|  |
| --- |
| **영상인식을 이용한 식단관리 앱**  2014104104 서원준  **요 약**  최근 건강에 대한 관심이 증가함에 따라, 운동과 식단에 대한 중요성이 강조 되고 있다. 하지만 먹을 때 마다 음식에 대한 영양정보를 찾아보고 먹기란 굉장히 어려운 일이며, 식단을 각 개인에 맞추어 관리해주는 서비스는 찾아보기 힘들며 또한 고가이다. 따라서 휴대하기 간편한 모바일 기기를 이용하여 개인이 섭취하고자 하는 음식의 칼로리와 영양정보를 보다 쉽게 알 수 있게 하여 좀더 쉬운 식단조절을 가능하게 해주는 어플리케이션을 개발하고자 한다.  위의 의도를 가지고서 저번학기에 진행했던 창의적종합설계(캡스톤디자인1)의 결과와 더불어 부족했던 부분인 영상인식을 통해서 여러 음식을 같이 찍어도 각각 인식이 되는 것과, 이번 학기의 중점 부분인 사진으로 음식의 양을 대략적으로 판단, 무슨 음식인지만 판단하고 양은 무조건 1인분으로 계산했던 기존의 방식에서 사진 한번으로 여러 음식의 종류, 양, 총 영양정보까지 알 수 있는 기능을 추가하여 실생활에서 더욱 사용이 용이한 앱을 만드는 것이 목표이다. |

**1. 서론**

**1.1. 연구배경**

최신 스마트 장비들의 다양성과 스마트 장비의 수요가 기하급수적으로 늘어나고 있음에 따라 어플리케이션 개발에 많은 관심이 모이고 있다.

더불어 사람들의 건강에 대한 관심도 증가함에 따라 효율적이고 계산적인 식단을 통해 더 나은 건강을 유지하려는 사람들이 많아졌다.

하지만 기존의 어플리케이션의 경우 이미 짜여진 운동 루틴과 식단들을 알려주는 기능만을 하여 사용자가 먹는 식단을 유동적으로 조절할 수 있는 권한을 주기 힘들고 그로 인해 해당 식단을 유지하지가 매우 힘들었다.

기존의 어플리케이션처럼 틀에 박힌 식단을 알려주는 기능을 벗어나 섭취할 음식들을 분석해 영양정보를 제공하여 사용자가 스스로 조절 할 수 있도록 자율성과 편의성을 동시에 제공하여야 한다고 생각했다. 따라서 카메라를 통해 음식들을 인식하여 그 각각의 음식들의 칼로리와 영양정보를 제공하고 그 결과를 통해 자율적으로 식단을 선택하고 조절할 수 있다.

하지만 기존의 결과물이 한번의 사진촬영으로 모든 기능이 완수되는 것이 아닌, 메뉴를 선택하고, 인식이 안된 메뉴를 추가하는 추가적인 동작이 요구되었기 때문에 편의성이나 기존의 결과에 대해 차별성을 느끼지 못한 것에 대해 안타깝게 생각하였다. 따라서 이번의 연구로써 사진촬영 한번으로 여러 음식을 구분하고, 각각의 양을 체크해서, 그 영양정보를 계산하여 저장하는 기능을 수행하여 편의성이나 차별성 면에서 기존의 식단 관리 앱 들보다 발전된 모습을 보이게 하는 것이 목표이다.

**1.2. 연구목표**

스마트 폰 만 있으면, 언제 어디서든 자신이 먹는 식단을 촬영하여 그 칼로리와 영양정보를 확인 할 수 있는 점과 정리된 식단을 한눈에 봐서 보안해야 할 점을 찾을 수 있는 점을 이용하여, 식단을 조절하는데 어려움을 최대한 줄이는 데에 활용할 것이다.

