버전 1.0

2019.11.13

Design Specification

팀9

가웅현

조현우

차승호

목차

[1. Preface 5](#_Toc24580696)

[1.1 Readership 5](#_Toc24580697)

[1.2 Document Structure 5](#_Toc24580698)

[A. Preface 5](#_Toc24580699)

[B. Introduction 5](#_Toc24580700)

[C. System Architecture 5](#_Toc24580701)

[D. Front-end Architecture 5](#_Toc24580702)

[E. Back-end Architecture 5](#_Toc24580703)

[F. Protocol Design 5](#_Toc24580704)

[G. Database Design 5](#_Toc24580705)

[H. Testing plan 5](#_Toc24580706)

[I. Development Environment 5](#_Toc24580707)

[J. Development Plan 5](#_Toc24580708)

[K. Index 5](#_Toc24580709)

[1.3 Version Update History 5](#_Toc24580710)

[2. Introduction 5](#_Toc24580711)

[2.1 Applied Diagram 5](#_Toc24580712)

[A. UML 5](#_Toc24580713)

[B. Class Diagram 5](#_Toc24580714)

[C. State Diagram 5](#_Toc24580715)

[D. Sequence Diagram 6](#_Toc24580716)

[E. ER Diagram 7](#_Toc24580717)

[2.2 Applied Tools 7](#_Toc24580718)

[A. draw.io 7](#_Toc24580719)

[B. ERDCloud 8](#_Toc24580720)

[C. PowerPoint 8](#_Toc24580721)

[D. Web Sequence Diagrams 9](#_Toc24580722)

[3. System Architecture - overall 10](#_Toc24580723)

[3.1 Objectives 10](#_Toc24580724)

[3.2 System Organization 10](#_Toc24580725)

[A. Frontend Application 11](#_Toc24580726)

[B. Backend Application 12](#_Toc24580727)

[4. Front-end Architecture 13](#_Toc24580728)

[4.1 Sub-System 13](#_Toc24580729)

[A. User Information 13](#_Toc24580730)

[B. Search Video 18](#_Toc24580731)

[C. Information of Video 21](#_Toc24580732)

[D. Recommend Video 25](#_Toc24580733)

[5. Back-end Architecture 28](#_Toc24580734)

[5.1 Sign in 28](#_Toc24580735)

[A. Class Diagram 28](#_Toc24580736)

[B. Description 28](#_Toc24580737)

[C. Sequence Diagram 29](#_Toc24580738)

[5.2 Sign Up 30](#_Toc24580739)

[A. Class Diagram 30](#_Toc24580740)

[B. Description 30](#_Toc24580741)

[C. Sequence Diagram 31](#_Toc24580742)

[5.3 My page 32](#_Toc24580743)

[A. Class Diagram 32](#_Toc24580744)

[B. Description 32](#_Toc24580745)

[C. Sequence Diagram 33](#_Toc24580746)

[5.4 Search 34](#_Toc24580747)

[A. Class Diagram 34](#_Toc24580748)

[B. Description 34](#_Toc24580749)

[C. Sequence Diagram 35](#_Toc24580750)

[5.5 Specification 36](#_Toc24580751)

[A. Class Diagram 36](#_Toc24580752)

[B. Description 36](#_Toc24580753)

[C. Sequence Diagram 38](#_Toc24580754)

[5.6 Recommend 39](#_Toc24580755)

[A. Class Diagram 39](#_Toc24580756)

[B. Description 39](#_Toc24580757)

[C. Sequence Diagram 40](#_Toc24580758)

[5.7 Ranking 41](#_Toc24580759)

[A. Class Diagram 41](#_Toc24580760)

[B. Description 41](#_Toc24580761)

[C. Sequence Diagram 42](#_Toc24580762)

[6. Protocol Design 42](#_Toc24580763)

[6.1 JSON 42](#_Toc24580764)

[6.2 Protocol Details 43](#_Toc24580765)

[A. Sign up 43](#_Toc24580766)

[B. Sign in 43](#_Toc24580767)

[C. My Page 44](#_Toc24580768)

[D. Video ranking 44](#_Toc24580769)

[E. Search 45](#_Toc24580770)

[F. Review 45](#_Toc24580771)

[G. Recommend Video 46](#_Toc24580772)

[H. Delivery/Market information 46](#_Toc24580773)

[7. Database Design 47](#_Toc24580774)

[7.1 ER Diagram 48](#_Toc24580775)

[A. Entities 48](#_Toc24580776)

[7.3 Table 50](#_Toc24580777)

[A. User 50](#_Toc24580778)

[B. authority 50](#_Toc24580779)

[C. User\_watched 50](#_Toc24580780)

[D. Video 51](#_Toc24580781)

[E. Category 51](#_Toc24580782)

[F. video specification 51](#_Toc24580783)

[G. Delivery&purchase 51](#_Toc24580784)

[8. Testing plan 52](#_Toc24580785)

[8.1 Testing Policy 52](#_Toc24580786)

[A. Development Testing 52](#_Toc24580787)

[B. Release Testing 52](#_Toc24580788)

[C. User Testing 52](#_Toc24580789)

[8.2 Test Case 52](#_Toc24580790)

[A. User Management System 52](#_Toc24580791)

[B. Searching System 53](#_Toc24580792)

[C. Recommend System 54](#_Toc24580793)

[D. Delivery/Market System 54](#_Toc24580794)

[9.Development Plan 55](#_Toc24580795)

[9.1 Frontend Environment 55](#_Toc24580796)

[A. Android Studio 55](#_Toc24580797)

[B. JAVA 56](#_Toc24580798)

[9.3 Backend Environment 56](#_Toc24580799)

[A. Cloud Firestore 57](#_Toc24580800)

[B. Nodejs 57](#_Toc24580801)

[C. Python 58](#_Toc24580802)

[10. Index 58](#_Toc24580803)

[10.1. Figure Index 58](#_Toc24580804)

[10.2. Diagram Index 59](#_Toc24580805)

[10.3. Table Index 60](#_Toc24580806)

# 1. Preface

## 1.1 Readership

본 문서의 독자는 시스템을 개발하는 소프트웨어 엔지니어, 시스템을 설계하는 시스템 설계자, 개발에 참여하는 클라이언트 엔지니어, 고객을 위한 서비스 팀 등 시스템을 개발하는데 관련된 모든 구성원들을 포함한다. 시스템을 개발하는데 외주업체를 이용한다면, 해당 업체에서 개발에 관련된 모든 구성원 또한 독자에 포함된다.

## 1.2 Document Structure

### A. Preface

본 문서의 예상되는 독자들과 전반적인 구조, 그리고 각 부분의 역할에 대해 설명한다. 또한 버전기록도 작성한다.

### B. Introduction

Introduction에서는 시스템을 개발하는데 사용된 모든 종류의 도구와 다이어그램에 대해서 서술한다. 또한 프로젝트가 다루는 시스템의 범위를 서술한다.

### C. System Architecture

System Architecture에서는 개발하고자 하는 시스템에 대하여 전반적인 내용을 서술한다. 시스템과 각 서브시스템의 구조를 기술하고 시스템 전체 기능이 각 서브시스템과 하드웨어에 어떻게 할당되었는지 설명한다.

### D. Front-end Architecture

Front-end에서는 클라이언트와의 상호작용을 클래스 다이어그램, 상태 다이어그램을 통해 기술한다.

### E. Back-end Architecture

Back-end에서는 클라이언트에서 서버로 요청이 들어올 때 서버의 작용을 클래스 다이어그램, 시퀀스 다이어그램을 통해 기술한다. 서버는 DB, API나 다른 시스템과 상호작용하며 결과를 만들어낸다.

### F. Protocol Design

Protocol Design에서는 서브시스템이 상호작용하는 프로토콜에 대해 설명한다. 프로토콜에 필요한 기본적인 데이터 format 중 JSON에 대해 설명하고, 각 프로토콜에서 사용되는 Attribute와 Value에 대해서 설명한다.

### G. Database Design

Database Design에서는 요구사항 명세서에서 기술한 데이터베이스 요구사항을 바탕으로 이를 수정하여 다시 요구사항을 작성하였다. 요구사항을 바탕으로 ER Diagram 을 작성하고, 이를 이용하여 Relational Schema, SQL DDL을 작성한다.

### H. Testing plan

Testing Plan에서는 시스템의 test case에 대해 설명한다. 전체 시스템이 개발된 후 의도한대로 실행되는지 검증하기 위해 test case를 설정하고 계획하는데 목적이 있다. 각 서브시스템에서의 예시 케이스를 설계하고 이를 토대로 시스템의 입, 출력 동작을 설명한다.

### I. Development Environment

Development Environment 에서는 사용한 프로그래밍 언어와 IDE, 패키지 등 개발자의 환경에 대해 설명한다. 시스템을 개발하기 위해 필요한 Frontend/Backend의 개발 환경과 API에 대해서 설명한다.

### J. Development Plan

Develop Plan 에서는 개발 계획에 대해 서술한다. 시스템을 구현하는데 필요한 도구, 언어, 라이브러리 등의 개발환경에 대해 서술한다. 또한 시스템 개발 일정을 설명한다.

### K. Index

본 문서에 포함된 그림, 테이블, 다이어그램, 관련자료 등 인덱스가 포함된다.

## 1.3 Version Update History

|  |  |  |  |
| --- | --- | --- | --- |
| 버전 | 수정일자 | 변경사항 | |
| 1.0 | 2019.11.13 | 최초 작성 |
|  |  |  |
|  |  |  |
|  |  |  |

# 2. Introduction

## 2.1 Applied Diagram

### A. UML

UML은 Unified Modeling Language의 약자로 소프트웨어공학에서 프로그램의 설계를 표현하기 위해 사용하는, 주로 그림으로 된 표기법이다. 객체지향 언어를 바탕으로 설계되었기 때문에 객체지향 모델링 언어라고도 불린다. 시스템을 external, interaction, structural, behavioral 등 여러 측면에서 바라보고 각각의 관점에서 작성한 설계 표기법이다. UML 광범위한 기호와 정의를 다루므로 개발자와 사용자 간의 통신 수단을 효율적으로 사용할 수 있으며 각각의 관점에서 추상적으로 또는 구체적으로도 접근할 수 있기 때문에 유용하다. UML은 뒤에 나오는 것 같이 Context diagram, package diagram, class diagram, state diagram, sequence diagram, ER diagram 등으로 구성되며 본 문서는 UML의 표기에 기반하여 모델링 되어 있다.

