(rows)
56   })
57 });
```

diary.js

화면에 나타나는 테이블에 예측한 데이터를 디비에서 읽어와 뿌려줌

Calorie Guide



REMOVE IMAGE-1796771775001715881.JPG

Show 10 entries

Search:

음식 이름	1	2	3	4
고칼로리 (100g)	탄/수화물	단백질	지방	
달걀소시	109	0	22.98	1.23
달걀소시 (오븐구이, 무지방, 슬라이스)	79	2.17	16.79	0.39
달걀소시 마늘맛 (냉장)	122	0.31	27.81	1
달걀소시 참깨맛 (냉장)	114	0.91	23.64	1.64
달걀소시 치즈맛	183	1.93	10.1	14.87
달걀소시 소세지 참깨고추맛 (냉장)	161	7.41	17.41	6.82
달걀소시 스테이크 (맛있습)	157	10.15	22.56	2.9
달걀소시 스테이크 고추맛 (맛있습)	152	9	22	3.1
달걀소시 아자파초 (다도식)	77	3.37	13.95	0.7
달걀소시 칠리 (오븐사)	117	0.7	27.4	0.8

Showing 1 to 10 of 48 entries

Previous

1

2

3

4

5

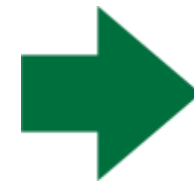
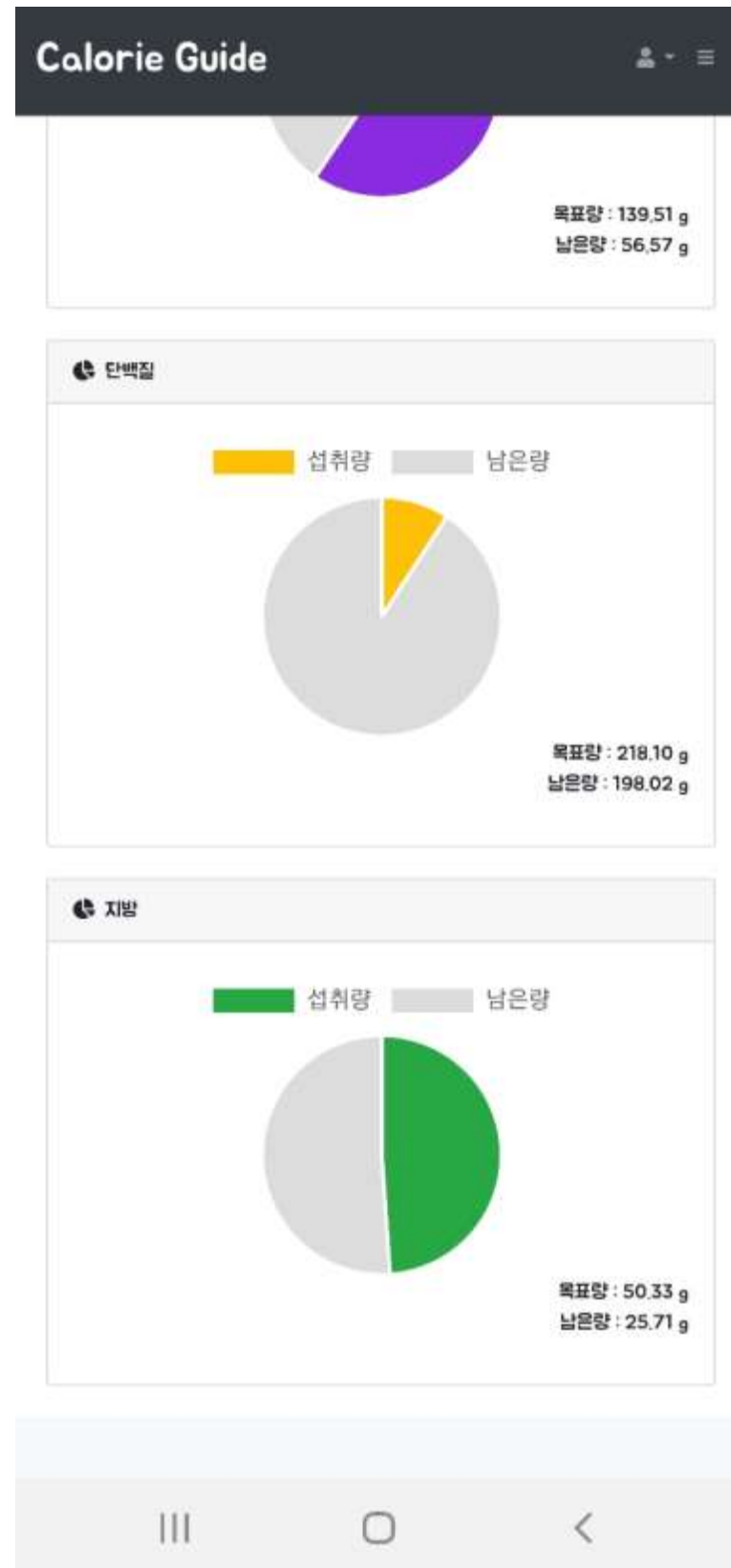
Next

III

O

<

3.3 일일목표



해당 날짜에 맞추어 각각의 날에 해당하는 일일 목표를 표현