저번 연구에서는 cloudvision 이라는 구글의 api를 이용하여 음식을 구별 하는 것까지를 결과물로 나타내었다. 하지만 그 api의 특성상 한 사진에 여러가지 음식이 들어갈 경우 정확한 구별이 되지 않고, 그 음식의 양을 정확히 알 수 없어, 영양정보를 체크하는 과정에서 부정확함이 발생하게 된다. 따라서 그 부적확함을 해결하는 것을 연구의 최종적인 목적으로 잡고, 우선 컴퓨터 상에서 여러 개의 음식 사진을 분할시키고, 그 음식 안의 양 높은 정확성으로 측정할 수 있는 것을 중간 목표로 잡는다.

연구의 결과물은 저번 캡스톤 디자인에서 사진 하나로 여러 개의 음식에 정확한 영양정보를 받을 수 있는 어플리케이션이 될 것이다.

**2. 관련연구**

**2.1. 음식을 인식하기 위한 기술**

카메라로 사진을 촬영하는 것은 간단한 일이지만, 그 사진에서 음식의 종류를 가려내는 것은 쉽지 않은 일이다. 현재 그를 위해 사용할 수 있는 기술은 2개정도가 있다.

**2.1.1. 영상처리**

영상 처리(Image processing) 또는 화상 처리는 넓게는 입출력이 영상인 모든 형태의 정보 처리를 가리키며, 사진이나 동영상을 처리하는 것이 대표적인 예이다. 대부분의 영상 처리 기법은 화상을 2차원 신호로 보고 여기에 표준적인 신호 처리 기법을 적용하는 방법을 쓴다.

20세기 중반까지 영상 처리는 아날로그로 이루어졌으며, 대부분 광학과 연관된 방법이었다. 이런 영상 처리는 현재까지도 홀로그래피 등에 사용되지만, 컴퓨터 처리 속도의 향상으로 인해 이런 기법들은 디지털 영상 처리 기법으로 많이 대체되었다. 일반적으로 디지털 영상 처리는 다양한 방법으로 쓰일 수 있으며 정확하다는 장점이 있고, 아날로그보다 구현하기 쉽기도 하다. 더 빠른 처리를 위해서 파이프라인과 같은 컴퓨터 기술들이 쓰이기도 한다.

* Open cv 사용예정 / 안드로이드에서도 사용 가능 / 사진에 2가지 이상의 음식 포함시, 각 음식으로 사진 분할/수저의 크기를 기준으로 음식의 양 나누기

**2.1.2.기계학습**

기계 학습(機械學習) 또는 머신 러닝(영어: machine learning)은 인공 지능의 한 분야로, 컴퓨터가 학습할 수 있도록 하는 알고리즘과 기술을 개발하는 분야를 말한다. 가령, 기계 학습을 통해서 수신한 이메일이 스팸인지 아닌지를 구분할 수 있도록 훈련할 수 있다.

기계 학습의 핵심은 표현(representation)과 일반화(generalization)에 있다. 표현이란 데이터의 평가이며, 일반화란 아직 알 수 없는 데이터에 대한 처리이다. 이는 전산 학습 이론 분야이기도 하다. 다양한 기계 학습의 응용이 존재한다. 문자 인식은 이를 이용한 가장 잘 알려진 사례이다.

이러한 기계학습으로 앞서 말한 영상의 인식에서 나온 데이터의 분석을 통해 음식의 종류를 구별 할 수 있다. 현재 이러한 것을 응용한 사례로 사람의 각 얼굴을 구별하는 안면인식 시스템 등이 보안등의 분야에 사용되고 있다.

->구글에서 오픈소스로 공개된 머신러닝 툴 텐서플로우

**2.1.2. 구글 cloud vision api**

구글에서 개발하고 무료로 배포한 api로 이미지를 인식하는데, 다양한 기능을 가지고 있다. 표정을 인식하여, 사진의 분위기를 파악하는 것부터, 사물, 음식, 회사 로고등을 인식하는 것도 가능하고, 연령제한 (ex. 성인 자료)등을 파악하는 것도 가능하다. 다양한 플렛폼에서 사용이 가능하며, 안드로이드도 그에 포함된다. 이미 전 단계에서 클라우드 비전을 사용하여 사진 속 한가지 음식을 구별하는 것은 구현하였으며, 이번 연구에서 그 기능을 영상처리를 통해서 더 확장 시킬 것이다

**2.1.2. 안드로이드 SQLite**

**SQLite**는 MySQL나 PostgreSQL와 같은 데이터베이스 관리 시스템이지만, 서버가 아니라 응용 프로그램에 넣어 사용하는 비교적 가벼운 데이터베이스이다. 영어권에서는 '에스큐엘라이트’ 또는 '시퀄라이트’라고 읽는다.