### B. Class Diagram

클래스 다이어그램은 시스템의 클래스, 클래스의 속성, 동작방식, 객체 간 관계를 효과적으로 표시하는 다이어그램이다. 시스템의 구조를 기술하며 객체 지향 모델링의 주요 속성이다. 클래스 다이어그램은 UML의 가장 기초가 되며 코드를 작성할 때 용이하다. 또한 객체 간의 관계를 알 수 있기 때문에 유용하다.

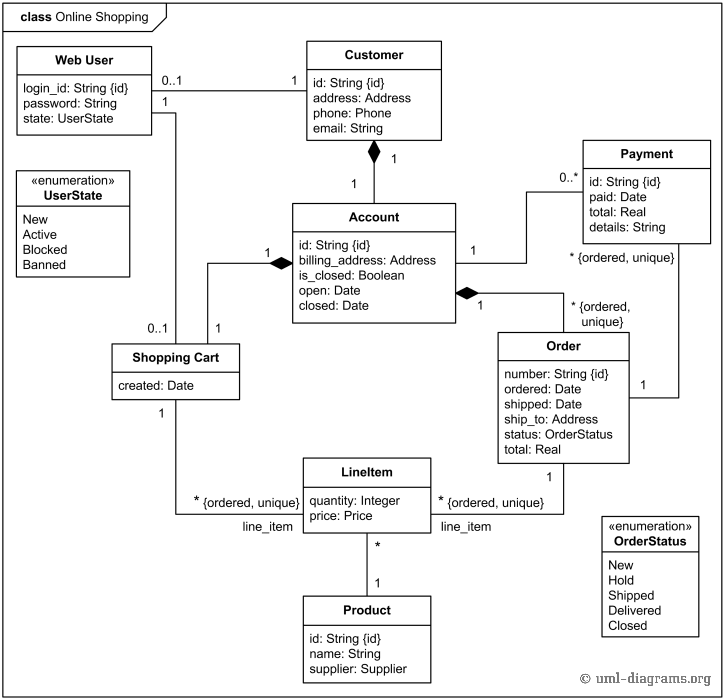


Figure . Class diagram

### C. State Diagram

State Diagram은 컴퓨터 시스템 내의 상태 변화를 보여주는 다이어그램으로 어떤 클래스에 속하는 객체의 상태와 영향주는 이벤트로 인한 상태 변이를 나타낸다. 개별 객체들의 상태들을 확인할 수 있고 각 통과하는 상태들을 발생하는 순서대로 나열하기 때문에 흐름을 알 수 있다.

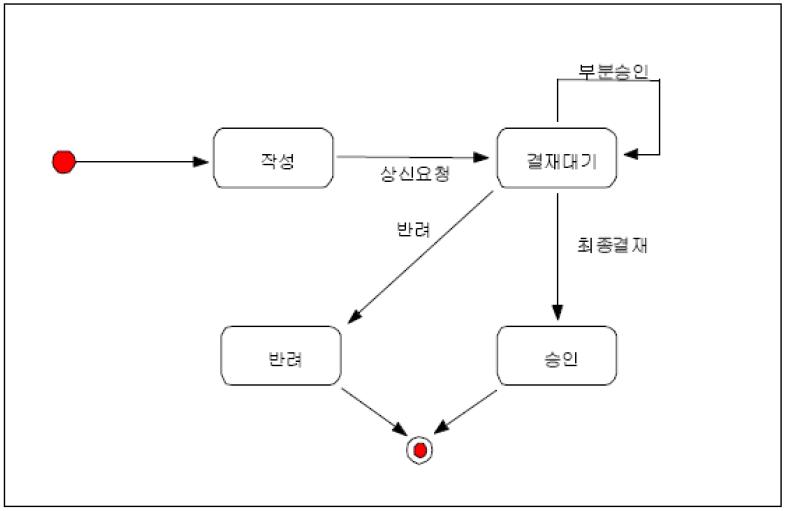


Figure . State diagram

### D. Sequence Diagram

Sequence Diagram은 시간 순서로 정렬되어 객체와 객체 사이의 상호작용을 보여주는 다이어그램이다. 보통 시나리오 등에 수행되는 객체들과 그 객체들 사이에 교환되는 메시지들, 행동등을 표현한다. 시간의 흐름순으로 나타내고 각 상호작용을 어떻게 하는지 간략하게 나와있기 때문에 전체적인 시스템 흐름을 알 수 있고 직관적이고 읽는 것도 매우 간단하다.

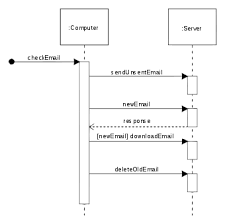


Figure . Sequence diagram

### E. ER Diagram

ER diagram은 데이터 모델링 분야에서 개체와 속성, 관계 사이의 구조화된 연결표현 방식이다. 보통은 데이터베이스에 구조 및 제약조건을 정의하는데 사용된다. ER diagram은 그래픽 형태로 표현하여 이해가 쉽고 Database를 사용하는 모델이라면 Database 종류에 종속적이지 않으면서도 사상이 가능하다.

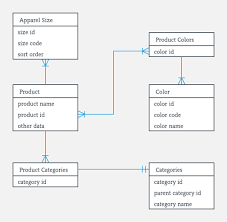


Figure . ER Diagram

## 2.2 Applied Tools

### A. draw.io

Draw.io는 인터넷 기반의, UML이나 다른 많은 다이어그램을 그릴 수 있는 사이트다. 클라우드 저장 지원이 잘 되어있고 사용법 또한 간단하기 때문에 많이 쓰이고 있다. 본 문서는 UML을 만드는데 사용된다.



Figure . draw.io

### B. ERDCloud

ERDCloud는 ER Diagram을 그리는데 특화되어 있는 웹사이트 기반의 툴이다. 간편하게 그릴 수 있으면서도 많은 기능이 있고 또한 파일을 저장하고 공유할 수 있다.



Figure . ERDCloud

### C. PowerPoint

파워포인트는 누구에게나 친숙하고 쉽게 사용 가능하다. 또한 다양한 도형을 만들 수 있으므로 UML 이외에 또는 위 툴들의 사용이 어려울 때 사용할 수 있다.



Figure . Powerpoint

### D. Web Sequence Diagrams

Web Sequence Diagrams는 웹에서 sequence diagram을 그릴 수 있는 도구이며 구글 로그인만으로 파일들을 모두 저장할 수 있게 해주고 또한 sequence diagram을 직접 그리지 않고도 코드만으로 다이어그램을 자동 생성 하도록 도와준다.

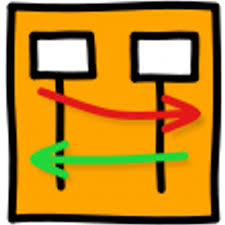


Figure . Web Seqence Diagrams

# 3. System Architecture - overall

## 3.1 Objectives

이 장에서는 시스템의 전체적인 구족의 설명하고 시스템 전체의 구조와 각 서브시스템의 대략적인 구조, 서브시스템 간의 관계를 서술한다.

## 3.2 System Organization

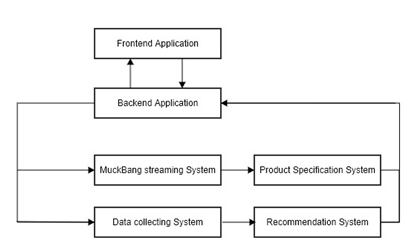


Diagram . System organization

Design-specification 의 모델로는 클라이언트-서버 모델을 적용 하였으며 Frontend client 인 안드로이드 어플리케이션에서 사용자와 상호작용을 하고, 어플리케이션과 JSON 을 이용한 HTTP request 를 통하여 데이터를 주고 받는다. 백엔드에서는 프론트엔드에서 받은 요청에 따라 데이터를 받거나, 추천시스템, 스트리밍 시스템, 먹방 아이템의 구체적 정보와 같은 것을 데이터로 송신하게 된다. 백엔드에서의 추천시스템은 먼저 동영상 스트리밍 사이트의 youtube api 를 이용하여 먹방 영상들의 데이터들을 수집한 뒤, python 으로 작성한 추천시스템을 이용하여 거기서부터 나온 output을 어플리케이션으로 보내준다. Product specification 은 정해진 식료품 사이트. 가격 비교 사이트를 이용하여 제품에 대한 가격, 판매처에 대한 정보를 제공할 수 있게 한다.

### A. Frontend Application

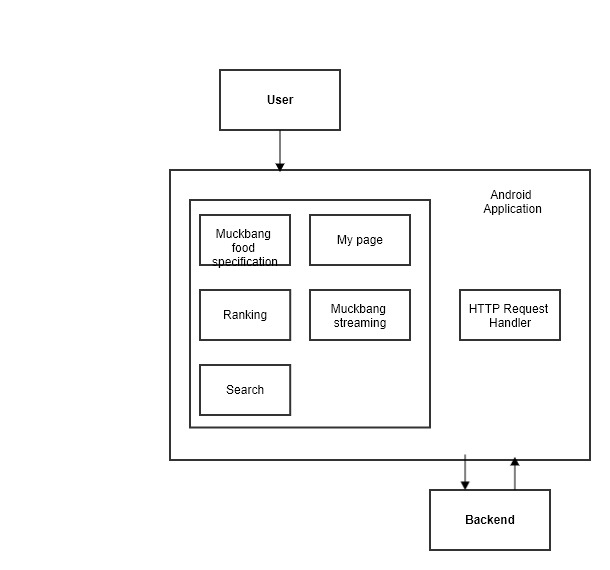


Diagram . Frontend application

Frontend 는 안드로이드 어플리케이션으로, 각각 페이지에 대한 xml 들을 작성하여 user interface를 구성하고, 페이지는 muckbang food specification , mypage, ranking, streaming, search 등의 항목이 구성되어 있다. 또한 각각의 컴포넌트에서 필요한 데이터를 http request handler 를 통해 요청하면, Request handler 가 미리 정해진 REST api 를 통하여 백엔드에게 정보를 주고, 받을 수 있게 된다. 데이터를 받으면 각각 컴포넌트에 해당 데이터를 제공하여 사용자에게 정보, 영상 등을 제공할 수 있게 한다.

