**Design Specification**

****

**ARIT Market**

**Group #5**

|  |  |  |  |
| --- | --- | --- | --- |
| **학번** | 2017310887 | **이름** | 손하민 |
| **학번** | 2016314609 | **이름** | 이홍균 |
| **학번** | 2015312193 | **이름** | 임현묵 |
| **학번** | 2017311337 | **이름** | 한상미 |

Content

[1. Preface 5](#_Toc24556468)

[1.1 Objective 5](#_Toc24556469)

[1.2 Readership 5](#_Toc24556470)

[1.3 Document Structure 5](#_Toc24556471)

[2. Introduction 8](#_Toc24556472)

[2.1 Objective 8](#_Toc24556473)

[2.2 Applied Diagrams 8](#_Toc24556474)

[2.3 Applied Tools 9](#_Toc24556475)

[2.4 Project Scope 9](#_Toc24556476)

[3. System Architecture 11](#_Toc24556477)

[3.1 Objective 11](#_Toc24556478)

[3.2 System Organization 11](#_Toc24556479)

[4. User Management System 13](#_Toc24556480)

[4.1 Objective 13](#_Toc24556481)

[4.2 Class Diagram 13](#_Toc24556482)

[4.3 Sequence Diagram 15](#_Toc24556483)

[5. Home & Mypage interface system 16](#_Toc24556484)

[5.1 Objective 16](#_Toc24556485)

[5.2 Class Diagram 16](#_Toc24556486)

[5.3 Sequence Diagram 19](#_Toc24556487)

[6. Category & Search System 21](#_Toc24556488)

[6.1 Objective 21](#_Toc24556489)

[6.2 ClassDiagram 21](#_Toc24556490)

[6.3 Sequence Diagram 23](#_Toc24556491)

[7. Product & Post System 25](#_Toc24556492)

[7.1 Objective 25](#_Toc24556493)

[7.2 Class Diagram 25](#_Toc24556494)

[7.3 Sequence Diagram 26](#_Toc24556495)

[8. AR Mode System 28](#_Toc24556496)

[8.1 Objective 28](#_Toc24556497)

[8.2 Class Diagram 28](#_Toc24556498)

[8.3 Sequence Diagram 29](#_Toc24556499)

[9. AR Measure System 30](#_Toc24556500)

[9.1 Objective 30](#_Toc24556501)

[9.2 Class Diagram 30](#_Toc24556502)

[9.3 Sequence Diagram 31](#_Toc24556503)

[10. Protocol Design 32](#_Toc24556504)

[10.1 Objective 32](#_Toc24556505)

[10.2 JSON 32](#_Toc24556506)

[10.3 Protocol description 32](#_Toc24556507)

[11. Database Design 35](#_Toc24556508)

[11.1 Objective 35](#_Toc24556509)

[11.2 ER Diagram 35](#_Toc24556510)

[11.3 Entity 36](#_Toc24556511)

[11.4 Relationship 38](#_Toc24556512)

[11.5 Relational Model 39](#_Toc24556513)

[12. Testing Plan 41](#_Toc24556514)

[12.1 Objective 41](#_Toc24556515)

[12.2 User Management System 41](#_Toc24556516)

[12.3 Home & Mypage System 41](#_Toc24556517)

[12.4 Category & Search System 43](#_Toc24556518)

[12.5 Product & Post System 43](#_Toc24556519)

[12.6 AR Mode System 44](#_Toc24556520)

[12.7 AR Measure System 45](#_Toc24556521)

[13. Development Environment 46](#_Toc24556522)

[13.1 Objective 46](#_Toc24556523)

[13.2 Front-End Environment 46](#_Toc24556524)

[13.3 Back-End Environment 47](#_Toc24556525)

[13.4 APIs and Tools 48](#_Toc24556526)

# Preface

## 1.1 Objective

Preface에서는 본 명세서의 예상 독자를 정의하고 문서의 구조와 각 부분의 역할에 대해 기술 한다.

## 1.2 Readership

본 문서의 독자는 시스템의 개발과 유지를 담당하는 사람들이라는 가정하에 서술한다. 이는 시스템을 실제로 개발하는 소프트웨어 엔지니어, 시스템을 설계하는 설계자, 클라이언트 엔지니어, 고객 지원을 위한 서비스 팀 등 시스템 개발에 관련되는 모든 구성원들을 포함한다.

## 1.3 Document Structure

1. **Preface**

Preface에서는 본 명세서의 예상 독자를 정의하고 문서의 구조와 각 부분의 역할에 대해 기술 한다.

1. **Introduction**

The objective of the following section is to introduce the diagrams and tools used for the System Architecture and to discuss the scope of this project.

1. **System Architecture**

This *System Architecture* section aims to describe the general structure of the system. Using block diagrams and the UML this section aims to represent the system as a whole as well as the interaction between subsystems.

1. **User Management System**

User Management System은 ARIT 앱에서 사용자의 계정을 관리하고 회원가입과 로그인을 담당하는 subsystem이다. 사용자의 계정을 user DB에 등록하고 가입된 사용자가 로그인을 할 시 user DB로부터 사용자의 정보를 가져온다. Class diagram, Sequence diagram을 통해 해당 시스템의 구조를 나타내고 설명한다.

1. **Home & Mypage System**

Home & Mypage Interface System은 ARIT 앱을 실행했을 때 볼 수 있는 메인 화면과 사용자가 자신의 계정 정보를 관리할 수 있는 마이페이지 화면을 담당하는 subsystem이다. 사용자의 계정 정보를 user DB에서 가져와 사용자의 계정을 변경할 수 있는 마이페이지 화면과 다른 기능을 담당하는 subsystem을 호출할 수 있는 메인 화면을 Class diagram과 Sequence diagram을 통해 나타낸다.

1. **Category & Search System**

Category & Search System은 ARIT 앱 내에 등록된 제품(게시글)들을 카테고리별로 나누어 보여주고 원하는 제품(게시글)을 검색하는 기능을 담당하는 subsystem이다. Product DB에서 카테고리에 해당하는 제품들을 가지고 오거나 query문을 통해 원하는 제품을 찾는 기능을 제공한다. Class diagram과 Sequence diagram을 통해 구조를 표현하고 시스템에 대하여 설명한다.

1. **Product & Post System**

사용자는 자신의 물품을 판매하기 위해서 게시글을 작성하게 된다. 여기서 본 시스템인 ‘ARIT’는 다양한 툴을 제공하여, 물품이 좀 더 효과적으로 구매자들에게 전달될 수 있도록 판매자를 보조하는 역할을 한다. 또한 사용자는 구매를 하기 위해 게시글을 열람하고, 게시글 안에서 가볍게 판매하는 사용자와 댓글로 소통한다. 본 시스템은 구매자의 제품에 대한 이해도를 높이기 위해 여러 기능을 제공한다.

1. **AR Mode System**

‘ARIT’의 핵심기능 중 하나인 AR배치 기능은 증강현실 기술을 이용하여 구매자가 있는 실제 공간에 판매자가 판매하는 물품을 배치한다. 구매자가 제품에 대한 정보 및 사양을 알아본다고 하더라도, 크기는 실제로 보지 않는 이상 감을 잡기 힘들다. 특히 노트북과 같이 크기에 민감한 제품이나, 데스크탑 본체의 사이즈 등 크기가 사진으로 구분이 잘 가지 않는 것들은 특히 그렇다. 이때, 제품의 크기를 육면체 등의 모양으로 보여주어, 제품의 크기를 실공간에서의 부피를 알 수 있게 한다. 육면체의 크기는 판매자가 AR 측정 기능을 통해 측정한 값을 기반으로 한다.

1. **AR Measure System**

‘ARIT’의 핵심기능 중 하나인, 물품의 크기를 인식하는 기능이다. 판매자는 물품의 크기를 게시글에 게재하도록 되어있는데, 모든 판매자가 물품의 사이즈를 수치로 알고 있는 경우는 적다. 직접 자 등을 이용하여 측정하거나, 다른 곳에서 정보를 알아볼 필요없이, 본 시스템은 AR측정 기능을 도입하여, 곧바로 제품의 크기(높이, 너비, 길이)를 측정할 수 있도록 한다.

1. **Protocol Design**

Protocol Design 에서는 ‘ARIT’에서 각 System 간의 communtication에 있어서 필요한 protocol에 대해서 기술하고자 한다. Protocol에 필요한 기본적인 data format에서는 XML, JSON 등이 있는데 ‘ARIT’에서는 JSON을 사용한다. 따라서 이에 대한 설명과 각 protocol에서 사용되는 Attribute 와 Value는 무엇인지 자세히 기술한다.

1. **Database Design**

The database design section aims to show how the project handles its data. With the requirements in mind, this section will describe the ER diagram as well as its relational schema.

1. **Testing Plan**

Testing Plan에서는 시스템이 의도한대로 실행되고 시스템 내부의 결함을 찾기 위해 testing을 하는 과정을 설명한다. 따라서 Testing Policy와 여러 Test Case에 대해 기술한다.