일반적인 RDBMS에 비해 대규모 작업에는 적합하지 않지만, 중소 규모라면 속도에 손색이 없다. 또 API는 단순히 라이브러리를 호출하는 것만 있으며, 데이터를 저장하는 데 하나의 파일만을 사용하는 것이 특징이다. 버전 3.3.8에서는 풀텍스트 검색 기능을 가진 FTS1 모듈이 지원된다. 컬럼을 삭제하거나 변경하는 것 등이 제한된다.

안드로이드에 내장되어 있어, 서버를 따로 구현하지 않고도 기록한 내용을 저장 할 수 있어 용이하다.

이는 미리 음식의 이름과 칼로리, 각 영양성분을 저장해 놓고, 클라우드 비전api가 내놓은 사진 분석 결과에 맞추어 저장된 값과 비교하기 위하여 쓰인다.

**2.5. 기존 연구의 문제점 및 해결 방안**

**2.5.1. 연구의 문제점**

기존 연구에서의 가장 큰 문제점은 두 개 이상의 음식이 한 사진에 들어있을 때 정확도가 아주 낮아지기 때문에, 한번에 두가지 이상의 음식을 섭취할 경우 사진을 두번 찍어 저장하거나, 사진은 한 사진만 찍고서 다른 메뉴를 수동으로 추가해 줘야하는 문제점이 있었다. 따라서 이번에는 그 문제점을 해결하기 위한 구체적인 방안을 소개하고자 한다.

**2.5.2. 해결 방안**

**2.5.2.1. 영상처리**

현재 나온 어플리케이션은 사람의 눈, 코, 입을 파악해 얼굴의 위치를 파악하여 그것을 다른 영상으로 대체하는 정도의 영상처리를 쓸 수 있다. 따라서 그릇의 형태 등을 파악하여 그릇이 몇 개인지 정도는 금방 알 수 있으며, 한 접시에 여러 개의 음식이 담겼더라고 해도 그 형태를 파악하여 그 구역을 지정하거나 나눌 수 있다. 따라서 여러 개의 음식을 각각 잘라내어 분석하면 더 높은 정확성으로 음식의 종류를 파악 할 수 있다. 또한 음식의 양을 판별하는 것도 식사에서 항상 쓰이는 컵이나, 수저 등의 모양을 파악하여 그 크기와 음식의 크기를 비교하면서 음식의 양도 측정해 낼 수 있을 것이다.

**3. 프로젝트 내용**

**3.1. 시나리오**

M전체적인 시나리오는 저번 프로젝트와 동일하다. 결과적으로 최종적인 연구의 결과는 영상처리로써 나오는 추가적인 편의성을 추가한 단계이기 때문이다. 따라서 본 프로젝트에서는 영상 처리를 주로 다룰 예정이며, 예상 되는 시나리오는 전 결과물에서 한, 두개의 엑티비티가 추가된 모습일 것으로 생각된다.

**3.1.1. MainActivity**

MainActivity는 두 개의 버튼을 가진다. 음식을 등록하는 버튼은 눌렀을 시 카메라나**** 앨범을 선택하는 창을 띄우고, 선택을 했을 경우에는 별도의 MainActivity 생성하지 않고 바로 앨범이나 사진을 통해 사진을 가져온다.