### B. Backend Application

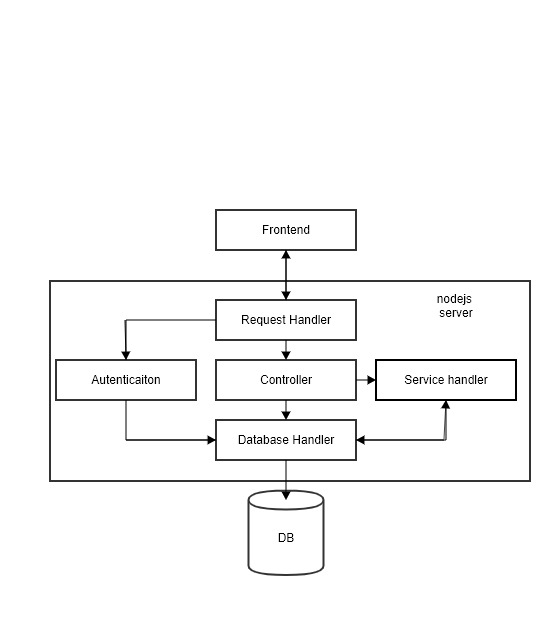


Diagram . Backend Application

백엔드 에서는 먼저 프론트에서 온 HTTP request 를 받아 어떠한 요청을 원하는 지 확인하게 된다. Authentication에서 권한이 있는지 확인 해주고, 컨트롤러를 통해 어떠한 service 에 대한 데이터를 원하는지 판단한다. 해당 서비스에 대한 핸들링을 해주고, 필요한 데이터인 경우 database handler 를 통하여 DB에 저장하고, 원하는 아웃풋을 다시 프론트엔드에 전달하게 된다. 예를 들어 프론트엔드에서 추천영상에 대한 정보를 요청하는 http request 가 들어오면, recommendation service handler 에서 받은 데이터를 기반으로 추천 시스템 알고리즘을 통해 그에 맞는 DB에 있는 영상을 제공해주고 요청 받은 정보도 저장하게 된다.

# 4. Front-end Architecture

## 4.1 Sub-System

### A. User Information

**Class Diagram**

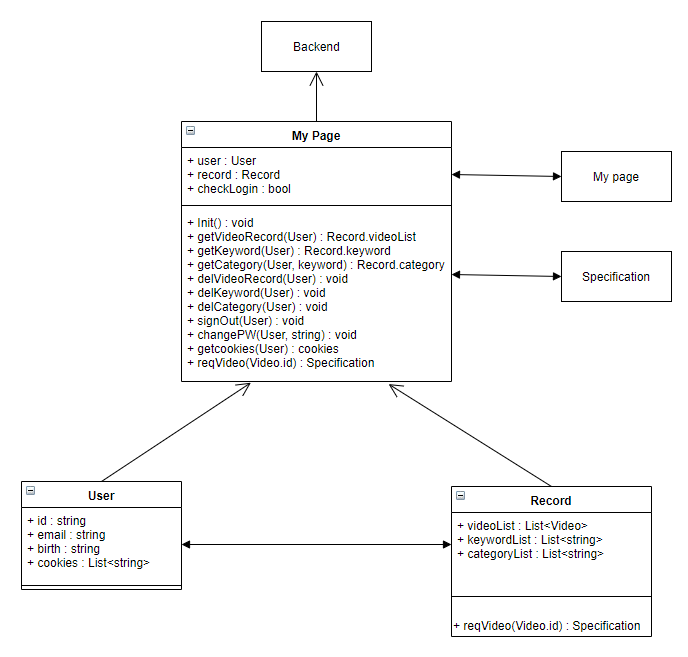


Diagram . user information class diagram

**Description**

(a) MyPage

MyPage는 기본적으로 어플리케이션 하나당 하나의 객체만 생성된다. MyPage는 하나의 유저와 하나의 기록을 가진다. 어플리케이션이 처음 시작할 때 이 객체는 만들어진다.

(1) Attributes

+ user : User : User class를 가진다.

+ record : Record : Record Class를 가진다.

+ checkLogin : bool : Login이 되어있는 상태인지 확인한다.

(2) Methods

+ Init() : void : 프로그램을 처음 시작할 때 실행된다. 쿠키를 가져와 로그인을 시도하고 로그인에 성공했다면 기록과 유저정보를 가져오고 그렇지 않다면 checkLogin이 false인 상태로 언제든 로그인을 시도할 수 있게 한다.

+ getVideoRecord(User) : Record.videoList : 사용자 기록 중 시청기록을 가져온다. 유저의 정보를 인자로 받아서 백엔드로 요청을 보내고 각 유저에 맞는 시청기록을 가져온다.

+ getKeyword(User) : Record.keyword : 사용자 기록 중 키워드를 가져온다. 유저의 정보를 인자로 받아서 백엔드로 요청을 보내고 각 유저에 맞는 키워드리스트를 가져온다.

+ getCategory(User, keyword) : Record.category : 사용자 기록 중 카테고리 리스트를 가져온다. 유저의 정보를 인자로 받아서 백엔드로 요청을 보내고 각 유저에 맞는 카테고리를 가져온다.

+ delVideoRecord(User) : void : 사용자 시청기록을 모두 지운다. 백엔드로 요청을 보내 그에 맞는 기록을 모두 지우고 getVideoRecord를 호출한다.

+ delKeyword(User) : void : 마찬가지로 키워드를 모두 지운다.

+ delCategory(User) : void : 마찬가지로 카테고리를 모두 지운다.

+ signOut(User) : void : 이 메소드를 호출하면 서버에 로그아웃을 하는 것을 알리고 User와 Record를 모두 지운다. checkLogin도 false로 만든다.

+ changePW(User, string) : void : PW를 변경할 때 사용한다. 서버에 변경할 비밀번호를 요청하면 서버는 user가 맞는지 확인 후 요청한 비밀번호로 변경한다.

+ getcookies(User) : cookies : user를 식별하기 위한 cookie를 가져온다.

+ reqVideo(Video.id) : Specification : 비디오 목록 중 어떤 하나를 클릭했을 때 그 비디오 고유id에 맞는 비디오 정보를 가져온다. Specification은 뒤에서 기술한다.

(B) User

User는 로그인할 때마다 새로 생성을 하고 동시에 한 객체밖에 존재할 수 없다. User는 사용자에 대한 정보를 저장하고 만약 정보가 더 필요하다면 추후에 속성을 추가할 수 있다.

(1) Attributes

+ id : string : 유저의 ID를 저장한다.

+ email : string : 유저의 email을 저장한다. 이메일은 나중에 비밀번호 찾을 때 등에 사용될 수 있다.

+ birth : string : 생년월일을 저장한다. 생년월일은 연령제한이나 비밀번호 찾기에 사용될 수 있다.

+ cookies : List<string> : 유저에 관련한 쿠키를 저장한다.

(C) Record

Record class는 user와 분리를 해 놨지만 실제로는 서로 연관되어 같이 쓰이는 클래스이다. 유저에 맞는 기록이 나오도록 1대1 대응을 해야 하며 유저가 로그인, 로그아웃 시에는 user와 같이 소멸되어야 한다. 하지만 시청기록 같은 부분은 계속해서 최신정보를 가져온다.

(1) Attributes

+ videoList : List<Video> : 유저에 대한 비디오 시청기록을 가지고 있는 리스트이다. Video 클래스는 뒤에서 다룬다.

+ keywordList : List<string> : 키워드에 대한 목록을 가지고 있는 리스트이다. 유저가 검색한 기록 등에서 가져온다. 추천영상 등에 사용된다.

+ categoryList : List<string> : 카테고리목록이다. 유저가 관심있어 하는 카테고리를 저장해 나중에 추천영상 등에 사용된다.

**State Diagram**

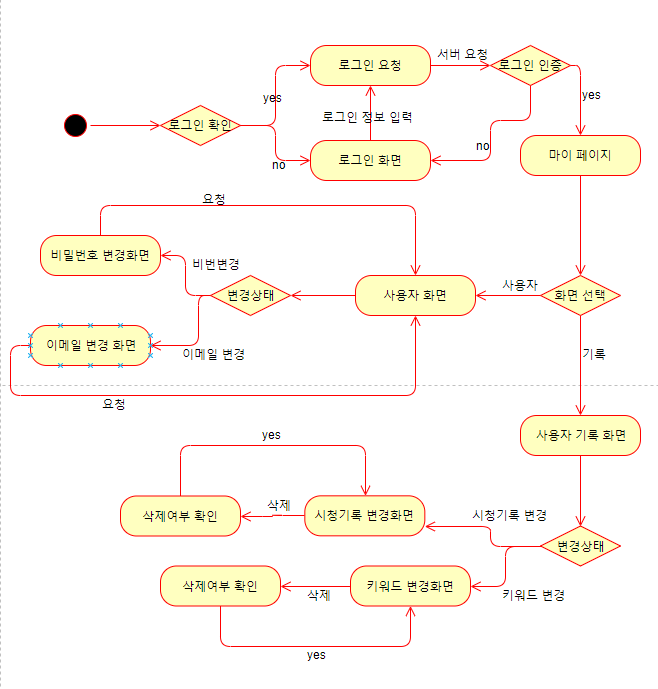


Diagram . user information state diagram

### B. Search Video

**Class Diagram**

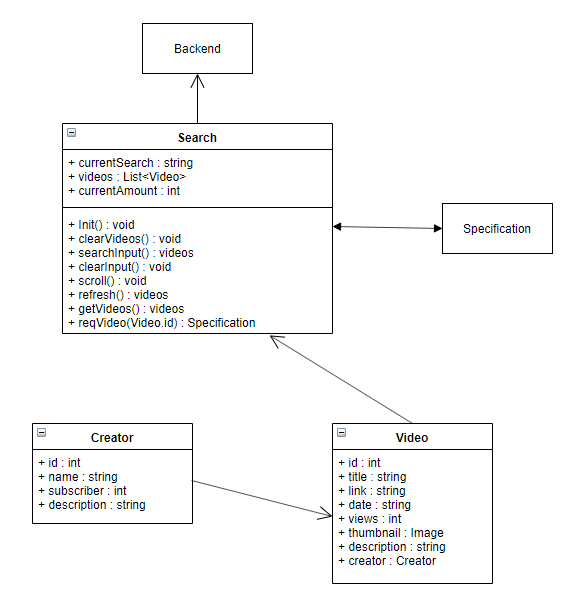


Diagram . search class diagram

**Description**

(a) Search

Search는 검색 버튼을 클릭 했을 때 작동하는 클래스이다. 기본적으로 하나의 객체만 생성되어 검색어를 입력, 요청 등의 연산을 수행하며 비디오 목록 뷰를 만들어준다. 비디오를 클릭하면 비디오 화면으로 넘어가게 한다.

