Design Specification

학우하구 : 대학생 중고 거래 챗봇

Team #3

2016311867	김우연
2017311656	윤응구
2015311849	이동환
2016312107	문경진
2014313932	전국민

Contents

1. Preface

1.1 Objective

이 장에서는 본 문서의 독자들과 문서의 전반적인 구조에 대해 서술한다. 그리고 문서 버전 관리 정책, 버전 업데이트 기록, 문서 변경 사항을 정리한다.

1.2 Readership

본 문서의 독자는 소프트웨어 엔지니어, 클라이언트 엔지니어, 고객 기술 지원을 위한 서비스 팀 등 이번 시스템 제작 및 유지보수에 관련된 모든 구성원으로 정의한다.

1.3 Document Structure

A. Preface

문서의 전체적인 개요에 대해 설명한다. 개요로는 예상 독자와 문서의 전반적인 구조, 버전 별 수정사항과 관리내용이 있다. 이를 통해 문서의 가독성을 높이고, 효율적인 관리를 가능하게 한다.

B. Introduction

시스템 설계에서 사용하는 다이어그램과 같은 모든 표현 도구에 대해 설명한다.

C. Overall System Architecture

전반적인 시스템 구조에 대해서 설명한다. 크게 프론트와 백엔드로 나누어 다이어그램을 활용하여 기술한다.

D. Backend System Architecture

Overall System Architecture의 백엔드 부분을 토대로 구체적으로 기술한다.

E. Frontend System Architecture

Overall System Architecture의 프론트 부분을 토대로 구체적으로 기술한다.

F. Protocol Design

Subsystem 이 상호 작용하는 과정과 방식에 대해 서술한다.

G. Database Design: ER Diagram, SQL

요구사항 명세서의 Database Requirement를 기반으로 데이터베이스 구조를 개선한 후, 다이어그램을 활용하여 구체적으로 작성한다.

H. Testing Plan

초기 의도와 같게 시스템이 작동하는지 확인하고, 결함을 찾기 위해 testing을 진행한다. 이 장에서는 Test Policy와 Test Case에 대해서도 기술한다.

I. Development Plan

실제 개발 단계에서 사용하는 기술과 개발 환경을 설명하고, 개발 일정을 기술한다.

J. Index

본문에서 사용된 그림, 표, 다이어그램 등의 색인을 기술한다.

1.4 Version of Document

A. Version Format

버전 번호는 Major number 와 Minor number 로 이루어져 있으며 (Major number).(Minor number)의 형식으로 표현한다. 문서의 버전은 1.0 부터 시작한다.

B. Version Management Policy

1) Design Specification을 수정할 때마다 버전을 업데이트 한다.

2) 단, 변경 간의 간격이 1시간 이내일 때에는 하나의 업데이트로 간주한다. 즉, 버전 번호를 변경하지 않는다.

3) 이미 완성된 파트를 변경할 때에는 Minor number를 변경한다.

4) 새로운 부분을 추가하거나 문서의 구성이 예전에 비해 명시적인 변화가 있을 경우 Major number를 변경한다.

Version	Modified Date	Explanation
1.0	2019.11.01	- 문서 목차 작성
2.0	2019.11.03	- Preface/Introduction/Architecture 작성
3.0	2019.11.04	- Protocol Design, Database Design 작성
3.1	2019.11.05	- Preface 목차 보충
4.1	2019.11.7	- Testing Plan, Development 작성 - Preface 최종 정리 - Index 작성

표1. Version of the Document

2. Introduction

2.1 Objective

시스템 설계에서 사용하는 다이어그램과 같은 모든 표현 도구에 대해 설명한다.

2.2 Applied Diagram

A. UML

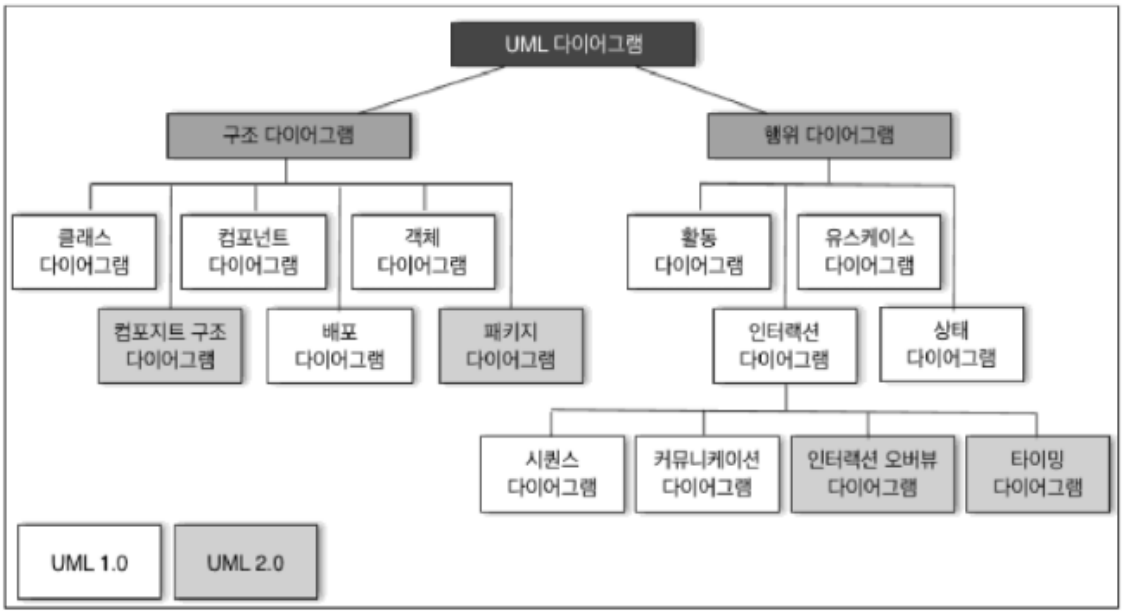


그림 1. UML 분류

통합 모델링 언어 (Unified Modeling Language, 이하 UML)는 객체 지향 소프트웨어 설계를 위해 사용되던 여러 종류의 다이어그램들을 합하여 만든 모델링 표기법으로, 시스템 개발 과정에서 개발자 간의 의사소통이 원활하게 이루어지게 하기 위한 객체 지향적 분석과 설계 방법론의 표준 지정을 목표로 하고 있다. 1994년 소프트웨어 방법론의 선구자인 Grady Booch, James Rumbaugh, Ivar Jacobson에 의해서 연구되었고, 1997년 객체관리그룹(OMG, Object Management Group)에서 객체 모델링 기술 (OMT; object modeling technique), OOSE 방법론 등을 통합하여 UML을 발표하였다.