**[그림 1]메인**

**3.1.2. FoodActivity**

FoodActivity는 찍거나 가져온 사진을 api를 통해 분석하여 나온 결과값을

****음식의 이름과 칼로리가 저장된 데이터 베이스와 비교하여, 맞는 음식들을

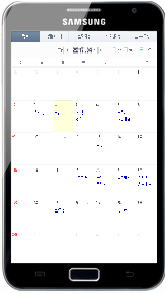
가져와 칼로리와 함께 띄워준다. 또한 인식이 되지 않더라도 사용자가 임의로 먹은 음식을 검색을 통해 추가 할 수 있도록 한다. 추가적으로 음식이 여러 개 일 경우에는 나누어진 사진 별로 맞는 음식을 고를 수 있게 한다.

**[그림 2]음식**

**3.1.3. SelectActivity**

때를 대비하여, 사용자가 음식으로 인식되어 추출된 부분의 사진이 나열된 이 엑티비티에서 음식을 인식한 사진만 골라 낼 수 있게 하는 기능을 가진다. 더불어 음식이 아닌 것이 인식되었을 때 뿐 아니라 본인이 먹지 않은 음식을 골라 낼 수도 있을 것이다.

**3.1.4. RecordActivity**

 **[그림 4]**는 메인 화면에서 기록 살펴보기 버튼을 통해 이동하거나, 음식을 분석한 후 저장을 했을 시 자동으로 넘어간다. 저장된 기록의 이름은 저장된 시간으로 분류가 되며, 자동으로 리스트에 업데이트가 된다. 메인에서 살펴보기 버튼을 통해 이동했을 시에는 인텐트 내부의 값을 살펴보고, 리스트에 저장을 하지 않는다. 리스트를 누르면, 먹은 식단과 음식 사진, 영양정보를 한번에 파악 할 수 있다. 이때 저장된 사진은 각각 나누어진 사진이 아닌 모든 음식이 들어있는 사진으로 한다.

기존의 결과물은 캘린더가 아닌 단순한 리스트를 사용하였으나 편의성을 증대시키기 위해 캘린더를 쓰거나 다른 디자인을 쓸 계획이다.

**[그림 3]캘린더 화면**

**3.2. 요구사항**

**3.2.1. 음식 구별에 대한 요구사항**

**-** 미리 많은 경우를 학습시켜 음식의 종류를 구별 할 수 있도록 한다.

**-** 학습시킨 음식의 경우, 그 영양성분과 칼로리를 입력해두어, 구별이 되었을 때 바로 그 정보를 출력하고 기록할 수 있도록 한다.

**-**구별된 음식에 대해서는 영상인식을 통해 그 음식의 대략적인 양을 판단 할 수 있게 되어야 한다.

-양이 판단된 음식은 1인분 음식의 영양정보를 바탕으로 각각 영양정보를 계산하여 저장한다.

**3.2.2. 기록에 대한 요구사항**

- 각 날짜 칸에는 음식이 저장된 시간을 남긴다.

- 저장하는 단계에서 각 음식이 아침, 점심, 저녁, 간식 등을 표시할 수 있도록 한다.

- 각 날에 섭취된 음식들을 모두 모아 하루 총량을 계산해서 표시한다.

- 각 날짜로 자세히 보기를 하면, 그때 정확한 식단을 나타낼 수 있도록 한다.

- 식단과 함께 찍은 사진도 확인 할 수 있게 한다.

**3.2.2. 편의성에 대한 요구사항**

- 분석한 결과가 중복될 경우 팝업을 띄워, 어떤 음식이 맞는 것인지 사용자에게 판단하게 한다.

- 분석을 실패할 경우나 추가적인 음식 섭취가 있을 경우, 리스트에서 검색을 통해 바로 추가할 수 있도록 하며, 그 과정에서 칼로리를 바로 알 수 있게 한다. (예시 참조)

- **SelectActivity**를 통하여 여러명이 함께 식사를 하는 사진을 찍어도 어느 정도 자신이 먹은 만큼의 음식을 구분하여 계산 할 수 있게 될 것이다.

**3.2.3. 영상인식에 대한 요구사항**

- 음식이 여러 개인 상황에서 각 음식을 분할할 경우 각 사진은 따로 기록하여 api를 사용하고, 분석 결과를 나타낼 때 보여져야 한다.