(1) Attributes

+ currentSearch : string : 현재 검색어를 표현한다. 현재검색어가 계속 입력창에 남아있게 한다.

+ videos : List<Video> : Video 클래스의 리스트이다. 이는 검색 결과인 비디오 목록들을 표현할 때 사용된다.

+ currentAmount : int : 현재 창에 보이는 비디오의 개수를 표시한다. 처음 뷰에 비디오 목록을 모두 다 표시하는 것이 아니라 스크롤 할 때마다 리스트를 동적으로 생성하게 한다.

(2) Methods

+ Init() : void : 생성자이다.

+ setCurrentSearch : 현재 검색어를 입력한다.

+ clearVideos() : void : 단순히 비디오 목록을 초기화 한다.

+ searchInput() : videos: 검색하는 기능이다. 현재 검색어를 자동으로 인자로 주어 서버로 요청한다. 서버는 비디오 목록을 반환한다.

+ clearInput() : void : 현재 검색어를 초기화 한다.

+ scroll() : void : 사용자가 스크롤을 내리다보면 이 메소드가 호출되어 화면에 비디오리스트를 더 만들어주고 currentAmount를 늘려준다.

+ refresh() : videos : 비디오를 새로고침 할 때 다시 서버로 똑같은 검색어를 호출한다.

+ getVideos() : videos : Video목록을 가져올 때 사용된다.

+ reqVideo(Video.id) : Specification : 비디오 목록 중 어떤 하나를 클릭했을 때 그 비디오 고유id에 맞는 비디오 정보를 가져온다. Specification은 뒤에서 기술한다.

(b) Video

비디오에 대한 간단한 정보를 표시한다. 객체 하나 당 하나의 비디오를 나타내며 비디오 목록 뷰에 사용된다.

(1) Attributes

+ id : int : 비디오의 고유 id이다. 비디오의 id만으로 서버와 통신할 수 있다.

+ title : string : 비디오의 제목이다.

+ link : string : 비디오 링크이다. 거의 사용되지 않는다.

+ date : string : 비디오 업로드 날짜이다.

+ views : int : 비디오 조회수이다.

+ thumbnail : Image : 비디오 썸네일로 비디오 목록에서 동영상 이미지가 나오는데 그 때 사용된다.

+ description : string : 비디오에 대한 간략한 설명을 나타낸다.

+ creator : Creator : 비디오를 만든 크리에이터에 대한 정보를 객체로 가진다.

(c) Creator

비디오를 만든 크리에이터에 대한 정보이다. 크리에이터의 이름, 구독자 수 등을 속성으로 갖는다.

(1) Attributes

+ id : int : 크리에이터의 고유 id이다.

+ name : string : 크리에이터의 이름이다.

+ subscriber : int : 크리에이터의 구독자 수이다.

+ description : string : 크리에이터에 대한 설명이다.

**State Diagram**

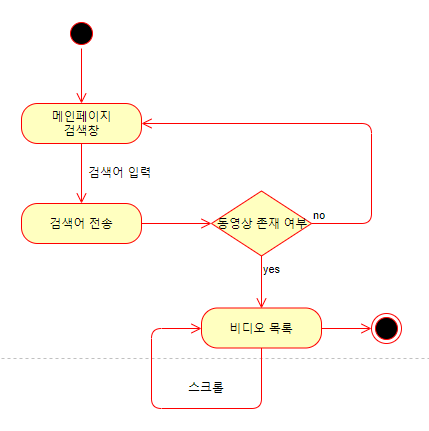


Diagram . search state diagram

### C. Information of Video

**Class Diagram**

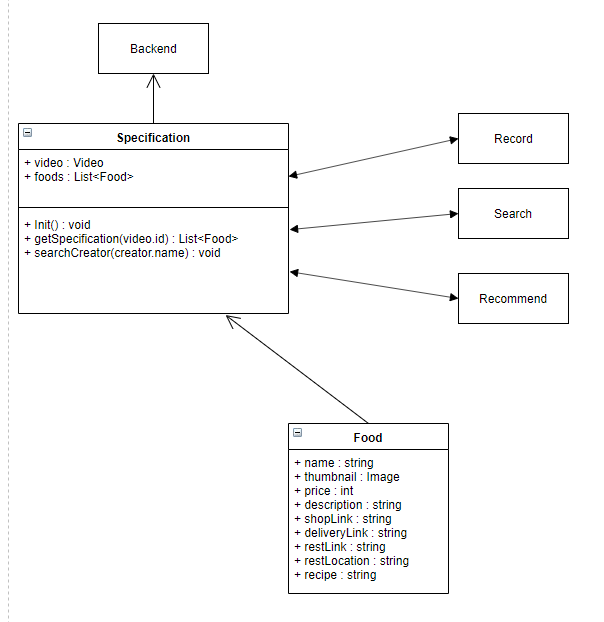


Diagram . specification class diagram

**Description**

(a) Specification

현재 클릭한 비디오에 대한 비디오 상세정보를 가져온다. 상세정보란 레시피나 구매경로, 음식정보 등을 포함한다. 현재 실행중인 비디오에 대한 정보이므로 기본적으로 한번에 하나의 객체만 존재한다. 그리고 Search나 Recommendation 같은 비디오 목록을 보여주는 객체에서 비디오를 클릭한다면 이 객체와 상호작용을 하게 된다.

또한 보여주는 정보는 많지만 대부분이 링크나 텍스트이기 때문에 많은 함수가 필요하지 않다.

(1) Attributes

+ video : Video : 현재 진행되는 비디오의 기본정보를 표시한다.

+ foods : List<Food> : 현재 진행되는 비디오의 음식리스트를 표현한다. Food Class는 그 다음에 설명한다.

(2) Methods

+ Init() : void : 생성자이다.

+ getSpecification(video.id) : List<Food> : 비디오의 id를 가지고 그 비디오와 연관된 food 리스트들을 서버에서 가져온다.

+ searchCreator(creator.name) : void : 크리에이터에 대한 정보를 검색할 수 있다. Search 클래스와 상호작용 하여 크리에이터의 이름으로 검색할 수 있게 해준다.

(b) Food

비디오 상세정보를 표시하는데 사용되는 객체이다. 기본적으로 먹방에 대한 정보는 먹방에 사용되는 음식의 정보들과 비디오, 크리에이터에 대한 정보의 집합이다. 음식에 대한 개개 독립적인 상세정보를 객체로 만들어 제공하는 것이 바람직하다.

(1) Attributes

+ name : string : 음식의 이름이다.

+ thumbnail : Image : 음식에 대한 사진이다.

+ price : int : 음식에 대한 가격이다. 음식에 대한 가격은 계속 변하기 때문에 추후에 없앨 수 있다.

+ description : string : 음식에 대한 간단한 설명이다.

+ shopLink : string : 음식에 대한 쇼핑몰 링크이다. 추후에 링크를 리스트로 만들어 여러 개로 확장할 수 있다.

+ deliveryLink : string : 마찬가지로 배달 링크이다.

+ restLink : string : 음식이 있는 레스토랑의 사이트 링크이다.

+ restLocation : string : 레스토랑이 위치한 좌표를 저장하고 있다가 나중에 맵으로 연동시킬 수 있다.

+ recipe : string : 음식을 만드는 레시피를 저장한다. string으로 저장해도 나중에 좋은 방안이 있다면 좋은 폼을 만들 수 있다.

**State Diagram**

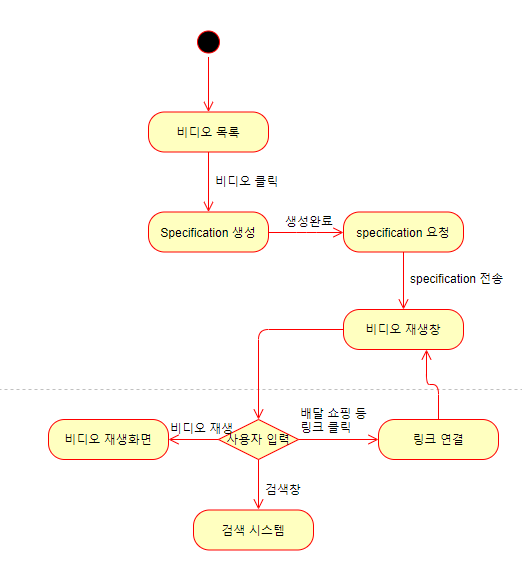


Diagram . specification state diagram

### D. Recommend Video

**Class Diagram**

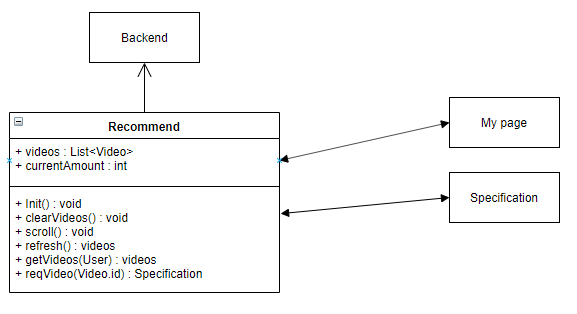


Diagram . Recommend Video class diagram

**Description**

(a) Recommend

Recommend는 Recommend 창을 눌렀을 때만 사용되는 클래스이다. 검색 클래스와 비슷하다. 하지만 프론트엔드 단에서는 딱히 하는게 없고 요청만 보내고 동영상을 받아서 띄우기만 한다.

(1) Attributes

+ videos : List<Video> : Video 클래스의 리스트이다. 추천 받은 동영상들의 리스트를 가진다.

+ currentAmount : int : 현재 창에 보이는 비디오의 개수를 표시한다. 처음 뷰에 비디오 목록을 모두 다 표시하는 것이 아니라 스크롤 할 때마다 리스트를 동적으로 생성하게 한다.

(2) Methods

+ Init() : void : 생성자이다.

+ clearVideos() : void : 단순히 비디오 목록을 초기화 한다.

+ scroll() : void : 사용자가 스크롤을 내리다보면 이 메소드가 호출되어 화면에 비디오리스트를 더 만들어주고 currentAmount를 늘려준다.

+ refresh() : videos : 비디오를 새로고침 할 때 다시 서버로 똑같은 검색어를 호출한다.

+ getVideos(User) : videos : Video목록을 가져올 때 사용된다. 사용자별 추천을 받기 위해 Mypage의 유저정보를 인자로 제공해야 한다.

+ reqVideo(Video.id) : Specification : 비디오 목록 중 어떤 하나를 클릭했을 때 그 비디오 고유id에 맞는 비디오 정보를 가져온다.