1. **Development Environment**

Development Environment 에서는 ‘ARIT’ 이라는 하나의 시스템을 개발하기 위해서 필요한 Front-end, Back-end의 개발 환경과 외부 API에 대해서 서술한다.

# Introduction

## 2.1 Objective

The objective of the following section is to introduce the diagrams and tools used for the System Architecture and to discuss the scope of this project.

## 2.2 Applied Diagrams

1. **Block Diagram**

Block diagrams are diagrams of system that represents relationships between these systems using connected lines. These diagrams are often referred to as a high-level description of the systems, as there are less detailed and aims to describe the overall concept of the entire system.

1. **Unified Modeling Language**

As system modeling mainly refers to the representation system using graphical or diagram notation, UML, or Unified Modeling Language, is known for its wide use in this regard. This language is particularly important in allowing communication between stakeholders, as this is also a high-level description of the system. In this project, the Class Diagram, Sequence Diagram, and State Diagram of the UML were used represent the system.

1. **Class Diagram**

Class diagrams are used in this document to show the object classes in the system and the association between these classes. There are two terms important within this diagram – the first is *Class*, and the second *Association.* Class is a representation of a single object in the system, which Association is the link between these objects, or classes, and the relationship between them.

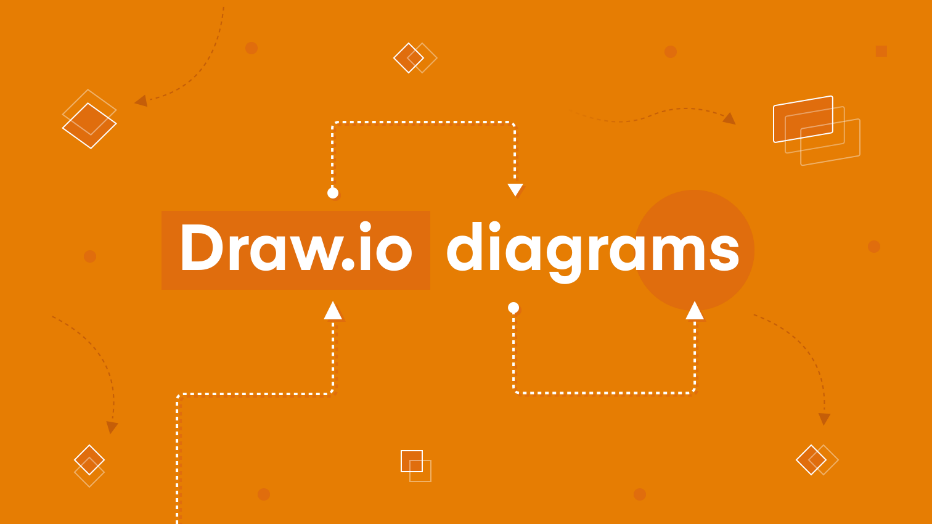
1. **Sequence Diagram**

Sequence diagrams are used in this document to show interaction between actors and system and also between system components. The two terms important in this diagram is Object and Actor. The actors are listed at either the top or the bottom portion of the diagram, and the interaction between objects are represented by arrows.

1. **Entity-Relationship Diagram**

Entity-relationship, or ER, diagrams are used to relate data within a specific domain. This diagram consists of entities and their relationships with other entities. Entities are represented as boxes, while the data associated with these entities are represented as ovals, and the relationship between these entities are represented with rhombuses.

## 2.3 Applied Tools



[Figure 1] Draw.io

For most diagrams used in this document, a free, online diagram tool called ‘draw.io’ was used. Draw.io is an easy-to-use open source software that allows engineers to comfortably and efficiently draw their diagrams.

## 2.4 Project Scope

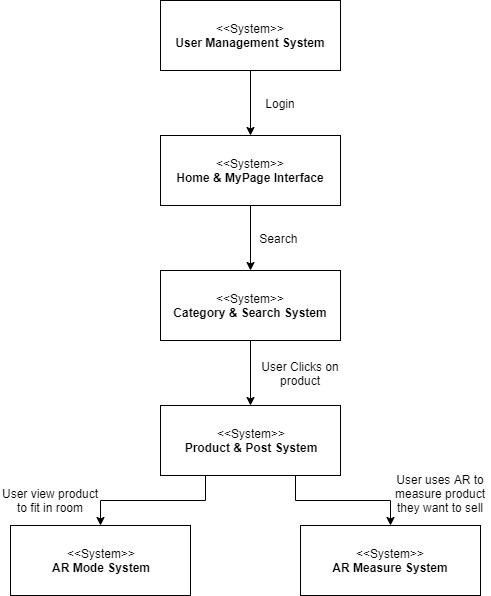
‘ARIT’ is a system accessible by anyone to purchase and sell IT products. The prominent feature ARIT provides is allowing measurements through AR as well as allow products to be placed inside of customer’s rooms through AR to allow the customer to see how a product will fit. Following, there are six (6) systems in ARIT. The first system is the *User Management System.* This system manages the information of the user – things like their ID and Password are stored in the database, and once a user enters their information, this system checks it against the database. They can also introduce new information through

# System Architecture

## 3.1 Objective

This *System Architecture* section aims to describe the general structure of the system. Using block diagrams and the UML this section aims to represent the system as a whole as well as the interaction between subsystems.

## 3.2 System Organization



**[Diagram 1] System Organization**

In *ARIT*, every user is given the privilege of both posting their own products or viewing products that other users have posted. This is all done through the online database, *Firebase*. Whenever a product is posted by a user, the central database is updated, and all users are given access to the newly modified database. Because of this, a dedicated server is not necessary with this system.

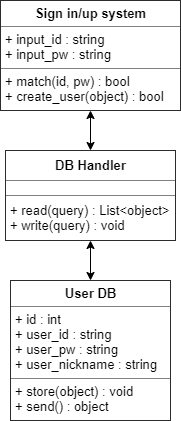
As seen above, when users first enter the app, they are taken to the *User Management System*. Here, they will be given the option of either signing in or signing up. Once a user has an account and they sign in, they are taken to the Home & MyPage Interface System. Within this system, users are given the option to search for products or simply view the categories available in the app. Despite what the user chooses, when they ultimately click a product they wish to view, they are taken to the *Product & Post System*. Within this system, users are given a detailed description of product they have chosen. With this option, they are able to view pictures, prices, etc. as well as take the product into AR mode with the *AR Mode System*, where they will able to view the product size with AR to see how the product will fit in their room. Another option with the *Product & Post System* would be to chose to post a product a user wants to sell. Here, the user will be able to include detailed descriptions of their product as well as measure their product using the *AR Measure System*.

# User Management System

## 4.1 Objective

User Management System은 ARIT 앱에서 사용자의 계정을 관리하고 회원가입과 로그인을 담당하는 subsystem이다. 사용자의 계정을 user DB에 등록하고 가입된 사용자가 로그인을 할 시 user DB로부터 사용자의 정보를 가져온다. Class diagram, Sequence diagram을 통해 해당 시스템의 구조를 나타내고 설명한다.

## 4.2 Class Diagram



**[diagram 2] user management class diagram**

1. **Sign in/up System**

**A.1 Attributes**

String input\_id : 사용자가 인풋으로 입력한 아이디

String input\_pw : 사용자가 인풋으로 입력한 비밀번호

**A.2 Methods**

Bool match(id, pw) : 사용자가 입력한 id와 pw으로 일치하는 계정을 찾는다. 일치하는 계정이 있다면 true를, 없다면 false를 반환한다.

Bool create\_user(object) : 사용자가 입력한 정보들을 가지고 계정을 생성한다. 만약 생성에 성공하면 true를, 실패하면 false를 반환한다.

1. **DB Handler**

**B.1 Attributes**

없음

**B.2 Methods**

List<object> read(query) : 데이터베이스를 조회하는 쿼리를 입력 받으면 데이터베이스에 질의한다.

Void write(query) : 데이터베이스에 저장하는 쿼리를 입력 받으면 데이터베이스에 질의한다.

1. **User DB**

**C.1 Attributes**

Int id : 데이터베이스에서 저장되는 고유 식별 번호

String user\_id : 사용자의 아이디

String uwer\_pw : 사용자의 비밀번호

String user\_nickname : 사용자의 닉네임 (별명)

**C.2 Methods**

Void store(object) : 데이터베이스에 데이터를 저장한다.

Object send() : 데이터베이스로부터 데이터를 읽어와 전달한다.