- 음식의 양을 구별할 때에는 사진에 그 크기의 기준이 될 물체(수저, 컵)등이 들어가도록 해야한다.

- 음식의 양을 측정할 때는 단순히 사진상에서의 면적을 조사할 것이 아니라 그릇의 모양과 숨겨진 범위를 고려하여 높은 정확도를 가질 수 있게 한다.

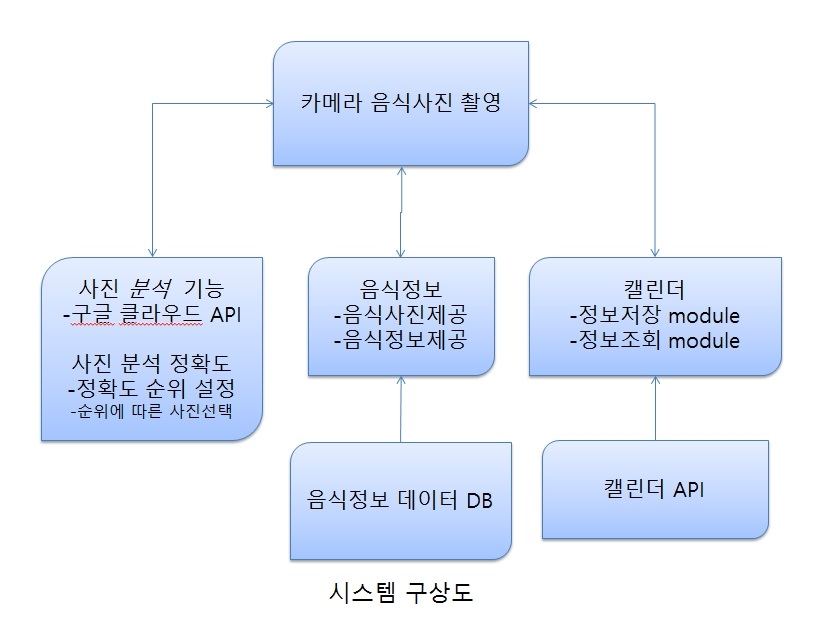


**[예시]**

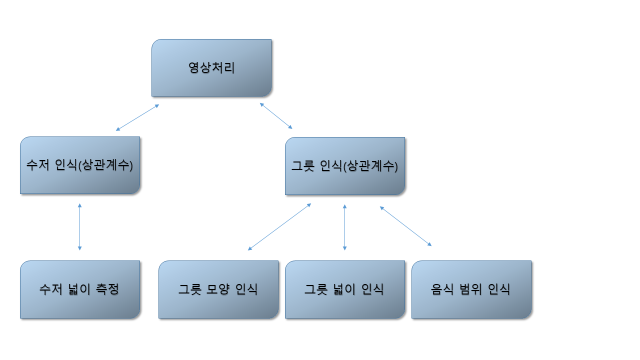
**3.3 시스템 설계**

**3.3.1. 시스템 구성도**

시스템 구성도는 다음과 같다.

****[그림5]

또한 이번 연구에서는 핵심적으로 영상 처리로써 음식의 양을 구별해 내는 기능을 수행해내야 하는데, 영상처리 파트만의 시스템 구상도는 다음과 같다.