**State Diagram**

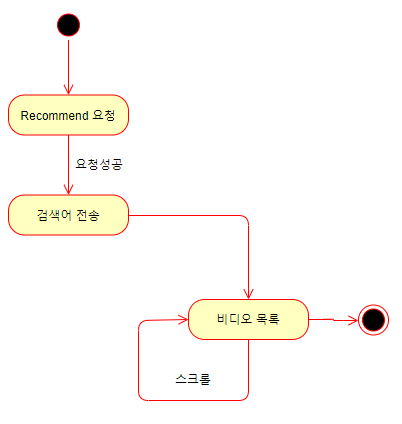


Diagram . Recommend State diagram

# 5. Back-end Architecture

## 5.1 Sign in

### A. Class Diagram

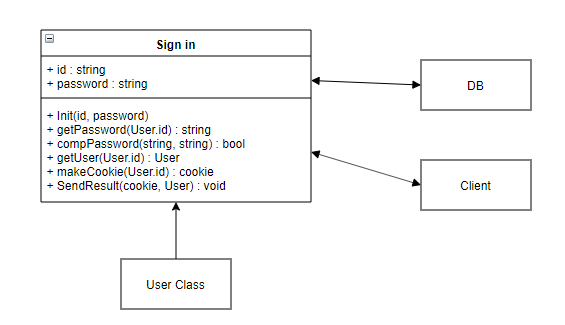


Diagram . Backend Sign in class diagram

### B. Description

Sign In은 로그인 요청이 올 때마다 새로 만들어져 반응하는 객체이다. id와 passward만이 요청인자로 주어지고 인증 후 쿠키를 만들어 준다.

+ Init(id, password) : 생성자이다.

+ getPassword(User.id) : string : 아이디에 해당하는 패스워드를 데이터베이스로부터 가져온다.

+ compPassword(string, string) : bool : 데이터베이스로부터 가져온 패스워드와 입력받은 패스워드를 비교한다.

+ getUser(User.id) : User : user의 아이디로 user에 관한 정보를 모두 가져온다.

+ makeCookie(User.id) : cookie : user가 로그인한 것을 인증하는 쿠키를 만든다. 쿠키는 토큰을 사용하여 server에서 보관한다.

+ SendResult(cookie, User) : void : 유저에게 로그인 성공했는지 여부를 반환한다. 성공했다면 쿠키와 유저정보를 준다.

### C. Sequence Diagram

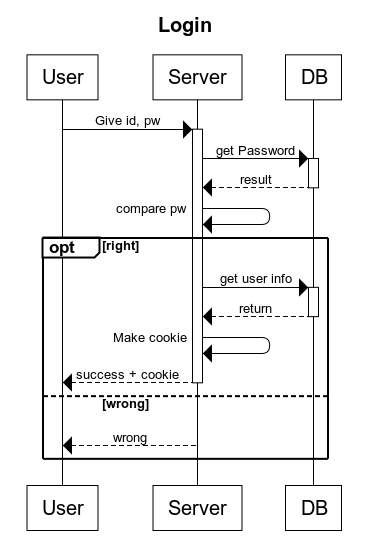


Diagram . Backend Sign in Sequence diagram

## 5.2 Sign Up

### A. Class Diagram

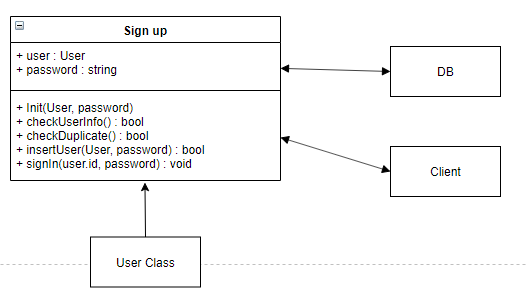


Diagram . Backend Sign up class diagram

### B. Description

Sign Up은 회원가입 요청이 오면 DB에 유저정보를 저장한다. 입력 폼이 요청으로 주어지면 객체를 새로 생성해 회원가입 후 소멸한다.

+ Init(User, password) : 생성자이다.

+ checkUserInfo() : bool : User의 입력이 적절한 입력인지 검사한다. 가령 빈칸이라던지 잘못된 언어 등을 사용하면 false를 반환한다.

+ checkDuplicate() : bool : db의 다른 유저정보와 중복이 있는지 확인한다. bool타입이다.

+ insertUser(User, password) : bool : db에 유저정보를 저장한다.

+ signIn(user.id, password) : void : 회원가입을 한다면 바로 로그인이 된 상태로 이용하기 위해 회원가입 성공 후 바로 signIn을 호출한다. SignIn 시스템에서 하는 행동을 비슷하게 따라한다.

### C. Sequence Diagram

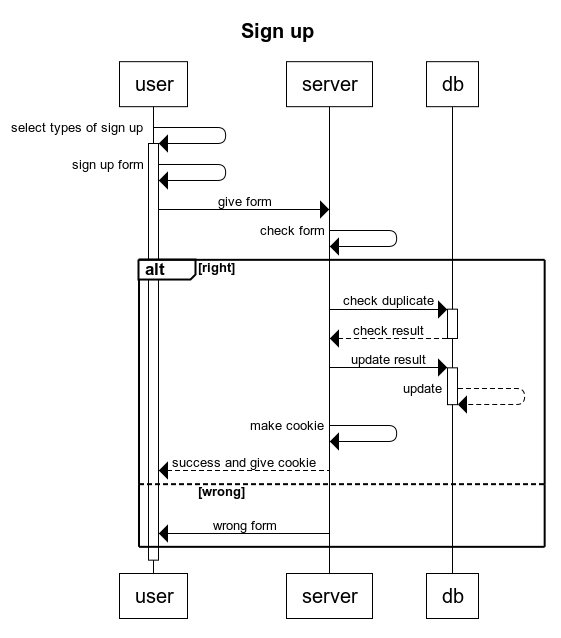


Diagram . Backend sign up sequence diagram

## 5.3 My page

### A. Class Diagram

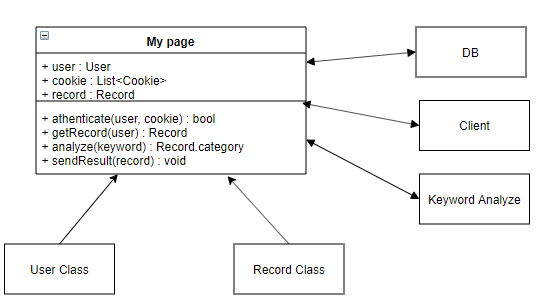


Diagram . Backend My page class diagram

### B. Description

(3) My Page

My page는 마이페이지 버튼을 사용자가 눌렀을 때 1번 생성되서 소멸하는 객체이다. 기본적인 인증과 유저와 관련한 모든 정보를 가져온다. 키워드를 통해 카테고리를 분석하여 제공한다.

+ athenticate(user, cookie) : bool : 사용자가 일치하는지 확인한다.

+ getRecord(user) : Record : 사용자와 관련한 기록들을 가져온다.

+ analyze(keyword) : Record.category : 키워드 분석 시스템을 사용하여 카테고리를 만든다. 카테고리범주는 사전에 정의되어 있고 인공지능을 통해 키워드와 관련있는 카테고리를 반환한다.

+ sendResult(record) : void : 사용자에게 정보를 제공한다.

### C. Sequence Diagram

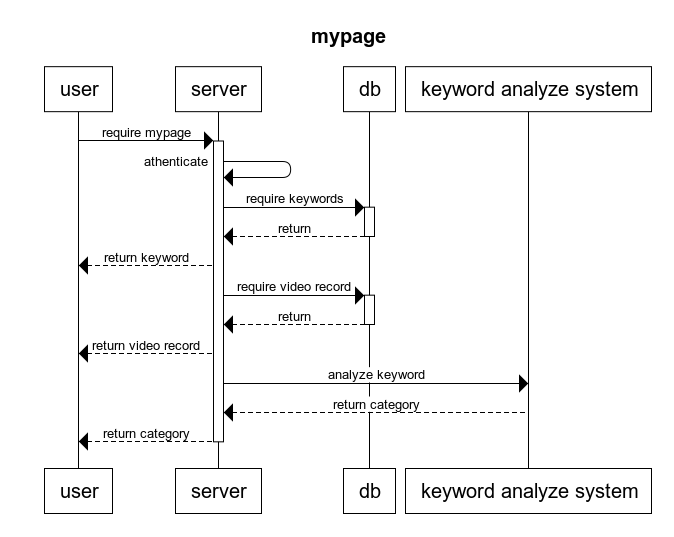


Diagram . Backend my page sequence diagram

## 5.4 Search

### A. Class Diagram

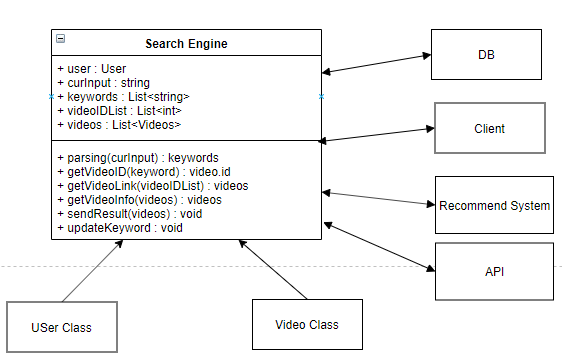


Diagram . Backend Search class diagram

### B. Description

Search engine은 검색기능과 함께 검색한 기록까지 남겨놓는다. 따라서 유저는 검색한 내용이 많을 수록 보다 정확한 동영상추천을 받을 수 있다.

+ parsing(curInput) : 입력받은 검색어를 정제한다. 불필요한 공백이나 이상한 문자는 없애고 의미가 있는 키워드들로 파싱한다.

+ getVideoID(keyword) : video.id : 키워드를 추천시스템에 주어 키워드와 밀접한 관련이 있는 비디오들의 id를 전송한다.

+ getVideoLink(videoIDList) : videos : video의 id만으로 db에서 비디오의 링크들을 얻을 수 있다.

+ getVideoInfo(videos) : videos : 링크들을 API로 보내 그 비디오와 관련된 비디오 정보들을 얻는다.

+ sendResult(videos) : void : 비디오정보들을 user에게 보낸다.

+ updateKeyword : void : 키워드 기록을 저장한다.