```
/* POST list page. */
router.post('/list', function(req, res, next) {
  var data = req.body;

  var id = data.id;

  var result = new Array();

  var newDate = new Date();
  var bgnde = newDate.toFormat('YYYY-MM-DD');
  newDate.setDate(newDate.getDate() + 1);
  var endde = newDate.toFormat('YYYY-MM-DD');

  var sql = 'SELECT * FROM GOAL WHERE `ID` = ? ORDER BY Reset DESC';
  var params = [id];
  conn.query(sql, params, function (err, rows, fields) {
    if(err) {
      console.log('query is not excuted. select fail...\n' + err);
    } else {
      result[0] = rows[0];
      var managerSql = "SELECT * FROM `MANAGER` A WHERE G_date BETWEEN ? AND ? AND `ID` = ?";
      var managerParams = [bgnde, endde, id]
      conn.query(managerSql, managerParams, function (err, rows, fields) {
        if(err) {
          res.send("error");
          console.log('query is not excuted. select fail...\n' + err);
        } else {
          for(var i = 0; i < rows.length; i++) {
            result[i + 1] = rows[i]
          }
          res.send(result);
        }
      })
    }
  })
});
```

goal.js

3.3 일일목표

goal, data로
계산 적용

goal.js

```
function scope(goal, data) {
  var totalScore = 0;
  if(goal * 0.9 <= data && goal >= data) {
    totalScore = 5
  } else if(goal <= data) {
    totalScore = 0
  } else if(goal * 0.8 <= data) {
    totalScore = 4
  } else if(goal * 0.7 <= data) {
    totalScore = 3
  } else if(goal * 0.6 <= data) {
    totalScore = 2
  } else if(goal * 0.5 <= data) {
    totalScore = 1
  }
  return totalScore;
}

function demerit(goal, data) {
  var totalScore = 0;
  if(goal < data && goal * 0.05 + goal >= data) {
    totalScore = -1
  } else if(goal * 0.05 + goal < data && goal * 0.1 + goal >= data) {
    totalScore = -2
  } else if(goal * 0.1 + goal < data && goal * 0.15 + goal >= data) {
    totalScore = -3
  } else if(goal * 0.15 + goal < data && goal * 0.2 + goal >= data) {
    totalScore = -4
  } else if(goal * 0.2 + goal <= data) {
    totalScore = -5
  }
  return totalScore
}
```


demerit, scope 적용

goal.js

```
var totalScore = 0;
var inKcal = 0, inTan = 0, inDan = 0, inGi = 0;
for(var i = 1; i < result.length; i++) {
    inKcal = inKcal + result[i].M_kcal
    inTan = inTan + result[i].M_tan
    inDan = inDan + result[i].M_dan
    inGi = inGi + result[i].M_gi
}
```

```
var G_kcal = result[0].G_kcal;
var G_tan = result[0].G_tan;
var G_dan = result[0].G_dan;
var G_gi = result[0].G_gi;
var G_id = result[0].ID;
```

```
var scopeKcal = scope(G_kcal, inKcal)
var scopeTan = scope(G_tan, inTan)
var scopeDan = scope(G_dan, inDan)
var scopeGi = scope(G_gi, inGi)
```

```
var demeritKcal = demerit(G_kcal, inKcal)
var demeritTan = demerit(G_tan, inTan)
var demeritDan = demerit(G_dan, inDan)
var demeritGi = demerit(G_gi, inGi)
```

```
totalScore = scopeKcal + scopeTan + scopeDan + scopeGi
            + demeritKcal + demeritTan + demeritDan + demeritGi
```

```
var userSql = "UPDATE USER SET `Ranking_score` = ? WHERE `ID` = ?;";
var userParams = [totalScore, G_id];
```

3.3 일일목표

섭취량 = M_data

초과량 = G_data(전체) - M_data

```
function chart(G_data, M_data, num, area, color) {  
  if(num == 1) {  
    new Chart(area, {  
      type: 'pie',  
      data: {  
        labels: ["섭취량", "남은량"],  
        datasets: [{  
          data: [M_data.toFixed(2), (G_data-M_data).toFixed(2)],  
          backgroundColor: [color, '#DCDCDC'],  
        }],  
      },  
    });  
  } else if(num == 2) {  
    new Chart(area, {  
      type: 'pie',  
      data: {  
        labels: ["섭취량", "초과량"],  
        datasets: [{  
          data: [M_data.toFixed(2), (M_data-G_data).toFixed(2)],  
          backgroundColor: [color, '#FF0000'],  
        }],  
      },  
    });  
  } else {  
    new Chart(area, {  
      type: 'pie',  
      data: {  
        labels: ["섭취량"],  
        datasets: [{  
          data: [M_data.toFixed(2)],  
          backgroundColor: [color],  
        }],  
      },  
    });  
  }  
}
```


3.3 일일목표

개인의 목표가 유지, 벌크업, 다이어트 중 어떤 것인지와 현재 체중까지 일일목표 부분에서 보여줌.

goal.html