B. Package Diagram

도메인에 대해 더 깊이 이해할수록 개념적 모델이 더 복잡해지고 크기가 커질 수 있다. 이렇게 복잡해진 시스템을 이해하기 위해서는 추상적인 개념들을 하나의 그룹으로 묶어서 용이하게 할 수 있다. 이렇게 만든 그룹들은 각각 class와 같은 개념으로 많은 인스턴스를 가지고, 오로지 시스템을 이해하기 위한 목적으로만 존재하게 된다. 이러한 그룹을 패키지(Package)라고 하는데, 패키지를 사용하면 전체 모델을 크기가 보다 작고 관리하기 쉬운 하위 집합으로 나눌 수 있다. 패키지는 UML 시스템 모델의 기본 구성 요소로, 요소들을 그룹으로 조직하기 위한 범용 메커니즘으로써 시스템 모델의 요소들을 조직하고 이해할 수 있도록 해준다. 패키지에 안에 담기는 것은 클래스에만 국한되지 않고, 하위 패키지들과 Use Case, Activity Diagram 등도 담을 수 있으며, 패키지의 표시 여부는 물론 패키지가 포함하는 요소의 표시 여부를 설정할 수 있다. 패키지를 구성할 때에는 여러 사람이 동의할 수 있는 형태로 구성되어야 하며, 패키지의 구성과 이름 체계는 개발자들이 쉽게 이해하고 사용할 수 있어야 한다. 패키지 내부의 모든 클래스들은 다양한 기능적 측면에서 유사한 면을 가진다. 패키지 내부의 클래스들은 서로 밀접한 관련성을 가지며, 다른 패키지의 클래스들과는 의존관계가 약하다. 이와 같이 패키지 다이어그램은 패키지 삽입 및 패키지 확장을 포함하여 모델 요소들을 패키지로 그룹핑 함으로써 조직 및 요소의 독립성을 묘사하고, 요소들 간의 관계를 시각화 하여 제공한다.

C. Deploy Diagram

배치 다이어그램(Deployment Diagram)은 네트워크, 하드웨어 또는 소프트웨어들을 실행 파일 수준의 컴포넌트들과 함께 표현한 다이어그램이다. 이들은 시스템을 구성하는 하드웨어간의 연결 관계를 표현하고, 하드웨어 자원에 대한 소프트웨어 컴포넌트의 배치 상태를 표현한다. 즉, 시스템이 실행되는 환경인 노드와 그 노드에 배치된 컴포넌트의 구성, 각 노드들 사이의 네트워크 특성이나 프로토콜과 같은 관계를 나타내는 다이어그램이다. 노드는 처리 능력을 가진 장치를 의미하며, 각각의 노드에는 프로세서나 디바이스들에 대한 사항을 표현하고 Deployment Diagram에서 직육면체로 표기한다. Deployment Diagram은 시

스템의 설계 단계의 마지막에 작성되는데, 이는 모든 설계가 마무리되어야 소프트웨어의 컴포넌트가 정의되고, 시스템의 하드웨어 사양도 확정되기 때문이다. 배치 다이어그램은 소프트웨어 시스템이 배치, 실행될 하드웨어의 자원들을 정의하고, 소프트웨어의 컴포넌트가 어떤 하드웨어 자원에 탑재되어 실행될지를 정의하는 목적을 가지고 있다. D

D. Class Diagram

가장 많이 사용되는 다이어그램이다. 시스템에서 사용되는 객체 타입을 정의하고, 그들 간의 존재하는 정적인 관계를 다양한 방식으로 표현한다. 즉, 시스템의 정적인 관점들을 가시화하고 구축을 위한 자세한 내용을 명세화한다. 클래스, 인터페이스, Collaboration 간의 관계를 나타내며, 객체지향 시스템 모형화에서 가장 공통적으로 많이 쓰인다.

E. State Diagram State Diagram

객체지향 모델링에서 클래스의 인스턴스 건(Event)에 의거한 시스템의 전체적인 작동을 상세히 기술하는 다이어그램이다. State Diagram은 상태의 변화에 의한 동작 또는 하나의 상태에서 다른 상태로 변화되게 하는 사건의 주어진 시간 동안의 상태를 나타낸다. 다시 말해, State Diagram은 상태와 상태의 변화를 포함하는데, 이러한 표기는 실제 구현에서 UI를 정의하는 데 도움을 준다. 상태 다이어그램은 어떤 이벤트에 대한 반응적인(Reactive) 특성을 가지는 객체에 대해 기술하기 위해 이용되며, 객체가 갖는 여러 가지 상태를 표현한다. 객체는 기존 상태에 따라 동일한 메시지에 대해서 다른 행동을 보이기 때문에, 상태 다이어그램을 통해 시스템 내의 객체가 가질 수 있는 상태가 어떤 것이 있는지에 대해, 또 각 상태를 나타낼 때 특정 이벤트에 대해 어떤 반응을 보일지에 대해 기술한다.

F. Sequence Diagram Sequence Diagram

시스템 내에서의 각 컴포넌트들이 주고받는 메시지의 흐름을 시간 순차적으로 표현하는 상호작용 다이어그램이다. 상호작용 다이어그램은 다이어그램의 수직 방향이 시간을 나타내고, 메시지 교류의 주체인 객체와, 객체를 통해 실질적인 데이터를 주고받는 메시지(오퍼레이터)를 보여주는 역할을 하며, 그 구성 요소로는 Actor, 객체, 메시지, 회귀 메시지, 제어 블록 등이 있다. Sequence Diagram은 각 컴포넌트들의 상호작용이 명확히 보이기 때문에 각 컴포넌트간의 관계와 각 컴포넌트들이 갖고 있는 속성, 행동들을 더욱 명확히 할 수

있다. Sequence Diagram은 시스템 내부의 동적인 움직임을 표현해주는 다이어그램이기 때문에, 속성 및 함수로 이루어진 Class Diagram을 동적 프로그래밍으로 설계하는 중요한 과정이다.

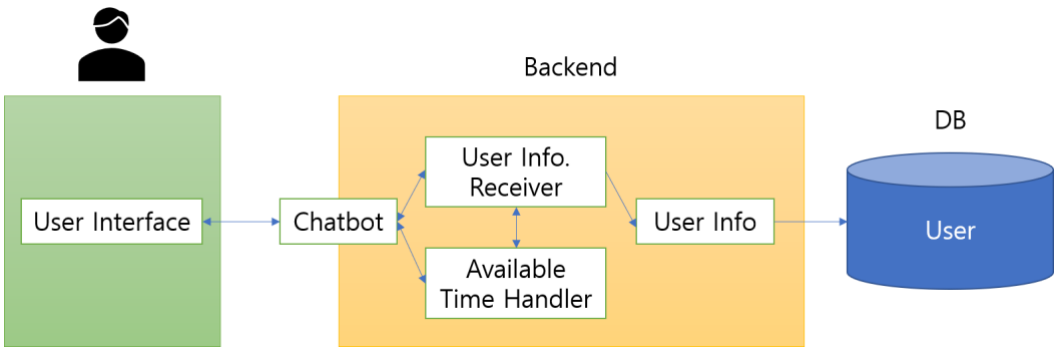
G. ER Diagram

ER Diagram은 데이터베이스에서 구조화된 각 데이터 개체들의 관계를 표현하기 위해 사용하는 다이어그램으로, UML에 포함되지는 않는다. 데이터가 저장되는 공간인 데이터베이스의 구조 및 그에 수반한 제약 조건들은 다양한 기법에 의해 정의될 수 있는데, 그 기법 중 하나가 개체- 관계 모델링(Entity-Relationship Modelling)이다. 이 ER모델링 프로세스의 산출물이 ER Diagram(Entity-Relationship Diagram)인데, 여기에서 개체(Entity)란 현실 세계의 객체로서 유형 또는 무형의 정보를 대상으로 서로 구별할 수 있는 것이며, 관계(Relationship)란 두 개 이상의 개체 사이에 연관성을 기술한 것이다. 따라서, ER Diagram은 조직의 정보 자원(Resource)을 전반적으로 계획하는데 있어서 필수적이며 유용한 도구이다.