### C. Sequence Diagram

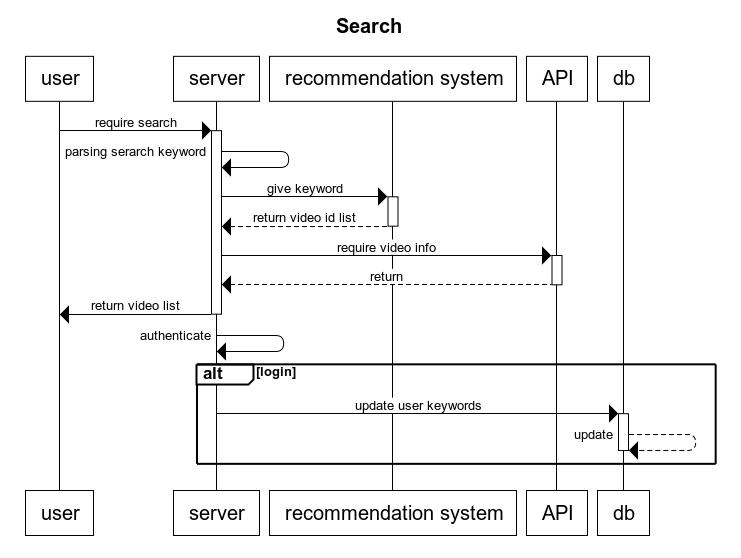


Diagram . Backend Search sequence diagram

## 5.5 Specification

### A. Class Diagram

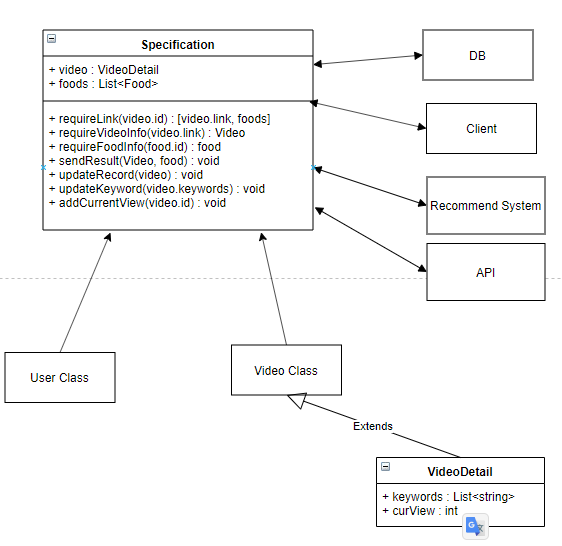


Diagram . Backend Specification class diagram

### B. Description

Specification은 동영상id를 요청받았을 때 그 동영상과 관련된 모든 정보를 제공하는 것이다. 또한 시청기록 등에 영향을 준다.

VideoDetail은 Video의 class와 비슷하지만 비디오와 관련된 키워드들, 전체 조회수를 더 가진다. 전체 조회수는 전체 유저들의 총 조회수이다. 이는 랭킹에 사용된다.

+ requireLink(video.id) : [video.link, foods] : video id를 가지고 db를 통해 링크와 id와 관련된 food 리스트를 얻어온다.

+ requireVideoInfo(video.link) : Video : 비디오 링크를 API에게 주어 비디오 정보를 받는다.

+ requireFoodInfo(food.id) : food : food id를 주어 food들에 대한 정보들을 가져온다.

+ sendResult(Video, food) : void : food와 비디오 관련정보를 준다.

+ updateRecord(video) : void : 동영상을 클릭한것이기 때문에 시청기록에 이 동영상을 저장한다.

+ updateKeyword(video.keywords) : void : keyword도 이 동영상과 관련한 키워드를 저장한다.

+ addCurrentView(video.id) : void : 이 동영상의 조회수를 1 추가한다.

### C. Sequence Diagram

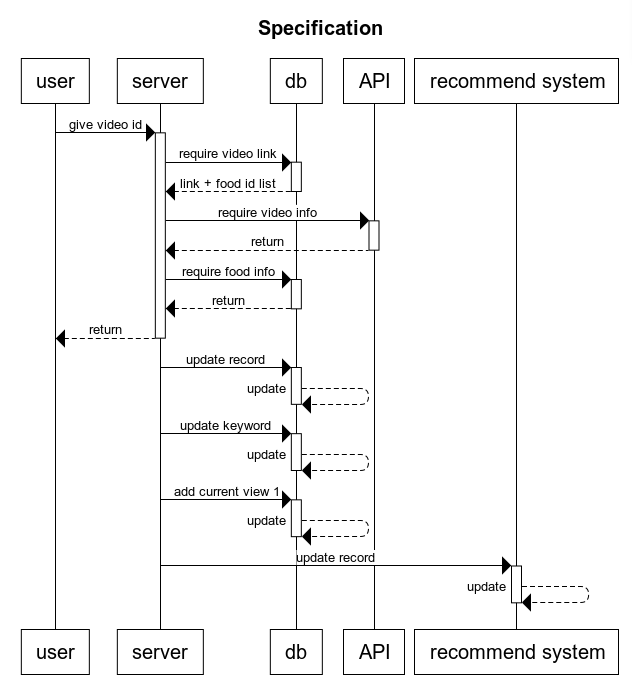


Diagram . Backend Specification sequence diagram

## 5.6 Recommend

### A. Class Diagram

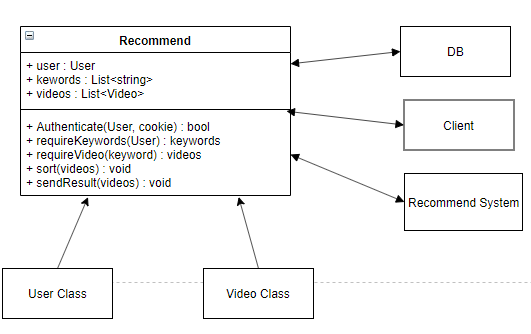


Diagram . Backend Recommend class diagram

### B. Description

Recommend는 user마다 personalized된 추천 동영상을 제공한다. 가중치나 분류같은 것들은 Recommend system을 사용하는데 이는 파이썬 오픈소스를 사용한다.

+ Authenticate(User, cookie) : bool : 유저가 맞는지 인증한다.

+ requireKeywords(User) : keywords : user와 관련한 키워드기록을 요청한다.

+ requireVideo(keyword) : videos : 키워드를 주어 키워드와 관련된 비디오를 받는다.

+ sort(videos) : void : 비디오를 모두 받으면 비디오를 가중치에 따라 정렬한다.

+ sendResult(videos) : void : 정렬된 비디오를 유저에게 제공한다.

### C. Sequence Diagram

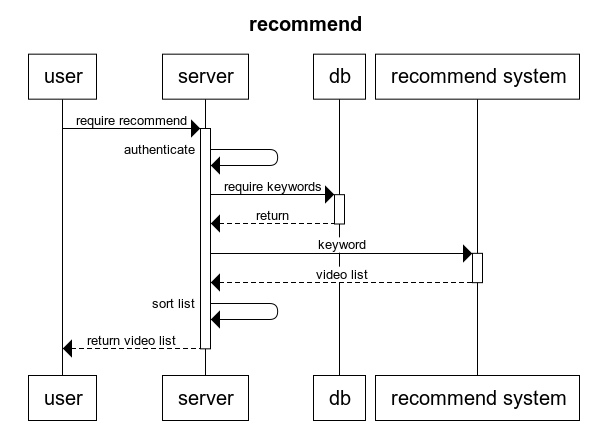


Diagram . Backend recommend sequence diagram

## 5.7 Ranking

### A. Class Diagram

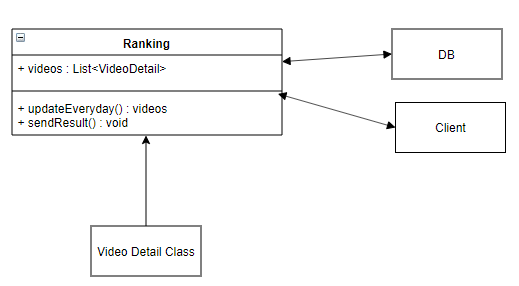


Diagram . backend Ranking class diagram

### B. Description

Ranking은 하루마다 한번씩 db로부터 총 조회수 top 100인 데이터를 제공받는다. 그 100가지 동영상의 정보를 서버에서 저장하고 있다가 클라이언트가 요청시 바로 제공한다.

+ updateEveryday() : videos : 매일마다 그 날동안 조회수가 높은 동영상들을 데이터베이스로부터 추출한다.

+ sendResult() : void : 클라이언트가 비디오를 요청했을 때 바로 보내준다.

### C. Sequence Diagram

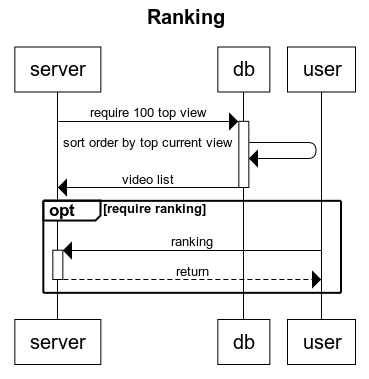


Diagram . Backend Ranking sequence diagram

# 6. Protocol Design

## 6.1 JSON

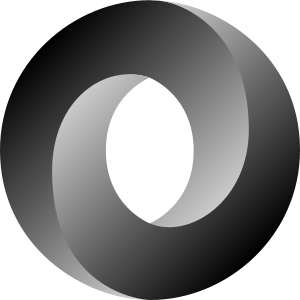


Figure . JSON

JSON(JavaScript Object Notation)은 웹과 컴퓨터 프로그램에서 용량이 적은 데이터를 교환하기 위해 데이터 객체를 속성, 값의 쌍 형태로 표현하는 형식이다. JavaScript 토대로 개발되었으며, 여러 프로그래밍 언어에도 사용할 수 있어 독립형 언어이고 텍스트로 기술하여 사람도 쉽게 읽고 작성할 수 있다. 웹 브라우저와 웹 서버 간 비동기 통신, 웹 서버 간의 데이터 교환 등에 주로 사용된다. Firefox, Internet Explorer, Chrome 등 대부분 최신 웹 브라우저는 JSON 전용 parser 기능을 내장하고 있으므로 안전하고 빠른 방법이다.