```
function detail() {
    var jsonArray = {
        id : getCookie("userData")
    }

    $.ajax({
        url: '/user/myPage/detail', // 요청 할 주소
        type: 'POST', // GET, PUT
        data: JSON.stringify(jsonArray), // 전송할 데이터
        contentType: "application/json",
        dataType: 'JSON', // xml, json, script, html
        //beforeSend: function(jqXHR) {}, // 서버 요청 전 호출 되는 함수 return false; 일 경우 요청 중단
        success: function(data) {
            var goal = ""
            if(data[0].Goal == 1) {
                goal = "유지"
            } else if(data[0].Goal == 2) {
                goal = "벌크업"
            } else if(data[0].Goal == 3) {
                goal = "다이어트"
            }
            var date = new Date();
            var nowDate = date.getFullYear() + "년 " + ("0" + (date.getMonth() + 1)).slice(-2) + "월 " + ("0" + date.getDate()).slice(-2) + "일";

            $('.breadcrumb').after("<div><text>" + nowDate + "</text></div><div style='text-align: right; margin-bottom: 10px;'><text>몸무게 : " + data[0].Weight
        }, // 요청 완료 시
        error: function(data) {
            console.log("error")
        }, // 요청 실패.
        //complete: function(jqXHR) {} // 요청의 실패, 성공과 상관 없이 완료 될 경우 호출
    });
}
```

시연 영상

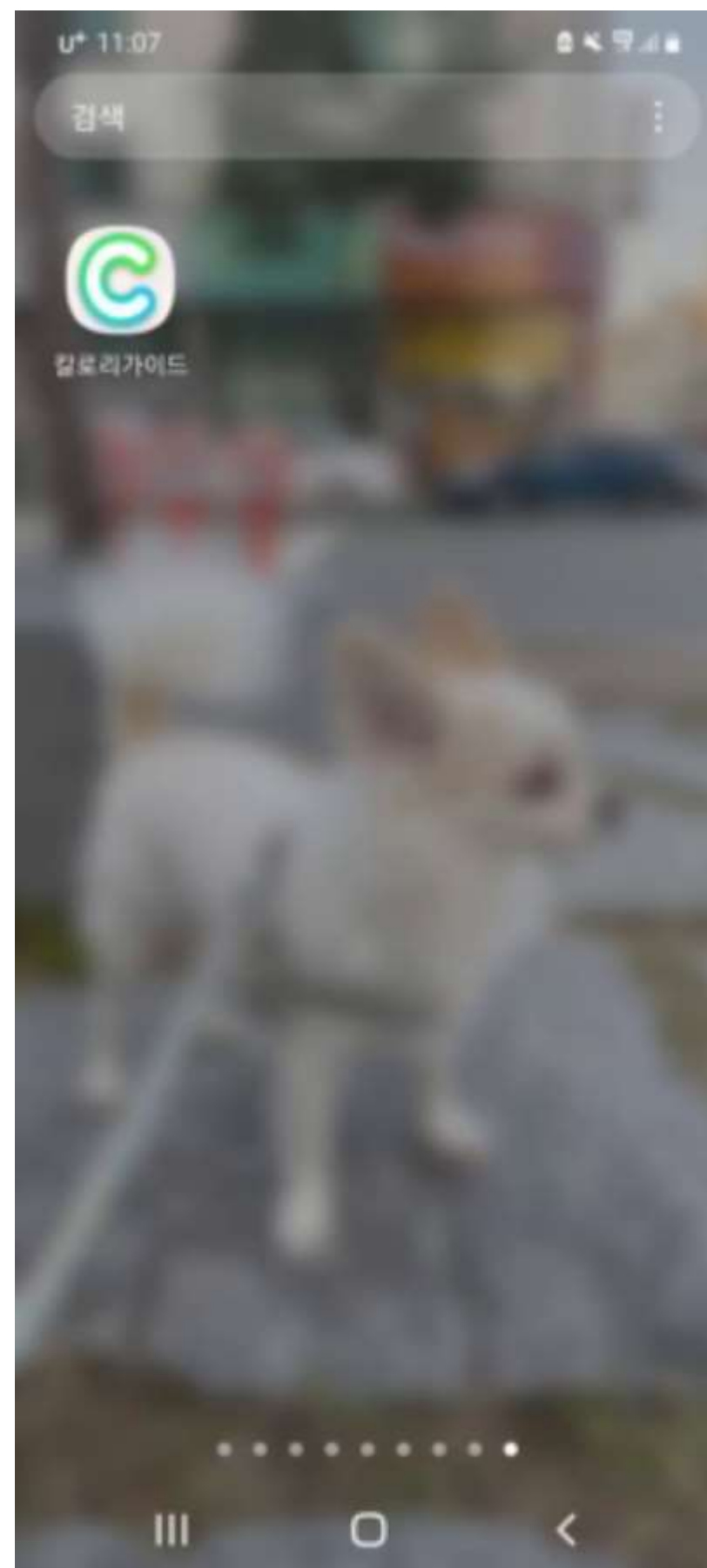


4.1 시연영상

4.2 Q&A



4.1 시연영상



Q&A