2.3 Project Scope

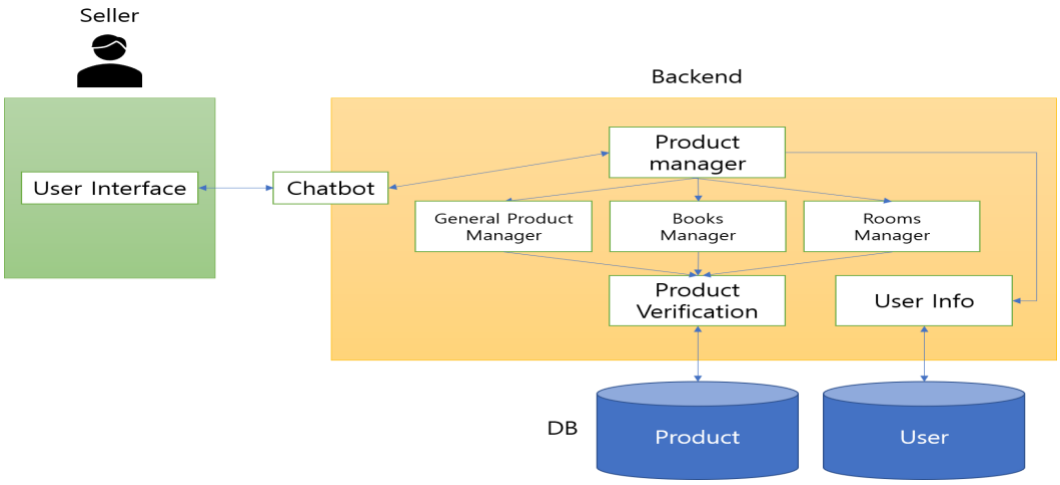
대학로봇 중고나라(대학생 중고 거래 챗봇) 서비스는 카카오에서 제공하는 챗봇을 기반으로 사용자에게 편리한 중고거래를 제공한다. 그에 따라 크게 User Subscription System, Seller System, Buyer System으로 나뉜다.

먼저 User Subscription System은 판매자와 구매자 모두 서비스를 이용하기 전 가입하여 정보를 저장하기 위한 시스템이다. 사용자로부터 대학 정보를 받아 DB에 저장하며, 여기에는 사용자의 거래 가능 시간 정보가 포함된다.



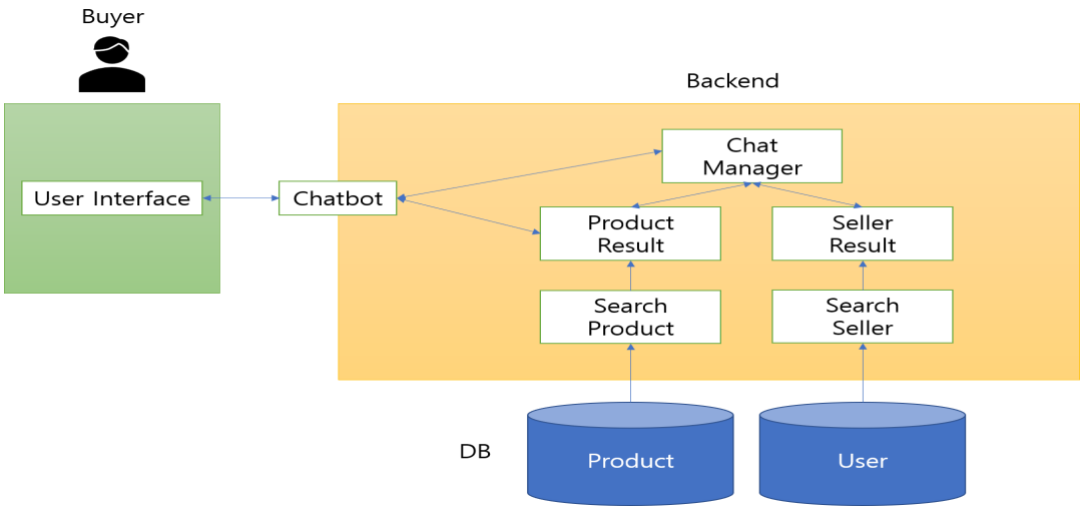
<그림 3. 가입 시스템 구조>

Seller System 의 경우, 판매자가 물품을 등록할 때에는 Product Manager를 거친다. 판매자가 입력한 정보는 종류에 따라 다르게 처리되며, 검증 절차를 거쳐 DB에 저장하기 위해 존재한다.



<그림 3. 판매 시스템 구조>

Buyer System의 경우, 구매자를 위한 시스템이다. 구매자는 상품 검색을 통해 원하는 물품을 찾을 수 있고, DB에서 결과를 검색하여 구매자에게 보여주도록 한다.



<그림4 . 구매 시스템 구조>

3. Overall System Architecture

3.1. Objectives

본 챕터에서 전체 시스템 아키텍처를 설명한다. 시스템 전체의 구조를 Frontend 부분과 Backend 부분으로 나누어 서술하고, 각 서브 시스템의 개략적인 구조 및 서브시스템 간의 관계들을 서술한다.

3.2. System Organization

프론트엔드 관련 플랫폼은 카카오 i에서, 백엔드 관련 플랫폼은 아마존 웹 서비스(AWS)에서 제공하는 것을 사용한다. API 호출 이벤트가 발생했을 때에만 코드를 실행하고, 서버리스로 API 요청을 처리할 수 있도록 했으며, Frontend Application 가 User와 모든 상호작용을 담당하고, Frontend Application 와 Backend Application은 JSON 을 기반으로 데이터를 송수신한다. 백엔드 애플리케이션은 프론트엔드로부터 들어오는 각 요청들을 컨트롤러로 분배하고, 필요한 객체 정보를 데이터베이스로부터 가져와 JSON 포맷으로 가공한 뒤 전달한다.