## 6.2 Protocol Details

### A. Sign up

**A.1 Request**

|  |  |
| --- | --- |
| Attribute | Value |
| UserID | 사용자의 아이디 |
| Password | 사용자의 비밀번호 |
| EMail | 사용자의 이메일 |

Table . sign up request

**A.2 Response**

|  |  |
| --- | --- |
| Attribute | Value |
| Signup\_Success | 회원가입 성공 여부 |

Table . sign up response

### B. Sign in

**B.1 Request**

|  |  |
| --- | --- |
| Attribute | Value |
| UserID | 사용자의 아이디 |
| Password | 사용자의 비밀번호 |

Table . Sign in request

#### B.2 Response

|  |  |
| --- | --- |
| Attribute | Value |
| Login\_Success | 로그인 성공여부 |

Table . Sign in response

### C. My Page

**C.1 Request**

|  |  |
| --- | --- |
| Attribute | Value |
| UserID | 사용자의 아이디 |

Table . my page request

**C.2 Response**

|  |  |
| --- | --- |
| Attribute | Value |
| UserID | 사용자의 아이디 |
| Email | 사용자의 이메일 |
| Watched\_Video | 사용자가 시청한 동영상 |
| Saved\_Video | 사용자가 저장한 동영상 |
| Subscribe\_Creator | 사용자가 구독한 크리에이터 |

Table . my page response

### D. Video ranking

**D.1 Request**

|  |  |
| --- | --- |
| Attribute | Value |
| Video\_SN | 동영상의 식별번호 |
| Video\_Info | 동영상 정보 |

Table . ranking request

**D.2 Response**

|  |  |
| --- | --- |
| Attribute | Value |
| Video\_Rank | 동영상 조회수 순위 |

Table . ranking response

### E. Search

**E.1 Request**

|  |  |
| --- | --- |
| Attribute | Value |
| Search\_word | 검색할 단어 |

Table . Search request

**E.2 Response**

|  |  |
| --- | --- |
| Attribute | Value |
| Video\_SN | 동영상의 식별번호 |
| Video\_Keyword | 동영상의 키워드 |
| Video\_Info | 동영상 정보 |

Table . search response

### F. Review

**F.1 Request**

|  |  |
| --- | --- |
| Attribute | Value |
| UserID | 사용자의 아이디 |
| Review\_Title | 리뷰 제목 |
| Review\_con | 리뷰 내용 |
| User\_Password | 사용자의 비밀번호 |

Table . review request

**F.2 Response**

|  |  |
| --- | --- |
| Attribute | Value |
| Review\_Success | 리뷰 등록 성공여부 |

Table . review response

### G. Recommend Video

**G.1 Request**

|  |  |
| --- | --- |
| Attribute | Value |
| UserID | 사용자의 아이디 |
| Watched\_Video | 사용자가 시청한 동영상 |
| Subscribe\_Creator | 사용자가 구독한 크리에이터 |
| Video\_Info | 동영상 정보 |

Table . recommend request

**G.2 Response**

|  |  |
| --- | --- |
| Attribute | Value |
| Recommend\_Video | 유저 맞춤 추천 동영상 |

Table . recommend response

### H. Delivery/Market information

**H.1 Request**

|  |  |
| --- | --- |
| Attribute | Value |
| Video\_Info | 동영상 정보 |

Table . shopping info request

**H.2 Response**

|  |  |
| --- | --- |
| Attribute | Value |
| Delivery\_Info | 배달 정보 |
| Market\_Info | 인터넷 쇼핑몰 정보 |

Table . shopping info response

# 7. Database Design

Require specification 에서 작성한 데이터 베이스를 바탕으로 세부적인 데이터베이스 설계에 대한 것을 기술하고, 이 때 diagram 은 ER Diagram 을 이용하여 기술한다. 이 장에서는 전체적인 Entity 간의 관계를 기술하고, 구글에서 개발한 baas인 Cloud firebase를 활용하여 DB를 관리하고 설계하였다.

.

## 7.1 ER Diagram

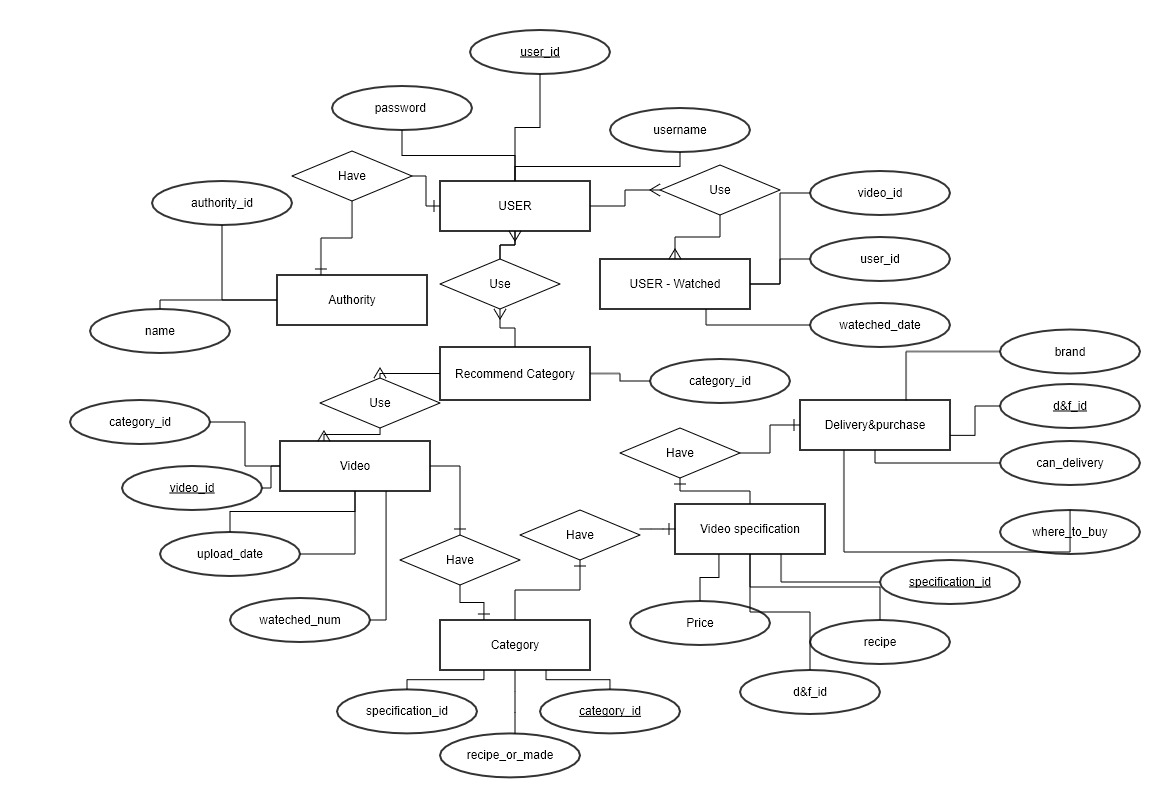


Diagram . ER diagram

ER diagram 으로 각각 entity 간의 관계와 해당 요소들을 기술하였고 본 프로젝트에서는 colud firebase 이용하기 때문에 개발 에서는 해당 포함하는 entity 들에 foreign key 개념이 아니라 하위 폴더에 저장하는 개념으로 사용 된다.

### A. Entities

a. User

User 에서는 유저 id , 유저 이름 (닉네임), password 정보를 가지고 있다.

b. Authority

Authority 에서는 firebase authority 기능에 구현되어 있는 유저마다 갖는 고유의 authority\_id와 이름의 정보를 가지고 있다.

c. User-Watched

User-Watched에서는 유저가 본 동영상에 대한 비디오 아이디, 그리고 시청한 유저의 아이디, 시청한 날짜에 대한 정보를 가지고있다.

d. Recommend Category

이 부분에서는 실제 backend 에서 front로 보내줄 추천 영상에 대한 카테고리들의 정보들이 담겨있다.

e. Video

video 에서는 video 의 category id, video id, 조회수, 업로드 날짜에 대한 정보들이 담겨있다.

f. Category

Category 에서는 비디오의 카테고리에 대한 정보들이 담겨있다. 하위 설명에 대한 키이름인 specification id 와 음식이 만들어서 판매되는 시제품인지, 아니면 레시피가 있는지에 대한 recipe\_or\_made, 그리고 카테고리 아이디가 포함되어있다.

g. Video specification

이 부분에서는 비디오의 음식에 대한 정보가 담겨있다. 가격, 레시피, 그리고 d&f id 가 담겨있는데 이는 배달음식인지 아닌 지에 대한 정보가 담겨있다.

h. Delivery&purchase

이 부분에서는 배달가능 여부 인 can\_delivery, 판매처인 where\_to\_buy, 그리고 d&f\_id 와 브랜드에 관한 정보가 담겨있다.

## 7.3 Table

본 시스템에서는 cloud firestore 를 이용 하는데 거기에 들어가는 DB 의 테이블에 대한 필드에 관한 정보를 기술하였다.

### A. User

|  |  |
| --- | --- |
| Name | Type |
| user\_id | Text String |
| Username | Text String |
| Password | Text String |

Table . user db table

### B. authority

|  |  |
| --- | --- |
| Name | Type |
| authority\_id | Text String |
| Name | Text String |

Table . autority db table

### C. User\_watched

|  |  |
| --- | --- |
| Name | Type |
| video\_id | Text String |
| user\_id | Text String |
| watched\_date | Text String |

Table . user\_watched db table

### D. Video

|  |  |
| --- | --- |
| Name | Type |
| category\_id | Text String |
| video\_id | Text String |
| upload\_date | Text String |
| watched\_num | Integer |

Table . video db table

### E. Category

|  |  |
| --- | --- |
| Name | Type |
| category\_id | Text String |
| specification\_id | Text String |
| recipe\_or\_made | Integer |

Table . category db table

### F. video specification

|  |  |
| --- | --- |
| Name | Type |
| specification\_id | Text String |
| price | Text String |
| d&f\_id | Text String |
| Recipe | Text String |

Table . specification db table

### G. Delivery&purchase

|  |  |
| --- | --- |
| Name | Type |
| brand | Text String |
| d&f\_id | Text String |
| can\_delivery | Text String |
| where\_to\_buy | Text String |

Table . purchase db table

# 8. Testing plan

## 8.1 Testing Policy

### A. Development Testing

Development test는 기술상의 성능(신뢰도, 가용도, 보전도, 적합성, 호환성)을 측정하고, 설계상의 중요한 모든 문제점이 해결되었는가를 확인하는 test이다. 개발 과정에서 component 단위로 test를 진행해야 한다.