## 4.3 Sequence Diagram

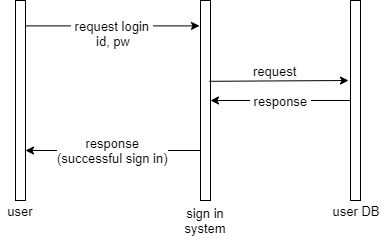
1. **Sign up System**

스크린샷이(가) 표시된 사진

자동 생성된 설명

**[diagram 3] sign up system sequence diagram**

1. **Sign in System**



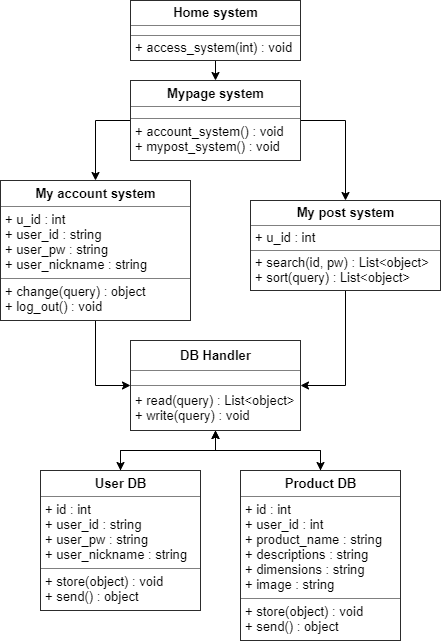
**[diagram 4] sign in system sequence diagram**

# Home & Mypage interface system

## 5.1 Objective

Home & Mypage Interface System은 ARIT 앱을 실행했을 때 볼 수 있는 메인 화면과 사용자가 자신의 계정 정보를 관리할 수 있는 마이페이지 화면을 담당하는 subsystem이다. 사용자의 계정 정보를 user DB에서 가져와 사용자의 계정을 변경할 수 있는 마이페이지 화면과 다른 기능을 담당하는 subsystem을 호출할 수 있는 메인 화면을 Class diagram과 Sequence diagram을 통해 나타낸다.

## 5.2 Class Diagram



**[diagram 6] home & mypage system class diagram**

1. **Home System**

**A.1 Attributes**

없음

**A.2 Methods**

Void access\_system(int) : 사용자가 원하는 메뉴에 접근하도록 한다.

1. **Mypage System**

**B.1 Attributes**

없음

**B.2 Methods**

Void account\_system() : 사용자가 자신의 계정을 관리하는 시스템에 접근하도록 한다.

Void mypost\_system() : 사용자가 자신의 글을 관리하는 시스템에 접근하도록 한다.

1. **My Account System**

**C.1 Attributes**

Int u\_id : 사용자의 고유 식별 번호

String user\_id : 사용자의 아이디

String user\_pw : 사용자의 비밀번호

String user\_nickname : 사용자의 닉네임

**C.2 Methods**

Object change(query) : 데이터베이스에 질의를 보내 정보를 수정한다.

Void log\_out() : 사용자의 로그아웃을 지원한다.

1. **My Post System**

**D.1 Attributes**

Int u\_id : 사용자의 고유 식별 번호

**D.2 Methods**

List<object> search(query) : 데이터베이스에 질의를 보내 원하는 게시글을가져온다.

List<object> sort(query) : 데이터베이스에 질의를 보내 원하는 정렬 방법으로 게시글을 정렬하여 가져온다.

1. **DB Handler**

**E.1 Attributes**

없음

**E.2 Methods**

List<object> read(query) : 데이터베이스를 조회하는 쿼리를 입력 받으면 데이터베이스에 질의한다.

Void write(query) : 데이터베이스에 저장하는 쿼리를 입력 받으면 데이터베이스에 질의한다.

1. **User DB**

**F.1 Attributes**

Int id : 데이터베이스에서 저장되는 고유 식별 번호

String user\_id : 사용자의 아이디

String uwer\_pw : 사용자의 비밀번호

String user\_nickname : 사용자의 닉네임 (별명)

**F.2 Methods**

Void store(object) : 데이터베이스에 데이터를 저장한다.

Object send() : 데이터베이스로부터 데이터를 읽어와 전달한다.

1. **Product DB**

**G.1 Attributes**

Int id : 데이터베이스에서 저장되는 고유 식별 번호

String product\_name : 게시글 제품의 이름

String descriptions : 게시글 제품 설명

String dimensions : 게시글 제품의 카테고리

String image : 게시글 제품의 이미지 경로

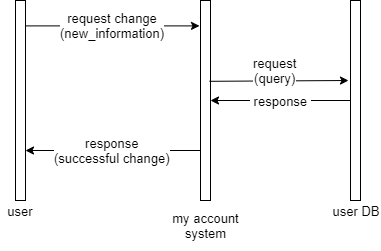
**G.2 Methods**

Void store(object) : 데이터베이스에 데이터를 저장한다.

Object send() : 데이터베이스로부터 데이터를 읽어와 전달한다.

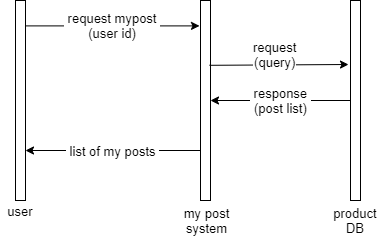
## 5.3 Sequence Diagram

1. **My Account System**



**[diagram 7] my account system sequence diagram**

1. **My Post System**



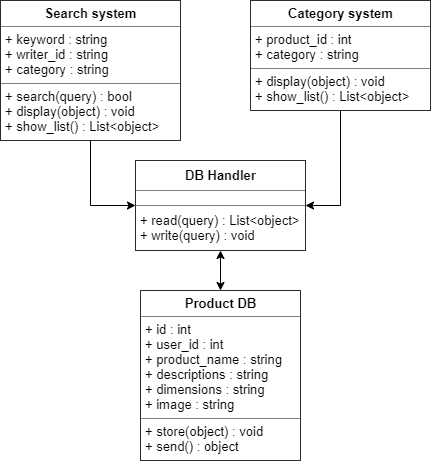
**[diagram 8] my post system sequence diagram**

# Category & Search System

## 6.1 Objective

Category & Search System은 ARIT 앱 내에 등록된 제품(게시글)들을 카테고리별로 나누어 보여주고 원하는 제품(게시글)을 검색하는 기능을 담당하는 subsystem이다. Product DB에서 카테고리에 해당하는 제품들을 가지고 오거나 query문을 통해 원하는 제품을 찾는 기능을 제공한다. Class diagram과 Sequence diagram을 통해 구조를 표현하고 시스템에 대하여 설명한다.

## 6.2 ClassDiagram



**[diagram 9] category & search system class diagram**

* 1. **Search System**

**A.1 Attributes**

String keyword : 검색 키워드

String writer\_id : 작성자 아이디

String category : 카테고리

**A.2 Methods**

List<object> search(query) : 검색 조건을 통해 검색된 게시글을 불러온다. 검색 결과가 없다면 null 값을 반환한다.

Void display(object) : 게시글을 UI를 통해 보여준다.

List<object> show\_list() : 검색된 게시글의 리스트를 보여준다.

* 1. **Category System**

**B.1 Attributes**

Int product\_id : 게시글의 고유 식별 번호

String category : 게시글의 카테고리

**B.2 Methods**

Void display(object) : 게시글을 UI를 통해 보여준다.

List<object> show\_list() : 검색된 게시글의 리스트를 보여준다.

* 1. **DB Handler**

**C.1 Attributes**

없음

**C.2 Methods**

List<object> read(query) : 데이터베이스를 조회하는 쿼리를 입력 받으면 데이터베이스에 질의한다.

Void write(query) : 데이터베이스에 저장하는 쿼리를 입력 받으면 데이터베이스에 질의한다.

* 1. **Product DB**

**D.1 Attributes**

Int id : 데이터베이스에서 저장되는 고유 식별 번호

String product\_name : 게시글 제품의 이름

String descriptions : 게시글 제품 설명

String dimensions : 게시글 제품의 카테고리

String image : 게시글 제품의 이미지 경로

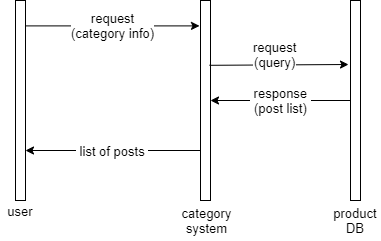
**D.2 Methods**

Void store(object) : 데이터베이스에 데이터를 저장한다.

Object send() : 데이터베이스로부터 데이터를 읽어와 전달한다.

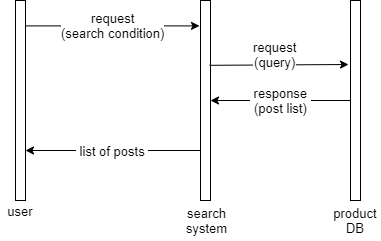
## 6.3 Sequence Diagram

1. **Category System**



**[diagram 10] category system sequence diagram**

1. **Search System**



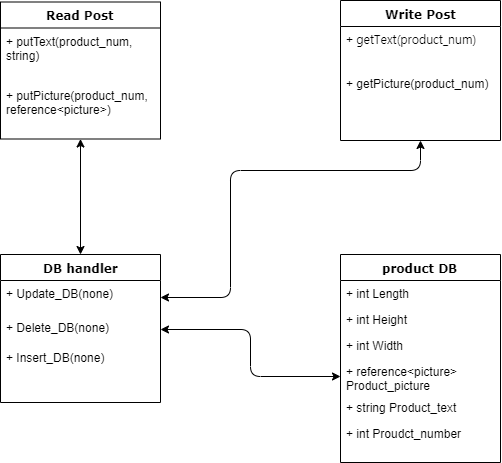
**[diagram 11] search system sequence diagram**

# Product & Post System

## 7.1 Objective

사용자는 자신의 물품을 판매하기 위해서 게시글을 작성하게 된다. 여기서 본 시스템인 ‘ARIT’는 다양한 툴을 제공하여, 물품이 좀 더 효과적으로 구매자들에게 전달될 수 있도록 판매자를 보조하는 역할을 한다. 또한 사용자는 구매를 하기 위해 게시글을 열람하고, 게시글 안에서 가볍게 판매하는 사용자와 댓글로 소통한다. 본 시스템은 구매자의 제품에 대한 이해도를 높이기 위해 여러 기능을 제공한다.