A. Frontend Architecture

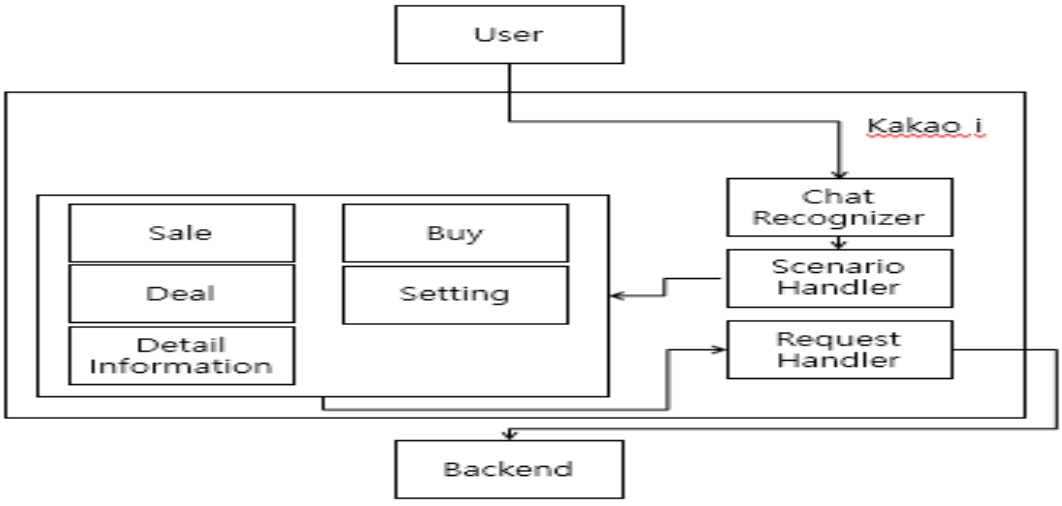


Diagram 1: System Architecture – Frontend

사용자와의 상호작용을 전담하는 시스템으로, Kakao I 프레임워크를 통해 각 컴포넌트 Sale, Buy, Deal, Setting, Detail Information 를 관리한다. User 의 요구가 Chat Recognizer를 통해 분류되어 Scenario Handler에게 적절한 컴포넌트를 선택하고, 각 컴포넌트가 백엔드와의 통신을 전담하는 Request Handler 에게 필요한 데이터를 요청하면, Request Handler는 미리 정해둔 적절한 프로토콜로 요청을 변환하여 백엔드에게 전달하여 준다.

B. Backend Architecture

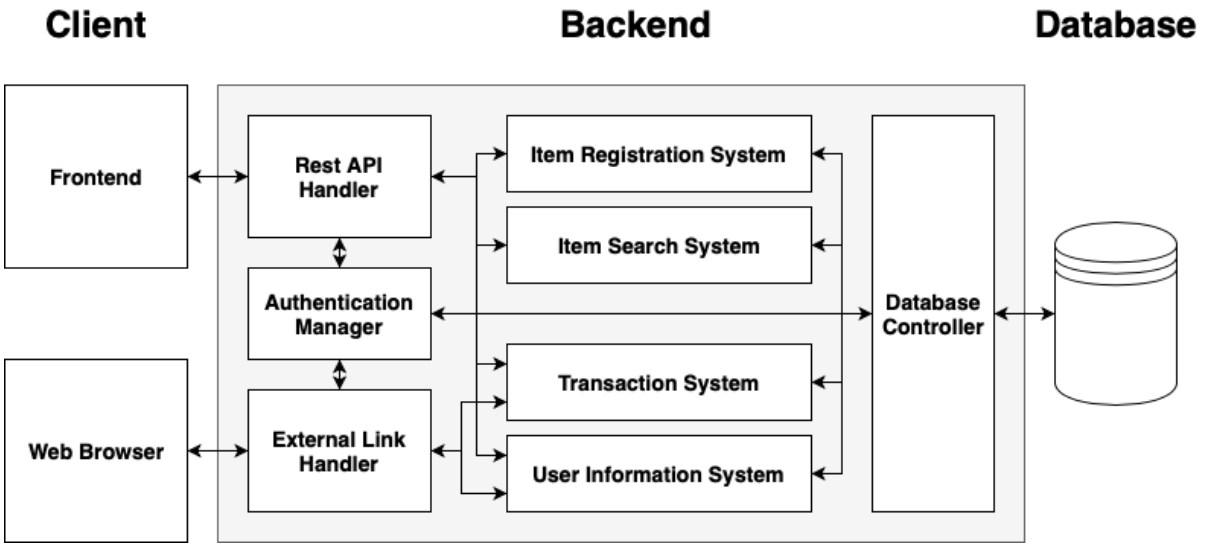


Diagram 2: System Architecture – Backend

Backend는 Frontend 또는 Web Browser에서 보낸 요청을 처리한다. 요청을 처리하기 위해서 Rest API를 이용한 요청과 Link를 이용한 요청을 처리하는 handler가 각각 존재하고 각 요청이 유효한지 확인하는 manager가 존재한다. 또한 서비스의 주요 기능을 담당하는 하위 시스템이 각각 존재하며 각 하위 시스템이 데이터베이스에 직접 접근하지 않고 데이터를 저장할 수 있게 데이터를 관리하는 controller가 존재한다.

4. System Architecture – Frontend

4.1. Objective

본 챕터에서는 User와 communication을 담당하는 Frontend Application의 구조와 각 component간의 구성과 관계를 서술한다.