### B. Release Testing

Release test는 서비스를 출시하기 전에 결함이나 문제가 없는지, 의도한대로 작동이 되는지 확인하는 test이다. 요구사항이 모두 반영되었는지 확인해야 하고 release 전에 완료해야 한다.

### C. User Testing

User test는 사용자의 환경에서 시스템을 test하는 것이다. 실제로 일어날 수 있는 시나리오와 현실적인 상황을 설정해야 한다.

## 8.2 Test Case

### A. User Management System

**A.1 Sign up**

1) User: 회원가입 버튼을 누르고 회원가입을 시도한다.

2) 시스템 동작: DB에 있는 ID와 중복 여부를 확인한다.

2-1) 성공

시스템 알림: “회원가입이 성공적으로 완료되었습니다.”

시스템 동작: DB에 새로운 ID와 정보를 입력한다.

2-2) 실패

시스템 알림: “이미 사용중인 ID입니다.”

시스템 동작: ID를 다시 입력하도록 한다.

**A.2 Log in**

1) User: 회원가입한 정보를 이용하여 로그인을 시도한다..

2) 시스템 동작: DB에 있는 ID에 등록되어 있는지 확인한다.

2-1) 성공

시스템 알림: “로그인 되었습니다.”

시스템 동작: DB에 있는 User 정보를 불러온다.

2-2) 실패

시스템 알림: “ID나 비밀번호가 잘못 입력되었습니다.”

시스템 동작: ID와 비밀번호를 다시 입력하도록 한다.

### B. Searching System

1) User: 동영상에 관련된 단어를 검색하여 동영상을 찾는다.

2) 시스템 동작: 동영상 정보 DB의 키워드에서 해당 단어가 있는지 확인하고 그 결과를 출력한다.

2-1) 성공

시스템 알림: “검색된 영상을 표시합니다.”

시스템 동작: 검색한 키워드가 있는 동영상을 출력한다.

2-2) 실패

시스템 알림: “해당하는 동영상이 없습니다.”

시스템 동작: 다른 단어를 검색할 수 있도록 한다.

### C. Recommend System

1) User: 추천 영상을 확인하기 위해 추천 영상 목록에 들어간다.

2) 시스템 동작: User가 지금까지 입력한 동영상 정보를 이용하여 User의 취향에 맞게 동영상 목록을 출력한다.

2-1) 성공

시스템 알림: “추천 동영상 목록을 만들었습니다”

시스템 동작: 분석한 동영상 목록을 출력한다.

2-2) 실패

시스템 알림: “취향을 분석할 수 있는 데이터가 없습니다.”

시스템 동작: 다른 동영상을 먼저 볼 수 있도록 한다.

### D. Delivery/Market System

**D.1 Delivery system**

1) User: 동영상을 시청하고 해당 음식을 배달로 시켜먹기 위해 배달 버튼을 누른다.

2) 시스템 동작: DB에서 해당 음식의 배달 정보를 불러온다.

2-1) 성공

시스템 알림: “배달 정보입니다.”

시스템 동작: 배달 정보를 어플리케이션에서 실행시킨다.

2-2) 실패

시스템 알림: “해당하는 배달 정보가 없습니다.”

시스템 동작: 이전 화면으로 돌아간다.

**D.2 Internet market system**

1) User: 동영상을 시청하고 해당 음식을 인터넷 쇼핑으로 주문하기 위해 인터넷 주문 버튼을 누른다

2) 시스템 동작: DB에서 해당 음식의 인터넷 쇼핑 정보를 불러온다.

2-1) 성공

시스템 알림: “인터넷 쇼핑 정보입니다.”

시스템 동작: 인터넷 쇼핑 정보를 인터넷으로 실행시킨다.

2-2) 실패

시스템 알림: “해당하는 쇼핑 정보가 없습니다.”

시스템 동작: 이전 화면으로 돌아간다.

# 9.Development Plan

이 챕터에서는 실제 개발을 할 때 어떠한 기술과 개발 환경을 사용할 것인지 설명한다.

## 9.1 Frontend Environment

### A. Android Studio



Figure . Android studio

안드로이드 스튜디오는 안드로이드 및 안드로이드 전용 어플리케이션을 위한 공식 통합 개발 환경으로 안드로이드 어플리케이션에 개발에 필요한 툴들을 모두 가지고 있다. xml 디자인을 이용하여 손쉽게 U.I 디자인을 할 수 있으며 지원하는 언어로는 코틀린, 자바가 있는데 이 두 언어를 이용하여 상세 기능 및 다양한 api 및 http request 를 할 수 있다.

### B. JAVA

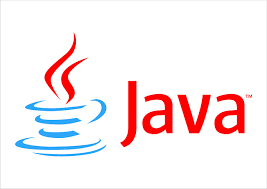


Figure . Java

대표적인 객체 지향 프로그래밍 언어로서 원래는 웹을 만드는 응용프로그램 개발용 기술이나 안드로이드 스튜디오 에서도 지원하는 언어로, 자바 JDK 를 설정하여 자바로도 기능 들을 구현 할 수 있게 된다.

## 9.3 Backend Environment

### A. Cloud Firestore

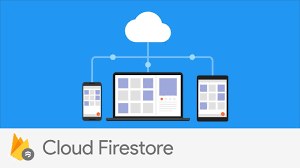


Figure . Cloud Firestore

구글에서 만든 모바일 앱 개발을 위한 firebase 의 최신 데이터베이스로서 데이터베이스에 대한 직관 적인 데이터 모델을 가지고 있는게 특징이다. 다른 데이터베이스 플랫폼 보다 풍부하고 빠른 쿼리와 우수한 확장성을 제공하고, 손쉽게 firebase function을 이용하여 DB에 접근하고 사용 할 수 있다.

### B. Nodejs



Figure . Node.JS

Node.js 는 대표적인 서버 프레임워크로서 다양한 모듈을 손쉽게 사용할 수 있고, 라이버러리도 풍부하여 다양한 환경을 구성할 수 있도록 해주는 플랫폼이다. 언어로는 javascript 를 쓰고 firestore function에도 해당 프레임워크를 지원한다.

### C. Python



Figure . Python

파이썬은 비교적 고차원의 언어로 배우기 쉽고 강력한 프로그래밍 언어이다. 효율적인 자료구조들과 객체지향 프로그래밍으로 효과적인 접근법을 제공한다. 해당 프로젝트에서는 recommendation system의 알고리즘을 구현 할 때 해당 언어를 쓸 것이다.

# 10. Index

## 10.1. Figure Index

[Figure 1. Class diagram 5](#_Toc24580627)

[Figure 2. State diagram 6](#_Toc24580628)

[Figure 3. Sequence diagram 6](#_Toc24580629)

[Figure 4. ER Diagram 7](#_Toc24580630)

[Figure 5. draw.io 8](#_Toc24580631)

[Figure 6. ERDCloud 8](#_Toc24580632)

[Figure 7. Powerpoint 9](#_Toc24580633)

[Figure 8. Web Seqence Diagrams 9](#_Toc24580634)

[Figure 9. JSON 42](#_Toc24580635)

[Figure 10. Android studio 55](#_Toc24580636)

[Figure 11. Java 56](#_Toc24580637)

[Figure 12. Cloud Firestore 57](#_Toc24580638)

[Figure 13. Node.JS 57](#_Toc24580639)

[Figure 14. Python 58](#_Toc24580640)

## 10.2. Diagram Index

[Diagram 1 . System organization 10](#_Toc24580641)

[Diagram 2. Frontend application 11](#_Toc24580642)

[Diagram 3. Backend Application 12](#_Toc24580643)

[Diagram 4. user information class diagram 14](#_Toc24580644)

[Diagram 5. user information state diagram 17](#_Toc24580645)

[Diagram 6. search class diagram 18](#_Toc24580646)

[Diagram 7. search state diagram 21](#_Toc24580647)

[Diagram 8. specification class diagram 22](#_Toc24580648)

[Diagram 9. specification state diagram 25](#_Toc24580649)

[Diagram 10. Recommend Video class diagram 26](#_Toc24580650)

[Diagram 11. Recommend State diagram 27](#_Toc24580651)

[Diagram 12. Backend Sign in class diagram 28](#_Toc24580652)

[Diagram 13. Backend Sign in Sequence diagram 29](#_Toc24580653)

[Diagram 14. Backend Sign up class diagram 30](#_Toc24580654)

[Diagram 15. Backend sign up sequence diagram 31](#_Toc24580655)

[Diagram 16. Backend My page class diagram 32](#_Toc24580656)

[Diagram 17. Backend my page sequence diagram 33](#_Toc24580657)

[Diagram 18. Backend Search class diagram 34](#_Toc24580658)

[Diagram 19. Backend Search sequence diagram 35](#_Toc24580659)

[Diagram 20. Backend Specification class diagram 36](#_Toc24580660)

[Diagram 21. Backend Specification sequence diagram 38](#_Toc24580661)

[Diagram 22. Backend Recommend class diagram 39](#_Toc24580662)

[Diagram 23. Backend recommend sequence diagram 40](#_Toc24580663)

[Diagram 24. backend Ranking class diagram 41](#_Toc24580664)

[Diagram 25. Backend Ranking sequence diagram 42](#_Toc24580665)

[Diagram 26. ER diagram 48](#_Toc24580666)

## 10.3. Table Index

[Table 1. sign up request 43](#_Toc24580667)

[Table 2. sign up response 43](#_Toc24580668)

[Table 3. Sign in request 44](#_Toc24580669)

[Table 4. Sign in response 44](#_Toc24580670)

[Table 5. my page request 44](#_Toc24580671)

[Table 6. my page response 44](#_Toc24580672)

[Table 7. ranking request 45](#_Toc24580673)

[Table 8. ranking response 45](#_Toc24580674)

[Table 9. Search request 45](#_Toc24580675)

[Table 10. search response 45](#_Toc24580676)

[Table 11. review request 46](#_Toc24580677)

[Table 12. review response 46](#_Toc24580678)

[Table 13. recommend request 46](#_Toc24580679)

[Table 14. recommend response 46](#_Toc24580680)

[Table 15. shopping info request 47](#_Toc24580681)

[Table 16. shopping info response 47](#_Toc24580682)

[Table 17. user db table 50](#_Toc24580683)

[Table 18. autority db table 50](#_Toc24580684)

[Table 19. user\_watched db table 50](#_Toc24580685)

[Table 20. video db table 51](#_Toc24580686)

[Table 21. category db table 51](#_Toc24580687)

[Table 22. specification db table 51](#_Toc24580688)

[Table 23. purchase db table 52](#_Toc24580689)