## 7.2 Class Diagram



**[Digram 12] Product & Post System**

putText(product\_num, string): 해당하는 제품 번호가 있는 DB에 string에 해당하는 텍스트를 업로드 하는 함수.

putPicture(product\_num, reference<picture>): 해당하는 제품 번호가 있는 DB에 picture에 해당하는 그림을 업로드 하는 함수.

Update\_DB(none): DB에 있는 특정 값을 Update하는 함수.

Delete\_DB(none): DB에 있는 특정 값을 Delete하는 함수.

Insert\_DB(none): DB에 특정 값을 Insert하는 함수.

getText(product\_num): 해당하는 제품번호가 있는 DB에 있는 텍스트를 가져오는 함수.

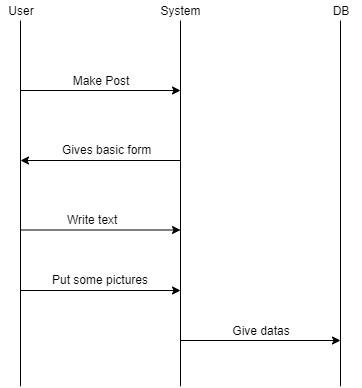
getPicture(product\_num): 해당하는 제품번호가 있는 DB에 있는 그림을 가져오는 함수.

Int length, height, width: 제품의 길이, 높이, 너비에 해당하는 변수

Int product\_num: 제품에 부여되는 있는 유니크한 숫자.

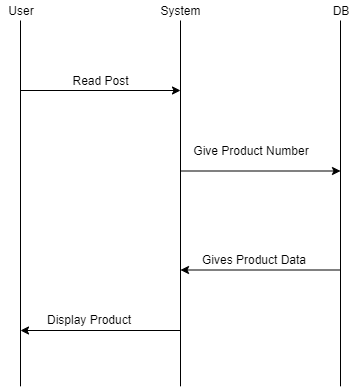
## 7.3 Sequence Diagram

1. **Post**



**[Diagram 13] Sequence Diagram of Post**

1. **Write**



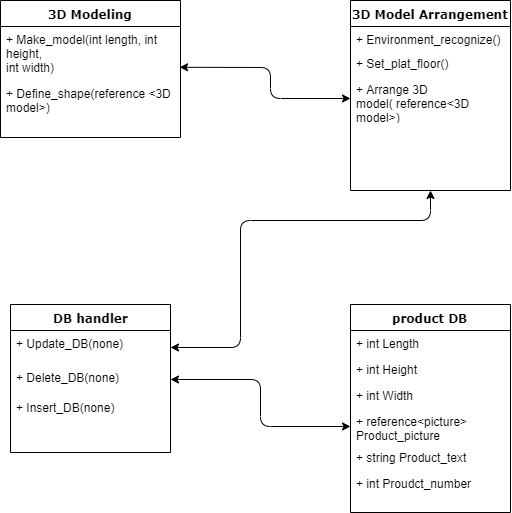
**[Diagram 14] Sequence Diagram of Write**

# AR Mode System

## 8.1 Objective

‘ARIT’의 핵심기능 중 하나인 AR배치 기능은 증강현실 기술을 이용하여 구매자가 있는 실제 공간에 판매자가 판매하는 물품을 배치한다. 구매자가 제품에 대한 정보 및 사양을 알아본다고 하더라도, 크기는 실제로 보지 않는 이상 감을 잡기 힘들다. 특히 노트북과 같이 크기에 민감한 제품이나, 데스크탑 본체의 사이즈 등 크기가 사진으로 구분이 잘 가지 않는 것들은 특히 그렇다. 이때, 제품의 크기를 육면체 등의 모양으로 보여주어, 제품의 크기를 실공간에서의 부피를 알 수 있게 한다. 육면체의 크기는 판매자가 AR 측정 기능을 통해 측정한 값을 기반으로 한다.

## 8.2 Class Diagram



**[Digram 15] AR Mode System**

Make\_model(int Length, int Height, int Width): 제품의 길이와 높이, 너비 정보를 가지고 3D model을 만드는 함수.

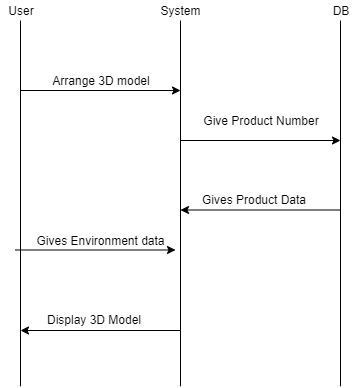
Define\_shape(reference<3D model>): 만들어진 3D model의 형태를 결정하는 함수.

Environment\_recognize(): 카메라에서 받은 화면으로 주의 환경을 인지하는 함수.

Set\_plat\_floor(): 인지한 환경 중에서 평평한 바닥을 찾아서 3D model을 배치할 준비를 하는 함수.

Arrange\_3D\_model(reference<3D model>): 세팅된 환경에 3D model을 배치하는 함수.

## 8.3 Sequence Diagram



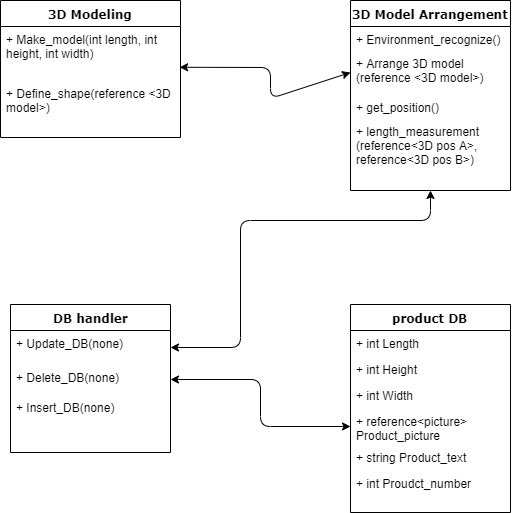
**[Digram 16] Sequence Digram of AR Mode System**

# AR Measure System

## 9.1 Objective

‘ARIT’의 핵심기능 중 하나인, 물품의 크기를 인식하는 기능이다. 판매자는 물품의 크기를 게시글에 게재하도록 되어있는데, 모든 판매자가 물품의 사이즈를 수치로 알고 있는 경우는 적다. 직접 자 등을 이용하여 측정하거나, 다른 곳에서 정보를 알아볼 필요없이, 본 시스템은 AR측정 기능을 도입하여, 곧바로 제품의 크기(높이, 너비, 길이)를 측정할 수 있도록 한다.

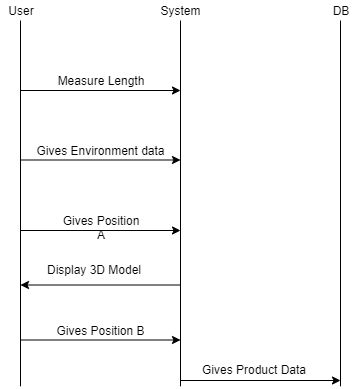
## 9.2 Class Diagram



**[Diagram 17] AR Measure System**

Length\_measurement(reference<3D pos A>, reference<3D pos B>): A지점과 B지점의 길이를 계산하는 함수.

## 9.3 Sequence Diagram



**[Diagram 18] Sequence Diagram of AR Measure System**

# Protocol Design

## 10.1 Objective

Protocol Design 에서는 ‘ARIT’에서 각 System 간의 communtication에 있어서 필요한 protocol에 대해서 기술하고자 한다. Protocol에 필요한 기본적인 data format에서는 XML, JSON 등이 있는데 ‘ARIT’에서는 JSON을 사용한다. 따라서 이에 대한 설명과 각 protocol에서 사용되는 Attribute 와 Value는 무엇인지 자세히 기술한다.

## 10.2 JSON

JSON은 JavaScript Object Notation의 약어로, XML과 더불어 대표적인 데이터 교환 방식이다. 자바스크립트를 기반으로 만들어진 데이터 표현 방식이지만, 프로그래밍 언어나 플랫폼에 독립적인 특성을 가지고 있어서, 다양한 언어에서 JSON을 활용할 수 있으므로, 본 시스템에서는 JSON을 기반으로 데이터 교환을 한다.

## 10.3 Protocol description

1. **Overview**

본 항목에서는 ‘ARIT’에서 사용되는 JSON의 메시지에서, Attribute와 Value에 대해서 자세하게 기술한다. Server와 Client 사이에서 전송되는 메시지를 request 와 response로 구분하여 설명한다. Request는 client가 server에 전송하는 값들을, response는 서버가 client의 request에 대해서 반응하여 나타내는 메시지를 의미한다.