4.2. Subcomponents

A. Detail Information

1. Class Diagram

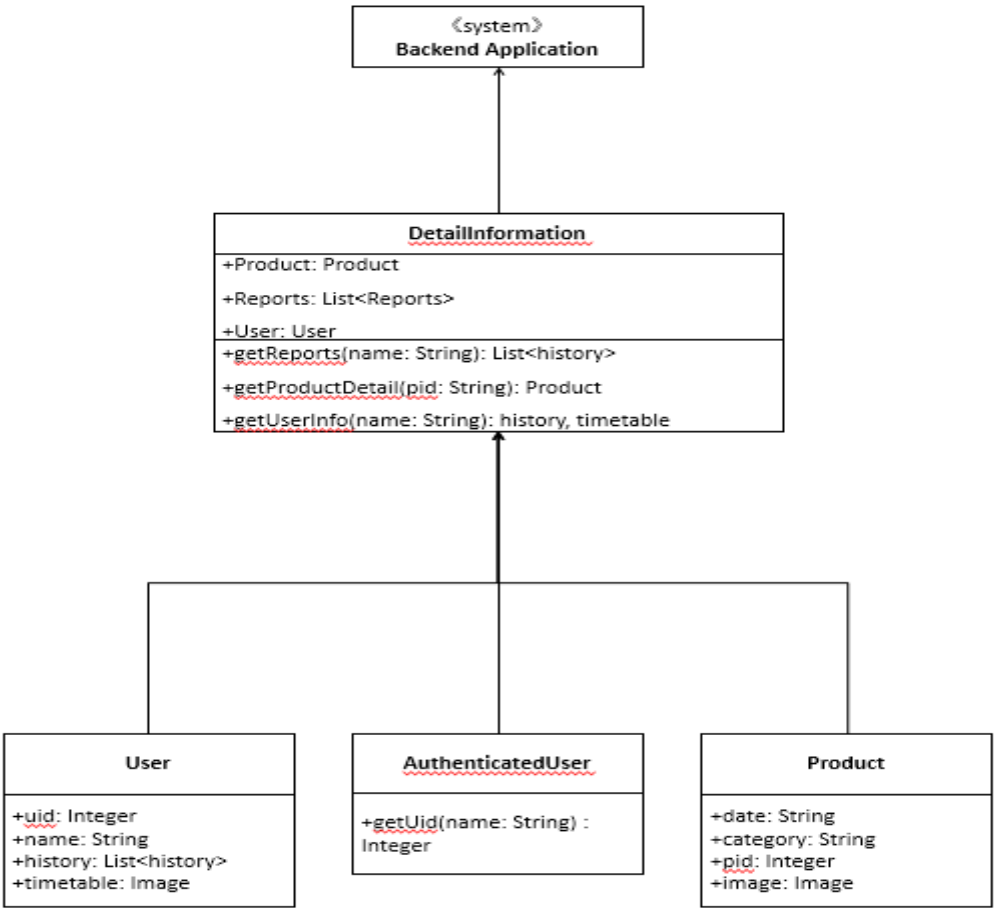


Diagram 3: System Architecture - Frontend - DetailInformation

1. DetailInformation – 상품 상세 조회 객체

A. attributes

+ Product: 현재 조회 상품 객체

+ Reports: 현재 조회 상품 판매자 판매기록

+ User: 현재 조회 상품 판매자

B. methods

+ getReports(name: String): List<history>: 판매자 판매 기록 조회

+getProductDetail(pid: String): Product: 해당 상품의 상세 정보를 조회

+getUserInfo(name: String): history, timetable: 거래를 위한 상품의 판매자의 정보

2. User – 유저 객체

+uid : User ID

+name : User Name

+history: User 거래 기록

+timetable: User 시간표

3. AuthenticatedUser – 인증 유저 객체

4. Product – 상품 객체

+date: 상품 등록 날짜

+category: 상품 등록 카테고리

+pid: 상품 id

+image: 상품 상세 이미지

B. Detail Information

1. Class Diagram

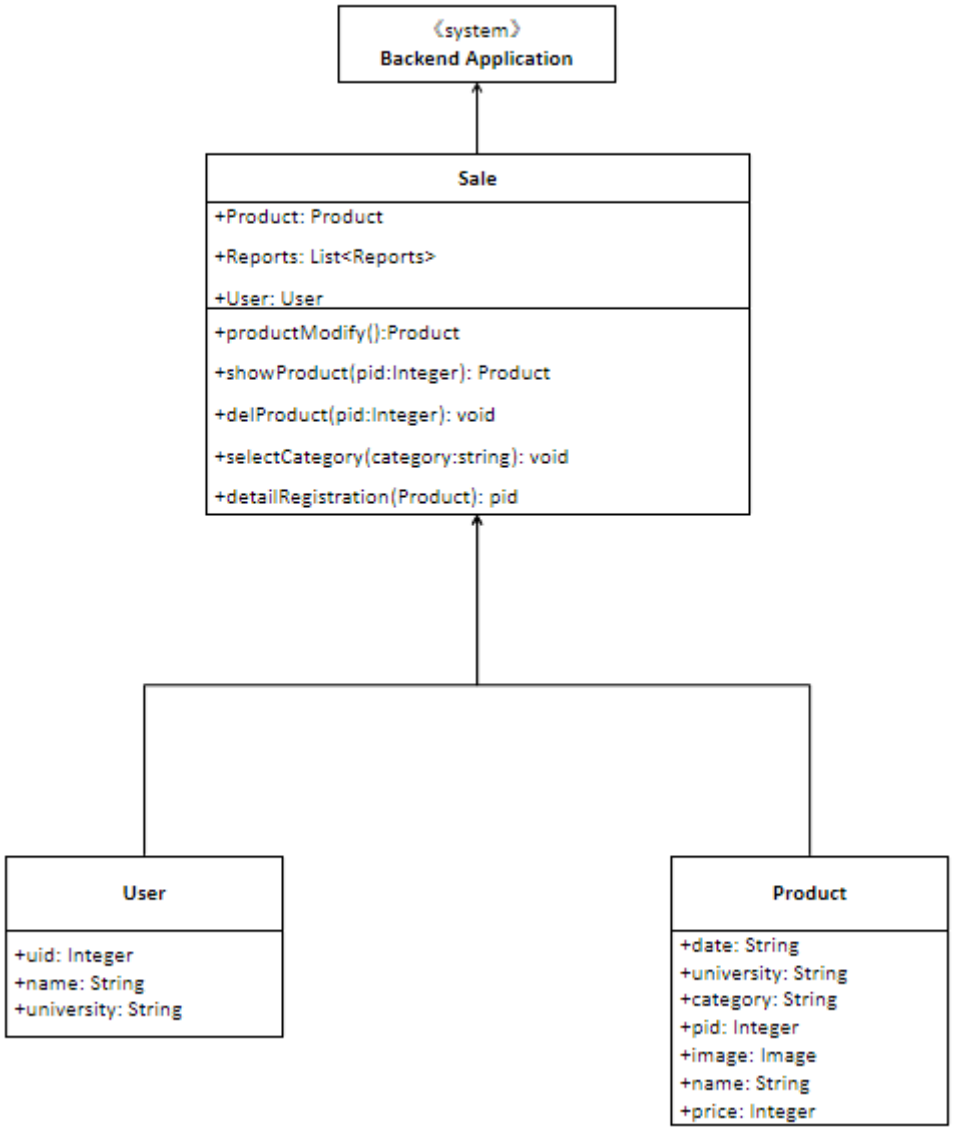


Diagram 4: System Architecture - Frontend - Sale

1. Sale – 상품 판매

A. attributes

+ Product: 판매 상품 객체

+ Reports: 판매자 판매기록

+ User: 상품 판매자

B. methods

+productModify(): 판매자 판매 상품 수정

+showProduct(pid:Integer): 현재 판매자 판매 상품 목록 열람

+delProduct(pid:Integer): 판매자 판매 상품 삭제

+selectCategory(category:string): 판매 상품 카테고리 선택

+detailRegistration(Product): 상품 상세정보 입력

2. User – 판매 유저 객체