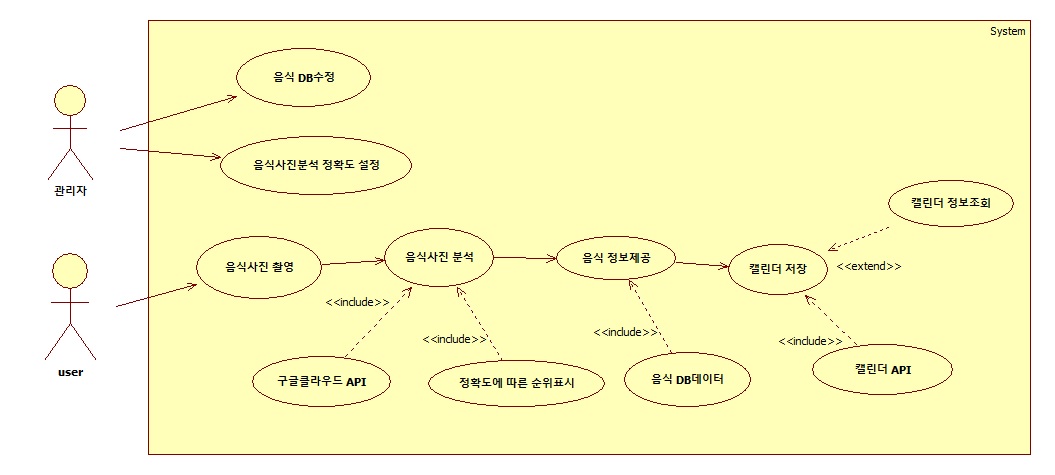
영상 인식파트 시스템 구상도

[그림6]

**3.3.2 UML Diagram을 통한 시스템 모델링**

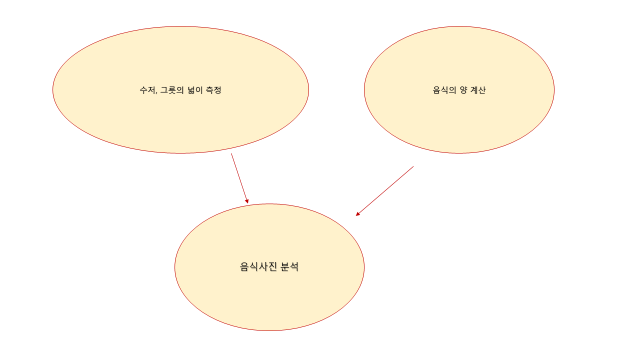
**3.3.2.1. Use Case Diagram**

유스케이스 다이어그램은 다음과 같다.

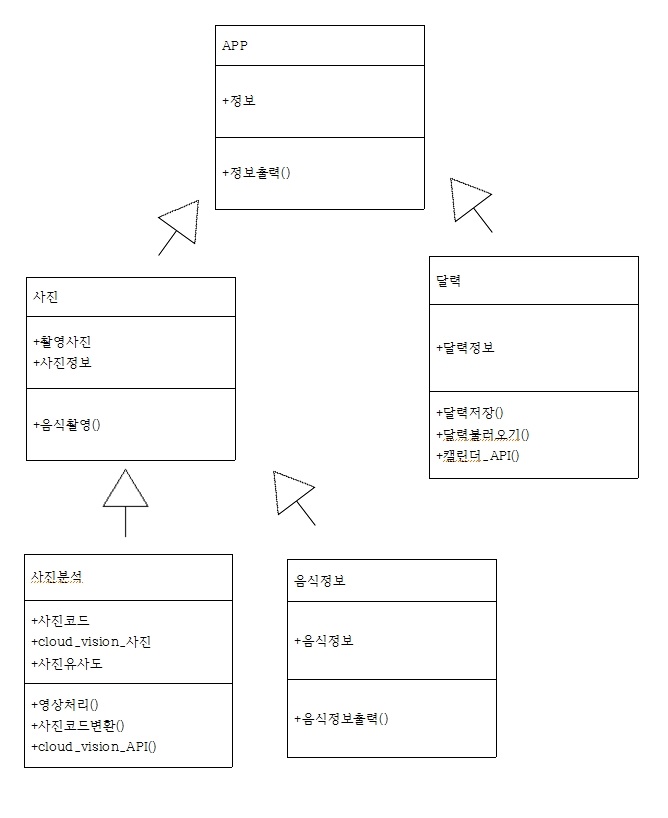
****

[그림7]

위의 유스케이스 다이어그램에서 음식사진 분석에 대한 부분을 이번 프로젝트에서 추가시킬 예정이다. 그 추가된 유스케이스의 내용은 다음과 같다



[그림8]