1. **Sign in**

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| **Request** | |
| ID | 사용자의 아이디 |
| Nickname | 사용자의 닉네임 |
| Password | 사용자의 비밀번호 |
| Phone-number | 사용자의 전화번호 |
| Address | 사용자의 주소 |
| Age | 사용자의 나이 |
| Sex | 사용자의 성별 |
| **Response** | |
| SignIn\_Success | 회원가입 성공여부 |

**[Table 1] Sign in Protocol**

1. **Log in**

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| **Request** | |
| ID | 사용자의 아이디 |
| Password | 사용자의 비밀번호 |
| **Response** | |
| Login\_success | 로그인 성공여부 |

**[Table 2] Login Protocol**

1. **Search Product Protocol**

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| **Request** | |
| Keyword | 사용자의 검색어 |
| **Response** | |
| Product\_SN | 제품 고유번호 |
| Product\_title | 제품 제목 |
| Product\_picture | 제품 사진 |
| Product\_size | 제품 크기 |
| Product\_content | 제품 설명 |
| User\_SN | 제품 올린 사용자 고유번호 |

**[Table 3]Search Product Protocol**

1. **PostProduct Protocol**

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| **Request** | |
| Product\_SN | 제품 고유번호 |
| User\_SN | 사용자 고유번호 |
| Product\_title | 제품 제목 |
| Product\_picture | 제품 사진 |
| Product\_size | 제품 크기 |
| Product\_content | 제품 설명 |
| **Response** | |
| Post\_Success | 게시글 성공 여부 |

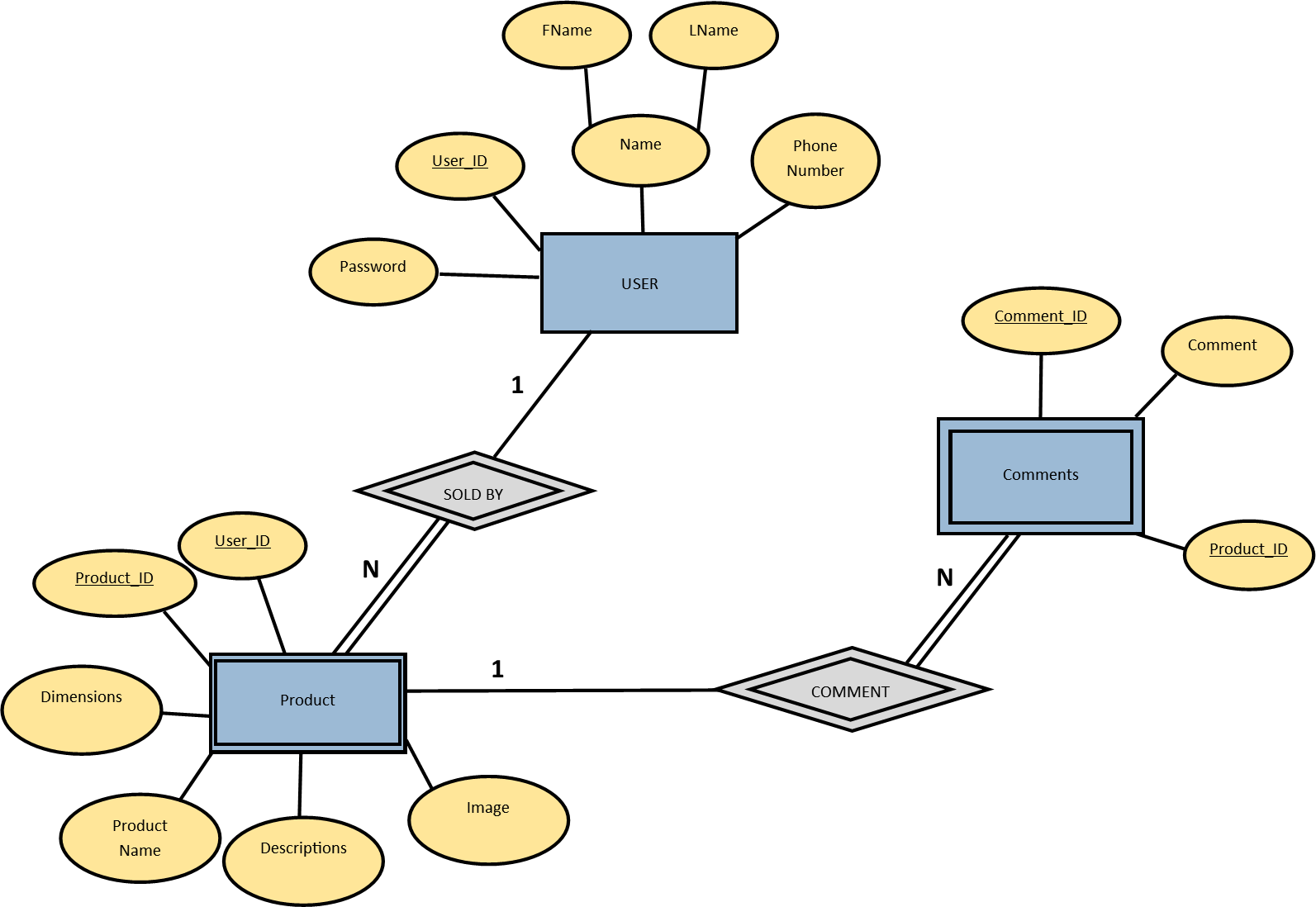
**[Table 4]Post Product Protocol**

# Database Design

## 11.1 Objective

The database design section aims to show how the project handles its data. With the requirements in mind, this section will describe the ER diagram as well as its relational schema.

## 11.2 ER Diagram

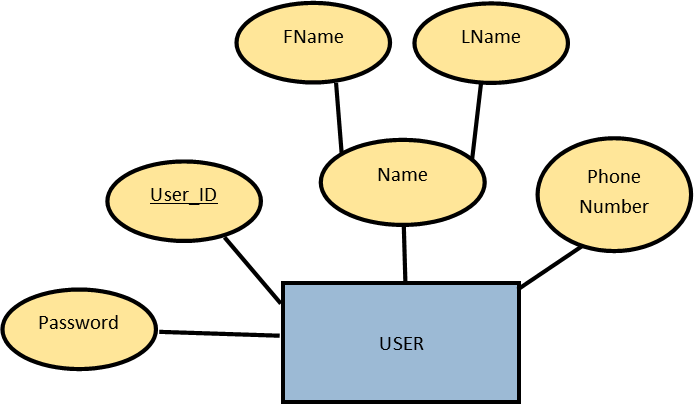


**[Diagram 19] DataBase**

Below is an ER diagram. Entities are represented by blue rectangles, while weak entities are represented by double rectangles. Their attributes are represented with yellow ovals, and multi-valued attributes have attributes branching from another attribute. The primary key with the attributes is underlined. The attributes are connected by a rhombus, which signifies the relation. Double lines represent that all attributes must have relation to the entity it has a relational to – that is, it cannot exist without a relation to another entity.

## 11.3 Entity

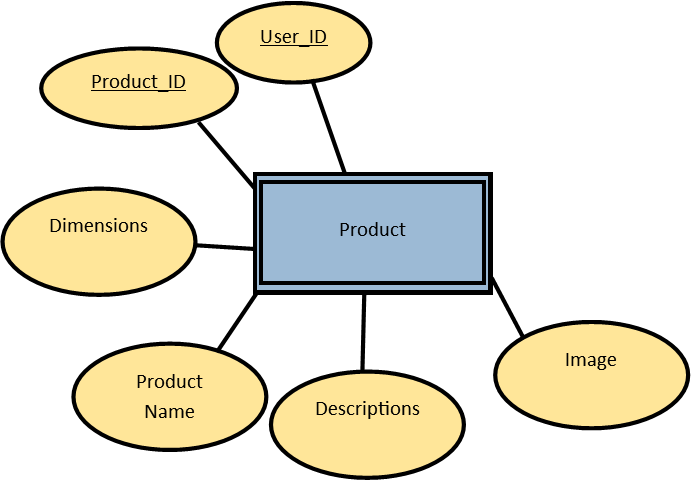
1. **User**



**[Diagram 20] User DB**

The *USER* entity represents the information stored for each user. The primary key is the *user\_id*, which must be unique for each user. The first and last name, password and phone number of each user is stored as well.

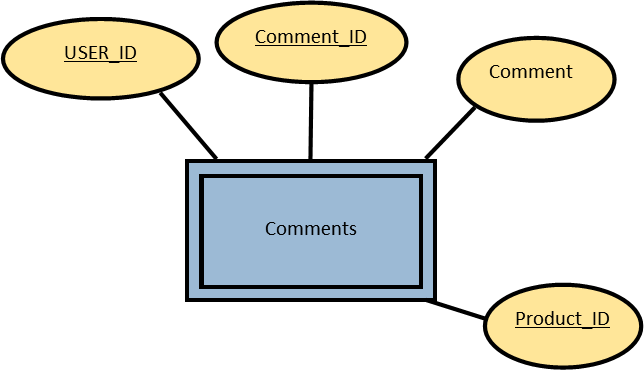
1. **Product**



**[Diagram 21] Product DB**

The *Product* entity is a weak entity that represents the information stored for each production. This entity has two primary keys, *User\_ID* and *Product\_ID,* as this entity cannot exist without a user being associated to it. This entity also has the attributes: *Dimensions, Product Name, Descriptions,* and *Image*.