+uid : 판매자 uid

+name : 판매자 이름

+university : 판매자 대학 정보

3. Product – 판매 상품 객체

+date: 판매 상품 등록 날짜

+university: 판매자 대학 정보

+category: 판매 상품 카테고리

+pid: 판매 물품 번호

+image: 판매 물품 이미지

+name: 판매 제목

+price: 판매 상품가격

C. Setting

1. Class Diagram

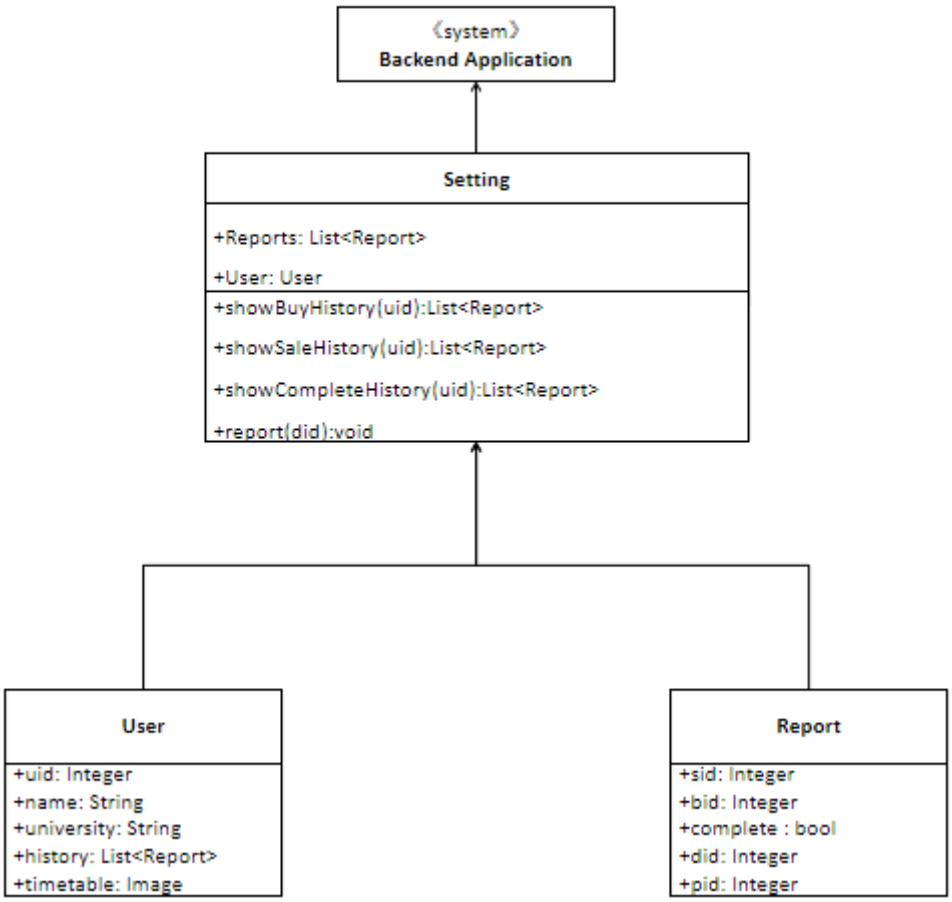


Diagram 5: System Architecture - Frontend - Setting

1. Setting : 설정

A. attributes

+ Reports: 유저 기록

+ User: 유저

B. methods

+showBuyHistory(uid): 유저 구매 기록

+showSaleHistory(uid): 유저 판매 기록

+showCompleteHistory(uid): 유저 거래 성사 기록

+report(did): 신고하기

2. User – 유저 객체

+uid : 유저 uid

+name : 유저 이름

+history: 유저 거래 기록

+university : 유저 대학 정보

3. Report – 거래 기록

+sid: 판매자 id

+bid: 구매자 id

+complete : 거래 성사 여부

+did: 거래 id

+pid: 상품 id

D. Deal

1. Class Diagram

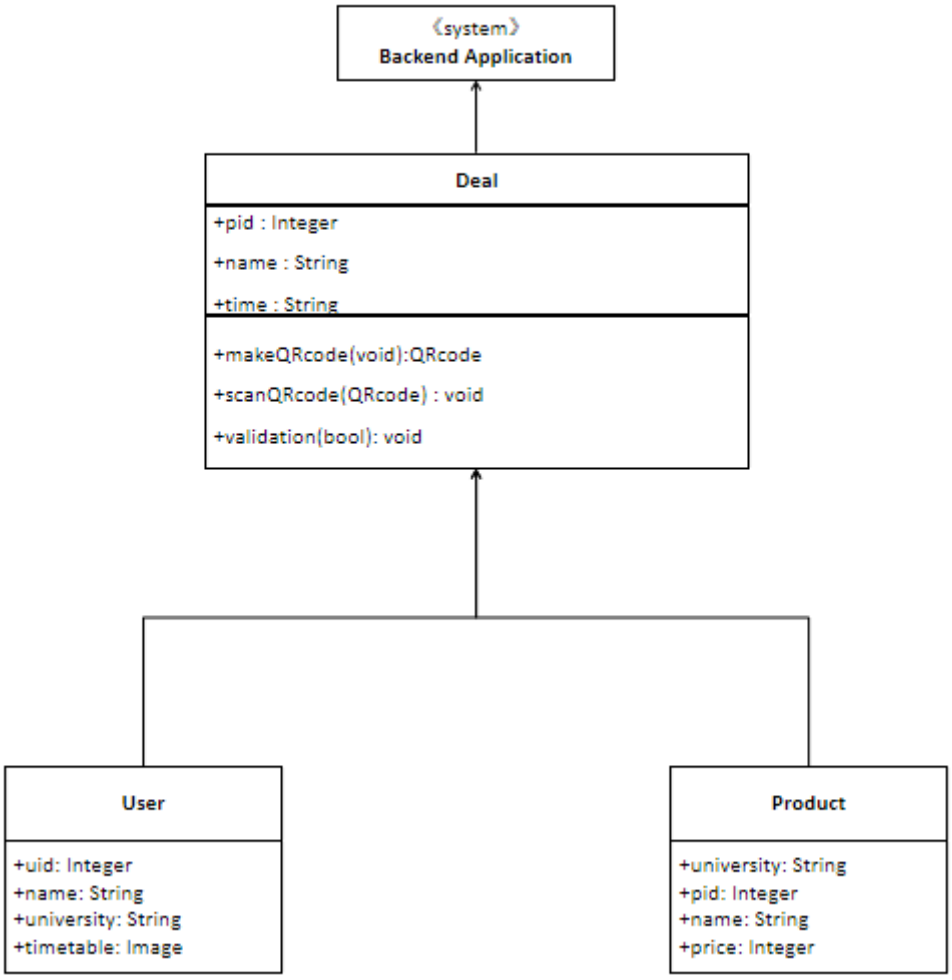


Diagram 6: System Architecture - Frontend - Deal

1. Deal : 거래

A. attributes

+ pid: 거래 물품 id

+ name: 거래 물품 제목

+ time: 거래 시간

B. methods

+makeQRcode(void): 거래 확인용 QR코드 생성

+scanQRcode(QRcode) : 거래 확인용 QR코드 인식

+validation(bool): 거래 확인

2. User – 유저 객체

+uid : 유저 uid

+name : 유저 이름

+university : 유저 대학 정보

+timetable : 거래 가능 시간표

3. Product – 상품 객체

+university: 대학 정보

+pid: 상품 id