**3.3.2.2. Class Diagram**

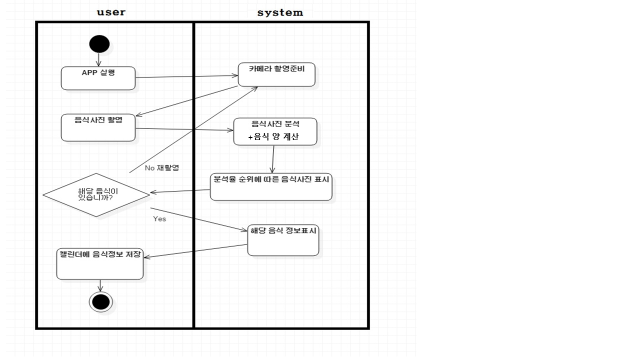
[그림 7]

클래스는 전체적인 프로그램을 담당하는 app 클래스에 사진 인식을 수행하는 클래스와 캘린더를 관리하는 클래스로 각각 나뉘게 된다. 이때 사진을 인식하는 클래스는 하위 클래스로 사진을 분석하는 클래스와 음식 정보를 저장하여 상황에 맞게 제공하는 클래스를 둔다.

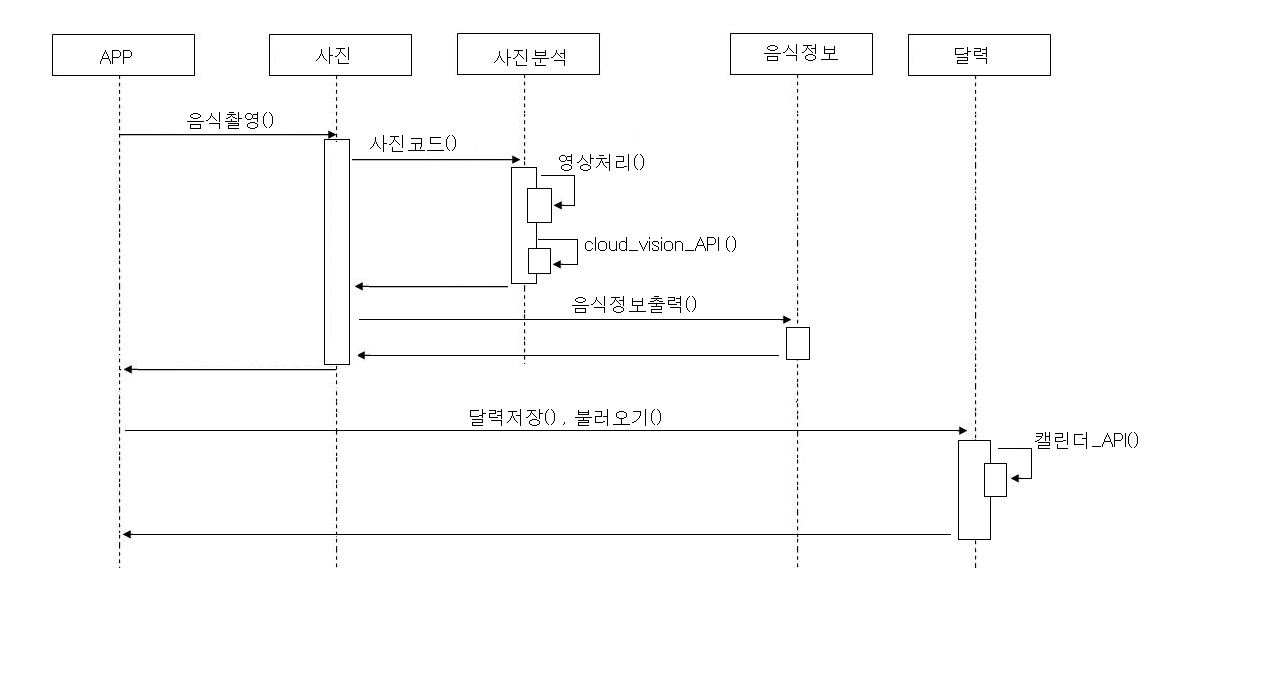
**3.3.2.3. Activity Diagram**

Activity는 사실 매우 간단하다. 사용자가 앱을 실행시켜서 사진을 찍으면, 그 사진에서 각각의 음식을 구별하여 분석 과정을 거쳐, 그 영양 정보를 출력하고, 저장해 준다.