1. **Comment**

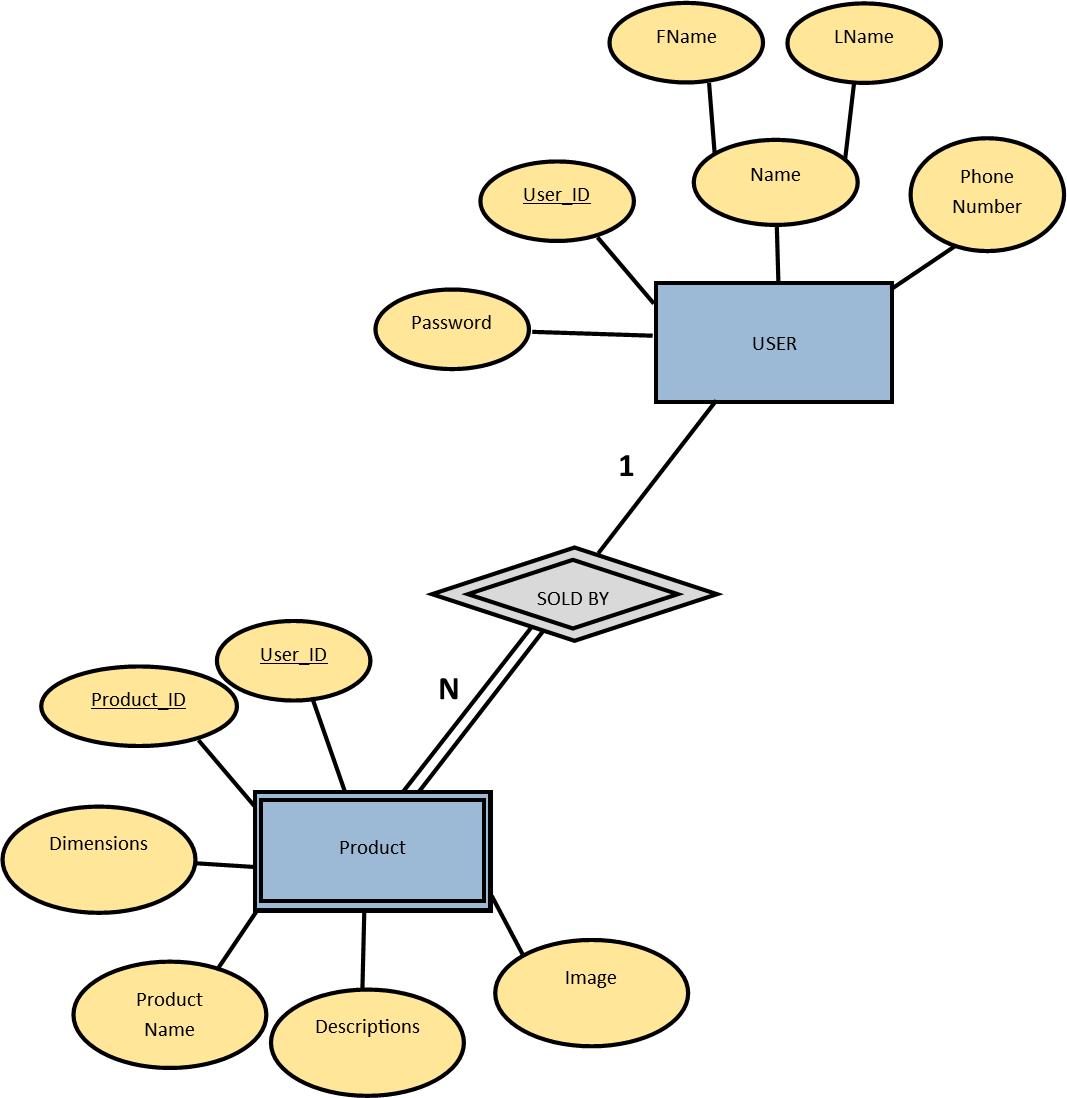


**[Diagram 22] Comment DB**

The *Comments* entity is also a weak entity that represents the information stored for each comment. As each comment must have a user and product related to it, this entity has three primary keys – *USER\_ID, Comment\_ID,* and *Product\_ID*. It also holds the information of the comment itself.

## 11.4 Relationship

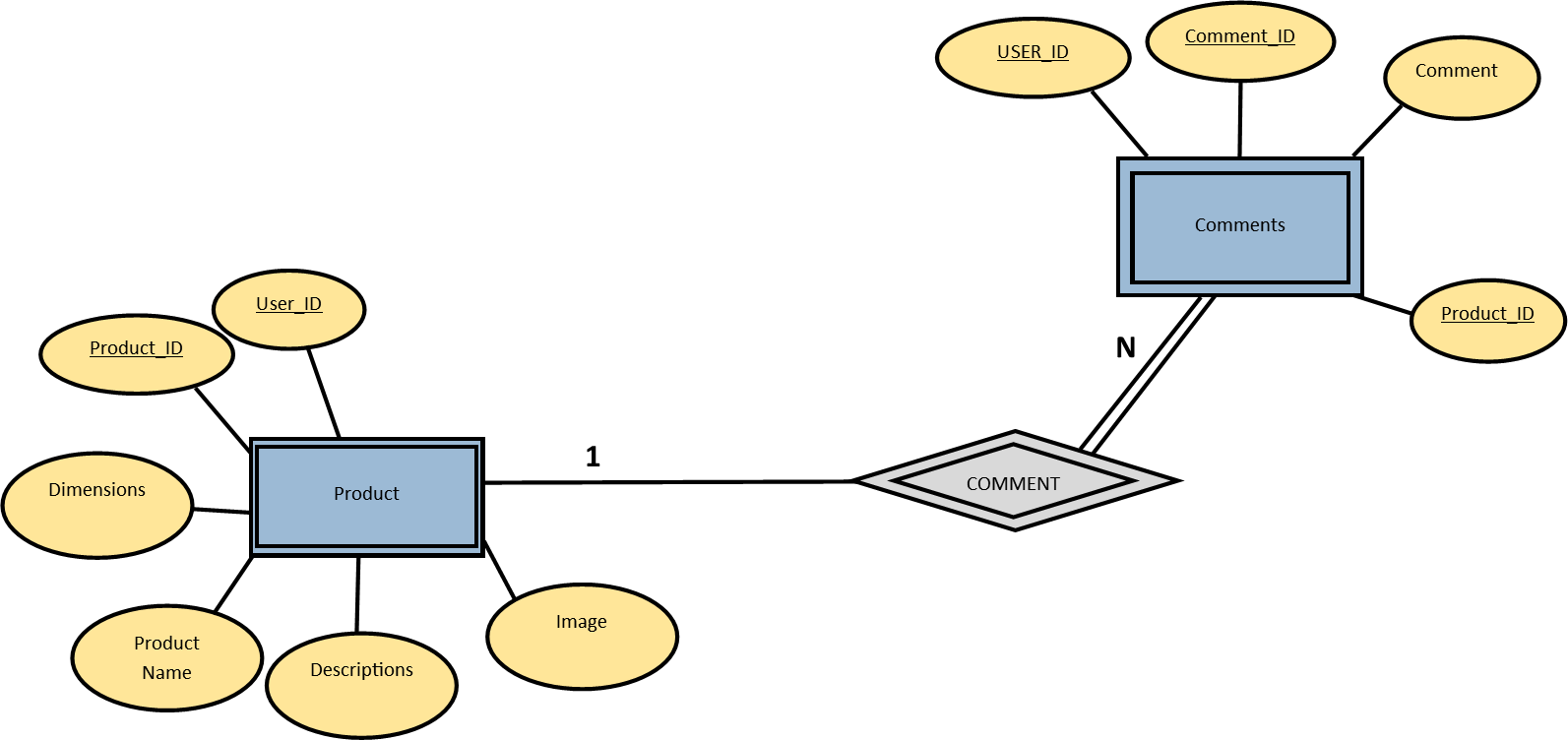
1. **Sold By**



**[Diagram 23] Sold By Relationship**

The *SOLD BY* relation is completed by a user posting a product. When a user posts a product, the database is updated accordingly. A single user is able to post multiple products, but a product cannot exist without a corresponding user.