+name: 상품 이름

+price: 상품 가격

E. Buy

1. Class Diagram

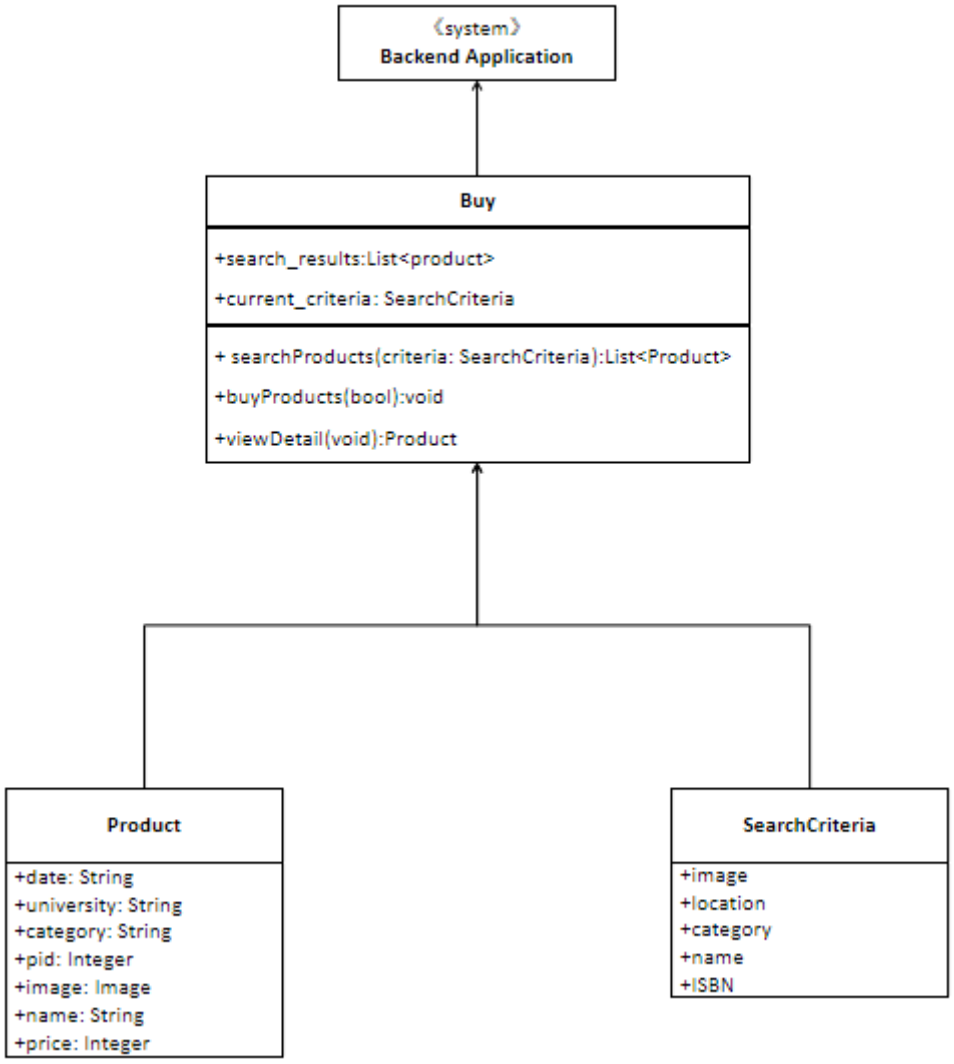


Diagram 7: System Architecture - Frontend - Buy

1. Buy : 물품 구매

A. attributes

+search_results: 검색 조건에 따른 검색 결과 상품 목록

+current_criteria: 현재 검색 조건

B. methods

+ searchProducts(criteria: SearchCriteria): 해당 검색 조건으로 상품을 검색한다.

+buyProducts(bool): 특정 상품을 구매한다

+viewDetail(void): 상품 상세 정보를 본다.

2. Product – 판매 상품 객체

+date: 판매 상품 등록 날짜

+university: 판매자 대학 정보

+category: 판매 상품 카테고리

+pid: 판매 물품 번호

+image: 판매 물품 이미지

+name: 판매 제목

+price: 판매 상품가격

3. SearchCriteria – 검색 조건 객체

+image: 이미지 검색을 통한 검색

+location: 위치 검색을 통한 검색

+category: 카테고리 분류에 따른 검색

+name: 상품 제목에 따른 검색

+ISBN: 도서 ISBN을 통한 검색

5. System Architecture – Backend

5.1. Objectives

이번 챕터에서는 Frontend와 Web Browser에서 보낸 요청을 처리하기 위한 Backend의 구조와 서비스의 각 기능을 담당하는 하위 시스템의 구조를 자세히 설명한다.

5.2. Overall Backend Architecture

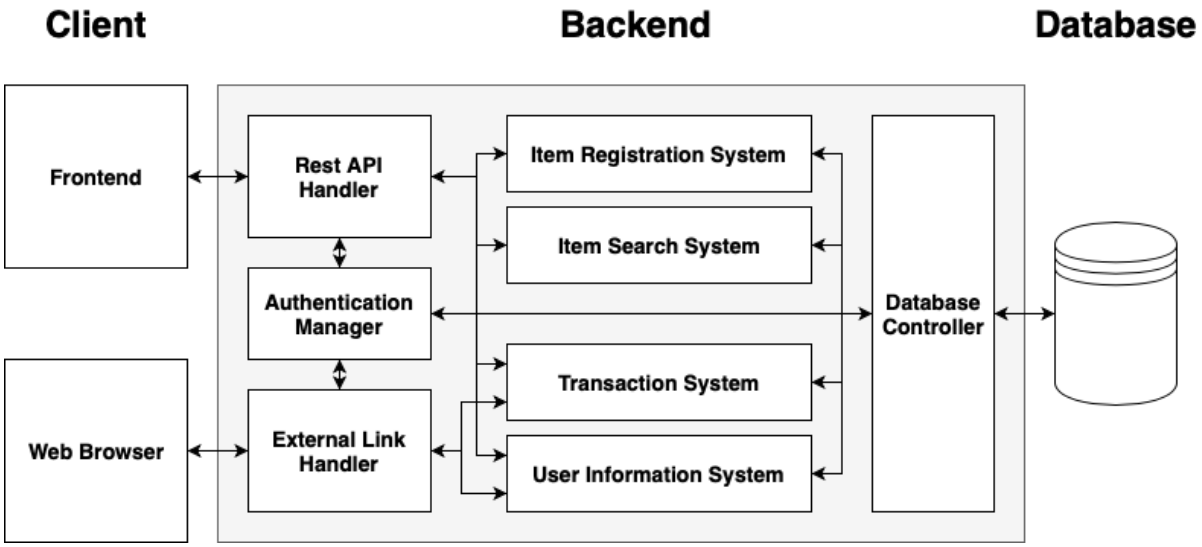


Diagram 8: System Architecture – Backend

앞선 챕터 3에서 설명한 바와 같이 Frontend 또는 Web Browser에서 요청을 보내고 Backend는 이를 처리하는 각각의 handler를 가지고 있다. 또한 요청이 유효한지 확인하는 manager가 있으며 서비스의 주요 기능을 담당하는 하위 시스템들이 존재한다. 마지막으로 하위 시스템들을 실제 데이터베이스 구조로부터 분리해주는 controller가 존재한다.