추가적으로 음식의 분석 과정에서 음식의 양을 측정하는 작업을 수행한다.



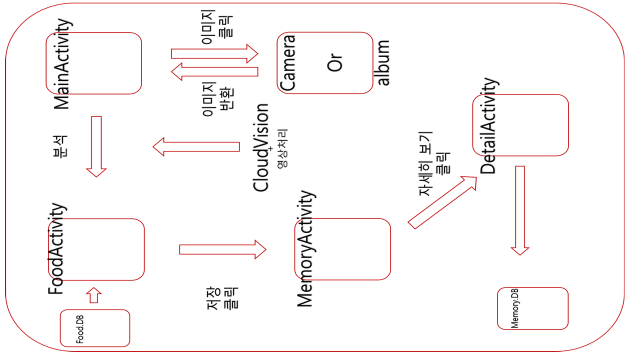
[그림8]

**3.3.2.4. Sequence Diagram**

[그림 9]

어플리케이션의 전체적인 Sequence diagram을 나타내고 있다. 사용자가 사진을 찍으면, 그 사진은 사진 클래스를 거쳐서 사진 분석 클래스로 넘어가게 되는데, 이때 영상처리를 통하여 음식 수량 측정 수치와, 사진 내부의 음식들을 각각 구별하여 저장하고, 그 저장된 단수의 음식들은 cloud\_vision\_API를 통하여, 무슨 음식인지를 인식되게 된다. 인식된 음식은 미리 저장된 음식데이터 베이스에서 비교되어 일치하는 음식의 정보를 사용자에게 출력하는 과정을 거쳐 사용자가 음식에 대한 정보를 인식하게 된다.

**3.3.2.5. State Diagram**



[그림 10]

각각의 주요 기능에 맞추어 디자인 하였다. 단순한 기능을 재빠르게 하는 것이 핵심이므로 몇 번의 버튼 클릭을 통해 저장까지 완료가 되는 것이 1순위였으며, 그러면서도 필요한 기능을 빼먹지 않고 넣는 것이 나머지였다. 또한 DB를 사용하게 되는 엑티비티를 한정하여, 조금 더 매끄러운 앱이 될 수 있도록 하였다.

**3.3.2.6. Event Table**

각 입력에 따라 발생할 수 있는 모든 이벤트를 표기하였다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CD\_PROJECT Event Table** | | | | |
| **Event** | **Trigger** | **Source** | **Use Case** | **Response** |
| **사용자 버튼 클릭** | **없음** | **사용자** | **입력대기** | **이미지 호출 수단 선택창** |
| **사용자가 카메라 클릭** | **파일 생성** | **사용자** | **촬영대기** | **카메라 호출** |
| **사용자가 앨범 클릭** | **파일 생성** | **사용자** | **선택대기** | **앨범호출** |
| **사용자가 사진 촬영, 선택시** | **반환** | **사용자** | **영상처리** | **영상처리로 넘어감** |
| **사진 분석** | **영상처리** | **사용자** | **값들(그릇크기, 음식 양 등) 저장** | **완료 메시지 출력**  **음식별 자동 분할.** |
| **음식 검색시** | **텍스트 에디터에 입력, 변경** | **사용자** | **검색** | **결과 리스트에 출력** |
| **리스트 클릭시** | **리스트에 정보 받아옴** | **사용자** | **정보 확인 완료** | **텍스트 뷰에 추가한 음식과 영양성분 더함** |
| **저장버튼 눌렀을 시** | **엑티비티 이동, 저장** | **사용자** | **현재까지 기록 확인** | **엑티비티 변경, 기록 리스트 출력** |
| **기록 리스트 클릭시** | **데이터 존재시** | **사용자** | **내부 데이터 확인** | **사진, 영양정보 출력** |

분석이나 저장, 불러오기등 사용자에게 단순한 요구사항을 적게 요구하기 때문에, 이벤트의 종류는 상당히 적다.