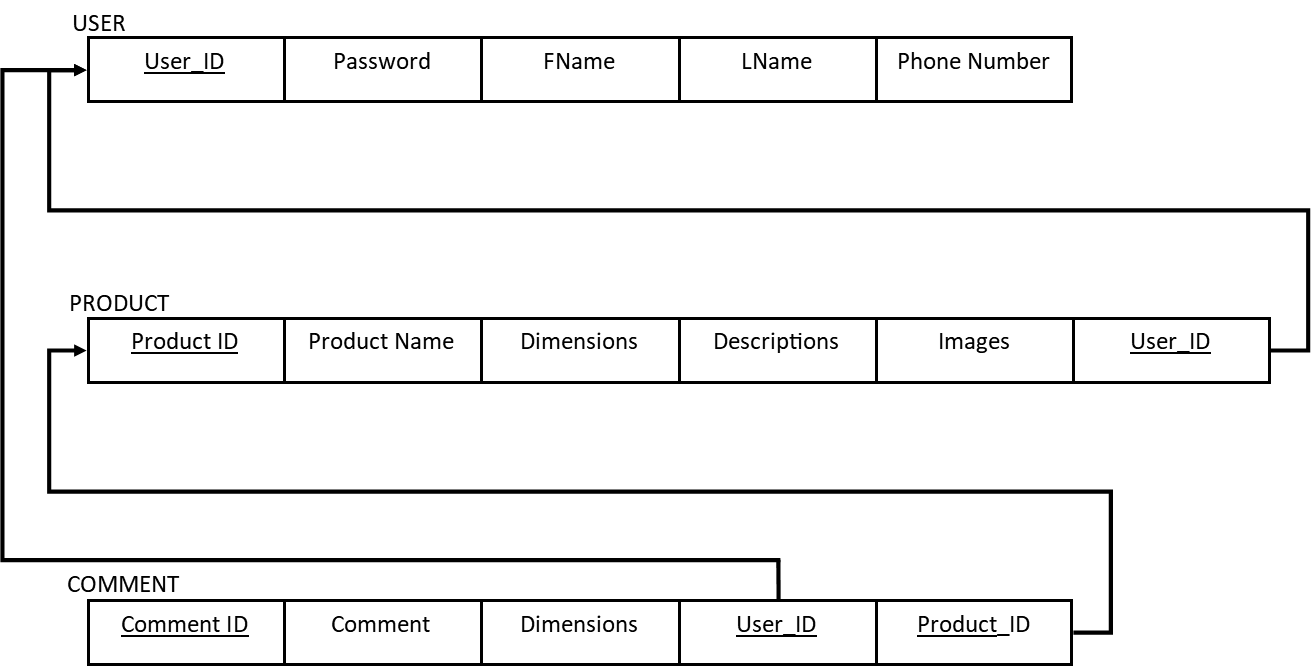
1. **Comment**



**[Diagram 24] Comment Relationship**

The *Comment* relation is done through a user posting a comment on a product. A product may have multiple comments by multiple users, but a comment cannot exist without a corresponding product.

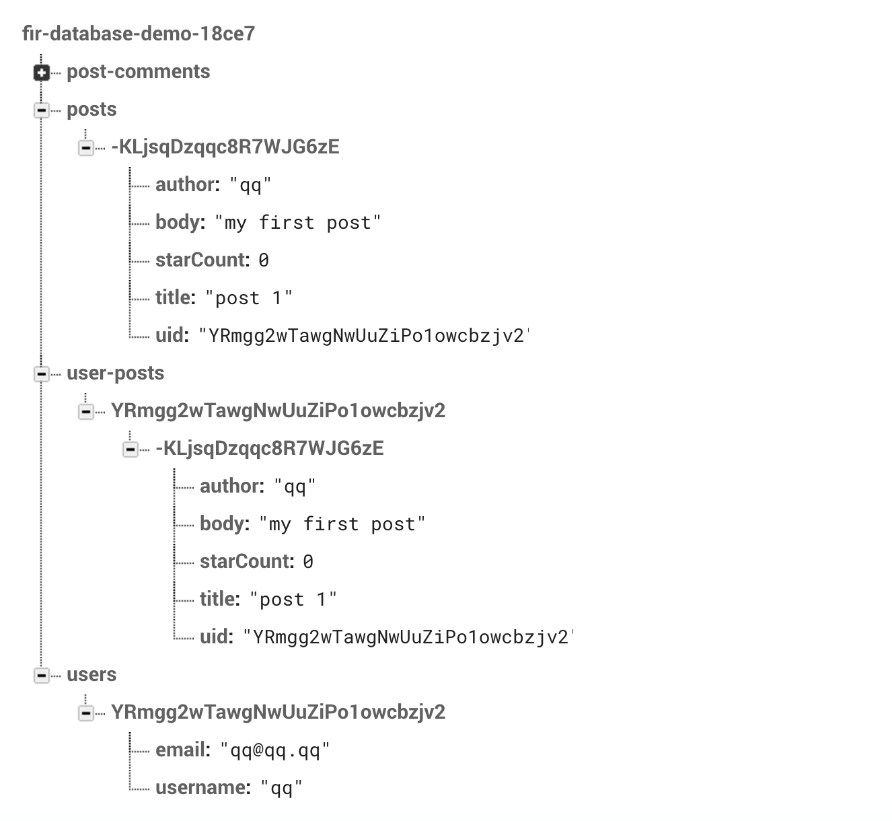
## 11.5 Relational Model



**[Diagram 25] Relational Model**

The diagram above is the relational model for the system. As seen in the ER diagram, there are three entities: *User, Product,* and *Comment*. Both *Product* and *Comment* are related to *User* through the unique *User­\_ID*, and *Comment* is related to *Product* through each unique *Product\_ID*.

However, as seen below, because Firebase stores data in a tree type structure, and can be represented as a JSON file format, the mySQL tabular format is not necessary. That is, SQL DDL is not necessary with Firebase.



**[Figure 2] example of Firebase DB**

# Testing Plan

## 12.1 Objective

## 12.2 User Management System

* + 1. **‘회원가입’ 버튼 누르기**

|  |  |
| --- | --- |
| User’s Action | 사용자가 계정을 등록하기 위해 ‘회원가입’ 버튼을 누름. |
| System’s Action | 사용자가 입력한 정보를 가져오고, 데이터베이스에 중복되는 회원이 있는지 확인하여 회원가입 성공 여부를 화면에 표시함. |
| Success | ‘회원 가입 성공’이라는 토스트를 띄움. |
| Failure | ‘회원 가입 실패’라는 토스트를 띄우고 어떤 이유로 실패했는지 토스트로 알려줌. 이유로는 중복되는 아이디를 사용하는 회원이 있거나 네트워크 문제 등이 있음. |

* + 1. **‘로그인’ 버튼 누르기**

|  |  |
| --- | --- |
| User’s Action | 사용자가 아이디와 비밀번호를 입력하고 ‘로그인’ 버튼을 누름. |
| System’s Action | 사용자가 입력한 정보를 가져오고, 데이터베이스에 일치하는 회원의 정보를 가져와서 홈 화면으로 넘어감. |
| Success | 회원의 정보를 가져와서 ‘OO회원님 환영합니다.’라는 토스트를 띄움. |
| Failure | 어떤 이유로 로그인에 실패했는지 토스트를 띄움. 이유로는 계정이 없는 상태이거나 아이디 또는 비밀번호를 잘못 입력하거나 네트워크 문제 등이 있음. |

## 12.3 Home & Mypage System

* + 1. **메뉴 선택**

|  |  |
| --- | --- |
| User’s Action | 사용자가 홈 화면에 있는 메뉴를 클릭함. |
| System’s Action | 사용자가 입력한 메뉴가 무엇인지 확인하고, 해당 시스템을 접근하여 해당 화면으로 넘겨줌. |
| Success | 해당 메뉴 화면으로 넘어감. |
| Failure | 네트워크 등의 문제이면 다시 홈 화면으로 넘어감. 만약 로그인 세션이 만료되었다면 로그인 페이지로 돌아감. |

* + 1. **회원 정보 수정**

|  |  |
| --- | --- |
| User’s Action | 사용자가 새로운 회원 정보를 입력하고 ‘수정’ 버튼을 누름. |
| System’s Action | 사용자가 입력한 새로운 회원 정보를 가져오고, 데이터베이스에 중복되는 정보인지 확인한 후 중복되지 않으면 회원 정보를 수정함. |
| Success | ‘수정 완료’라는 토스트를 띄움. |
| Failure | ‘수정할 수 없습니다.’라는 토스트를 띄우고 왜 수정할 수 없는지 이유를 토스트로 알려줌. 이유로는 사용자가 입력한 정보가 기존 다른 회원과 중복되거나 네트워크 문제 등이 있음. 로그인 세션이 만료되었다면 로그인 페이지로 돌아감. |

* + 1. **내 글 확인**

|  |  |
| --- | --- |
| User’s Action | 사용자가 내 글 확인 메뉴를 클릭. |
| System’s Action | 사용자의 고유 번호를 가져와 데이터베이스로부터 사용자의 번호로 작성된 게시글 목록을 가져옴. |
| Success | 리스트 뷰로 사용자가 작성한 게시글 목록을 띄워줌. |
| Failure | 로그인 세션이 만료되었다면 로그인 화면으로 돌아감. 작성된 게시물이 없다면 ‘작성된 게시글이 없습니다.’라는 토스트를 띄움. |

* + 1. **내 글 수정**

|  |  |
| --- | --- |
| User’s Action | 사용자가 자신의 글을 수정하고 ‘수정’ 버튼을 클릭. |
| System’s Action | 사용자가 입력한 글의 내용을 가져오고, 데이터베이스에 접근하여 글의 내용을 수정함. |
| Success | ‘수정 완료’라는 토스트를 띄움. |
| Failure | ‘수정 실패’라는 토스트를 띄우고 다시 내 글 목록 화면으로 돌아감. 이유로는 네트워크 문제 등이 있는데 만약 로그인 세션 만료로 인해 실패했다면 로그인 화면으로 돌아감. |

## 12.4 Category & Search System

* + 1. **카테고리 선택**

|  |  |
| --- | --- |
| User’s Action | 사용자가 카테고리 목록에서 원하는 카테고리를 선택하여 클릭. |
| System’s Action | 사용자가 선택한 카테고리 정보를 가져와서 데이터베이스에 질의하여 해당 카테고리에 속하는 게시글 목록을 가져옴. |
| Success | 해당 카테고리에 속하는 게시글 목록이 리스트 뷰를 통해 띄워짐. |
| Failure | 다시 카테고리 목록 화면으로 돌아감. 이유로는 네트워크 문제 등이 있는데 만약 로그인 세션 만료로 인해 실패했다면 로그인 화면으로 돌아감. |

* + 1. **‘검색’ 버튼 누르기**

|  |  |
| --- | --- |
| User’s Action | 사용자가 검색 키워드를 입력하거나 선택하고 ‘검색’ 버튼을 누름. |
| System’s Action | 사용자가 입력하거나 선택한 검색 조건을 데이터베이스에 질의하여 해당 조건을 만족하는 게시글 목록을 가져옴. |
| Success | 해당 게시글 목록을 리스트 뷰를 통해 띄워줌. |
| Failure | 조건이 잘못되었다면 ‘잘못된 조건입니다’라는 토스트를 띄우고 전 화면으로 돌아감. 그 외에도 네트워크 문제나 로그인 세션 만료 등의 이유가 있는데 로그인 세션이 만료되었다면 로그인 화면으로 돌아감. |

## 12.5 Product & Post System

* + 1. **‘작성’ 버튼 누르기**

|  |  |
| --- | --- |
| User’s Action | 사용자가 제품 글을 작성하기 위해 ‘작성’버튼을 누름. |
| System’s Action | 사용자의 데이터를 받아오고, 기본적인 글 양식이 있는 화면을 표시함. |
| Success | 기본적인 글 양식이 있는 화면이 정상적으로 표현되며, 글 작성과 사진을 업로드 할 수 있는 상태로 변함. |
| Failure | 글 작성 화면이 표시되지 않음. 이유로는 네트워크 문제, 또는 로그인을 하지 않고 ‘작성’버튼을 눌렀기 때문. 이때는 로그인 페이지로 넘어 감. |

* + 1. **‘제출’ 버튼 누르기**

|  |  |
| --- | --- |
| User’s Action | 사용자가 제품 글을 작성한 후 제출을 위해 ‘제출’버튼을 누름. |
| System’s Action | 제품 데이터를 업로드하고, 작성이 완료되었다는 화면이 나옴. |
| Success | 작성이 완료되었다는 내용이 나오고 몇 초 후 이전 작성이전 페이지로 돌아 감. ‘마이페이지’에서 작성 글 확인가능. |
| Failure | 글 작성 완료 화면이 표시되지 않음. 이유로는 네트워크 문제, 또는 필수 기입란에 작성하지 않은 경우. 또는 너무 많은 양의 사진을 업로드 했을 경우도 포함. 이때는 다시 작성 페이지로 돌아 감. |

* + 1. **보고싶은 제품 누르기**

|  |  |
| --- | --- |
| User’s Action | 사용자가 보고싶은 제품 글을 누름. |
| System’s Action | 제품 데이터를 가져와서, 데이터를 바탕으로 제품 글을 만들어서 보여줌. |
| Success | 제품 글이 정상적으로 나오며, 작성자가 작성했던 그대로 사용자가 확인가능. 또한 작성자의 연락처 및 간략한 주소 또한 표시됨. |
| Failure | 제품 글이 표시되지 않음. 이유로는 네트워크 문제, 또는 로그인을 하지 않고 제품 글을 눌렀기 때문. 이때는 로그인 페이지로 넘어 감. |

## 12.6 AR Mode System

* + 1. **‘배치’ 버튼 누르기**

|  |  |
| --- | --- |
| User’s Action | 사용자가 현실세계에 배치하고 싶은 제품을 배치하기위해 ‘배치’버튼을 누름. |
| System’s Action | 제품 데이터를 가져와서, 데이터를 바탕으로 3D model을 만든다. 만든 3D model을 카메라가 인식한 환경에 배치하고 배치한 화면을 보여준다. |
| Success | 3D model이 정상적으로 생성되며, 인식한 환경에 알맞게 배치된다. |
| Failure | 3D model이 정상적으로 생성되지 않아 제품의 크기대로 형성되지 않거나, 인식된 환경에 3D model이 비정상적으로 표시되어 크기를 가늠할 수 없는 상태. |

## 12.7 AR Measure System

* + 1. **‘측정’ 버튼 누르기**

|  |  |
| --- | --- |
| User’s Action | 사용자가 제품을 편리하게 측정하기 위해 ‘측정’버튼을 누름. |
| System’s Action | 카메라로부터 주변환경을 가져온다. 한점 Position A를 받아오고 시작점으로 줄모양의 3D model을 만든다. 그 뒤 다른 점 Position B를 받아온 뒤, 길이를 계산하여 DB에 업로드한다. |
| Success | 3D model이 정상적으로 생성되며, 지정한 점이 사용자가 지정한 위치에 지정된다. |
| Failure | 3D model이 비정상적으로 생성되며, 사용자가 의도한 지점에 점이 찍히지 않는다. Product DB에 데이터가 제대로 들어가지 않는다. |

# Development Environment

## 13.1 Objective

Development Environment 에서는 ‘ARIT’ 이라는 하나의 시스템을 개발하기 위해서 필요한 Front-end, Back-end의 개발 환경과 외부 API에 대해서 서술한다.

## 13.2 Front-End Environment

1. **Android Studio**



**[Figure 3] Logo of Android Studio**

‘ARIT’은 application을 기반으로 한 개발을 목표로 하기 때문에, 그중에서도 보편적으로 사용되는 Android 플랫폼, 그 중에서도, Android Studio IDE를 사용하여 Front-End 개발을 하도록 한다. 안드로이드 스튜디오(Android Studio)는 안드로이드 및 안드로이드 전용 어플 제작을 위한 공식 통합 개발 환경(IDE)이다. 2013년, 구글 I/O 컨퍼런스에서 구글의 제품 관리자인 Ellie Powers에 의해서 발표되었다. 오픈소스로서, Apache 라이선스 2.0으로 배포된다.

1. **Java**



[Figure 4] Logo of JAVA

‘ARIT’ 시스템의 Front-End 개발 언어로는 Android Application 개발에서 주로 사용하고, Front-end 개발에 사용하게 될 IDE, Android Studio에서 지원하는 JAVA로 결정하였다. JAVA는 객체 지향적인 프로그래밍 언어로, 현재 웹 어플리케이션 개발에 가장 많이 사용하는 언어 가운데 하나이고, 모바일 기기용 소프트웨어 개발에도 널리 사용되고 있다. 현재 Android Studio에서는 JAVA와 Kotlin이 주로 사용되는데, 현재 팀원들이 사용 가능한 언어인 JAVA를 본 시스템 Front-End 개발에 사용한다.

## 13.3 Back-End Environment

1. **Firebase**



[Figure 5] Logo of Firebase

‘ARIT’ 의 Back-End System에서 데이터 베이스를 관리하고, 정리된 데이터 베이스에서 원하는 data를 search하는 작업은 구글에서 제공하는 Firebase를 이용한다. 본 시스템의 Front-End 개발에 사용되는 Android Studio와 쉽게 연동하여 데이터 베이스를 관리할 수 있는 Firebase를 본 시스템의 데이터 베이스를 관리하는 오픈소스 실시간 채팅 어플리케이션으로 선택하였다.

1. **AR Core**



[Figure 6] Logo of AR Core

본 시스템에서는 AR 환경을 구현하기 위해서 AR Core라는 SDK(Software Development Kit)를 사용한다. AR CORE는 구글에서 개발한 AR 환경을 구축하기 위한 소프트웨어로, 안드로이드 스튜디오와의 원활한 연동을 위하여 위와 같은 프로그램을 사용하기로 결정하였다.

## 13.4 APIs and Tools

1. **Github**



[Figure 7] Logo of GitHub

본 시스템의 원활한 개발을 위한 개발자들끼리의 communication을 위하여 Github을 사용한다. Github는 분산 버전 관리 툴인 Git을 사용하는 프로젝트를 지원하는 웹호스팅 서비스이다. 원래 명령어 시스템인 Git을 Web과 GUI Application으로 편리하게 이용할 수 있다. 각각의 팀원들이 개발하는 subsystem을 각각의 branch로 생성하여 subsystem을 나누어 개발한다.