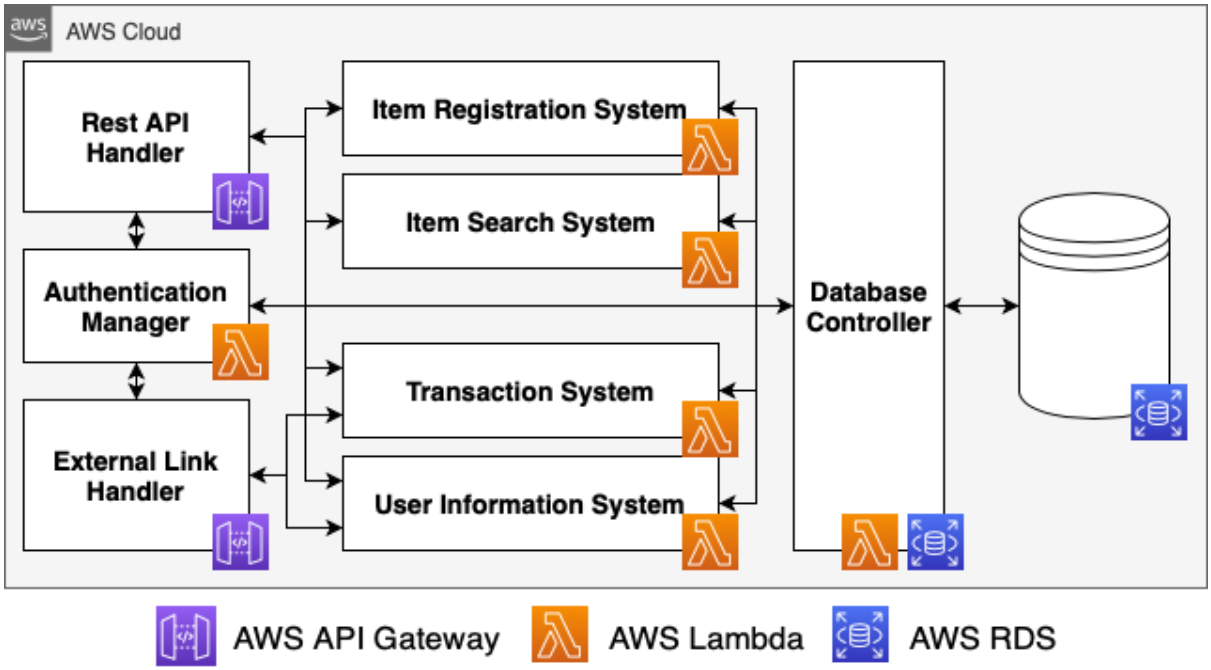


Diagram 9: Backend Architecture on Cloud Service

Backend를 구현하기 위한 방법으로 Amazon Web Services를 사용하는 것으로 결정하였다. 비용을 효율적으로 관리하기 위해서 일반적인 서버 방식의 서비스를 제공하는 AWS EC2 대신에 요청을 처리할 때만 코드를 실행시킬 수 있는 AWS Lambda를 사용한다. 각 요청들은 기본적으로 HTTP Request이므로 AWS API Gateway를 이용해 라우팅하며 요청에 맞는 Lambda 함수를 동작시키게 된다. 데이터베이스는 SQL 기반의 데이터베이스인 AWS RDS를 사용한다. 이렇게 구성된 Backend는 Serverless 구조이므로 데이터베이스를 제외한 모든 부분은 무상태적 특징을 지닌다.

5.3. Detailed Backend Architecture

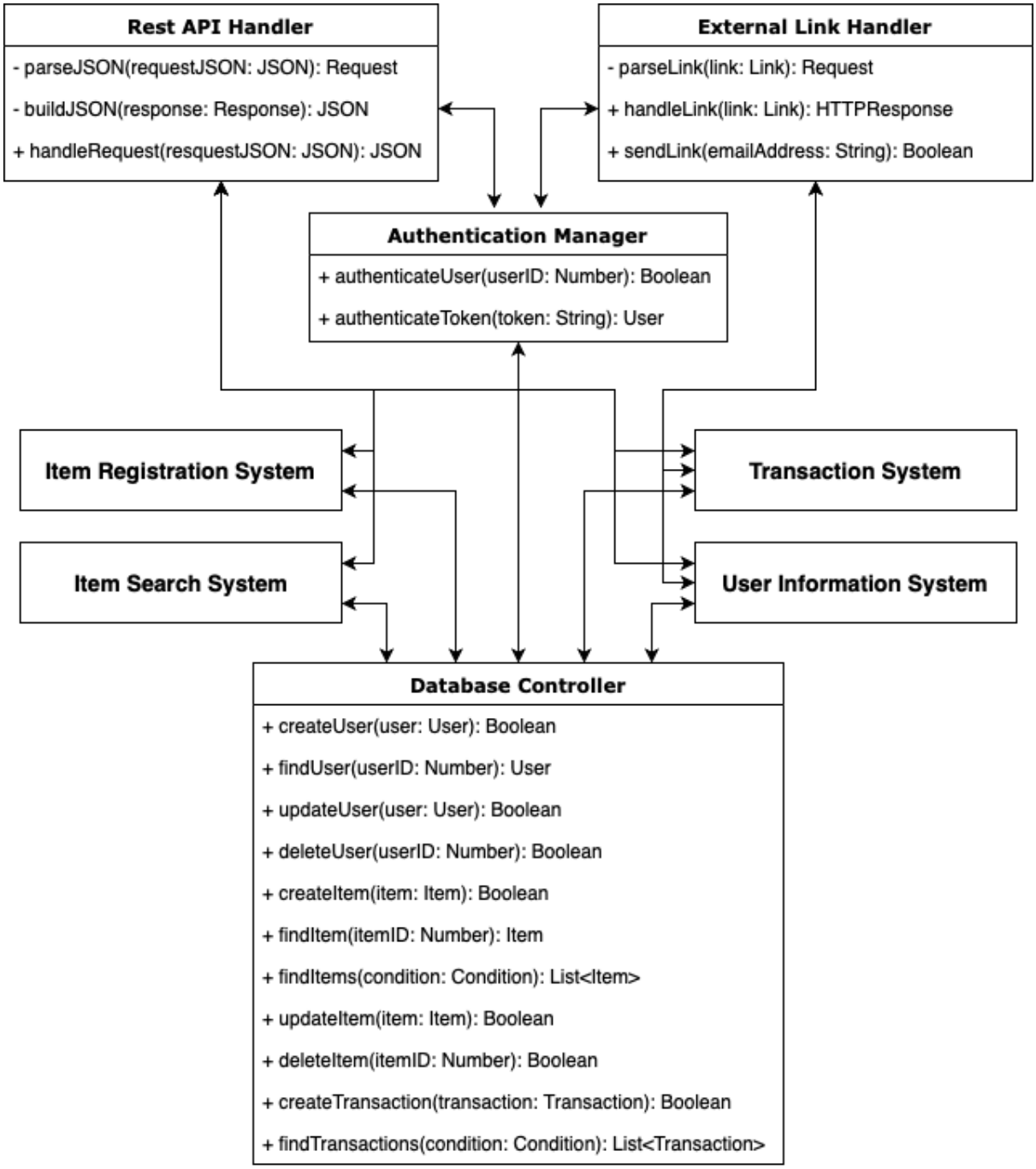


Diagram 10. Detailed Backend Architecture

1. Rest API Handler

A. parseJSON(requestJSON: JSON): 요청으로 보내진 JSON을 파싱하여 Request 객체로 변환하여 반환하는 프라이빗 메서드

B. buildJSON(response: Response): 응답으로 보낼 Response 객체를 JSON으로 변환하여 반환하는 프라이빗 메서드

C. handleRequest(requestJSON: JSON): Rest API를 이용해 보내진 요청을 분석하여 유효한지 확인한 후 적절한 하위 시스템을 이용해 요청을 처리한 후 응답을 보내는 메서드

2. External Link Handler

A. parseLink(link: Link): 요청으로 보내진 링크를 파싱하여 Request 객체로 변환하여 반환하는 프라이빗 메서드

B. handleLink(link: Link): Web Browser를 이용해 보내진 요청을 분석하여 유효한지 확인한 후 적절한 하위 시스템을 이용해 요청을 처리한 후 응답을 보내는 메서드

C. sendLink(emailAddress: String, link: Link): 사용자의 이메일로 요청 정보가 포함된 링크를 보내는 메서드

3. Authentication Manager

A. authenticateUser(userID: Number): User ID가 시스템에 등록되어 있는지 확인하고 등록 / 미등록 여부를 반환하는 메서드

B. authenticateToken(token: String): Token이 시스템이 발행했는지와 유효기간이 지나지 않았는지를 확인하고 해당 토큰을 발행받은 User를 반환하는 메서드

4. Database Controller

A. createUser(user: User): User 객체를 받아서 데이터베이스에 저장하고 성공 여부를 반환하는 메서드

B. findUser(userID: Number): User ID를 받아서 데이터베이스에서 검색하여 User 객체를 반환하는 메서드

C. updateUser(user: User): 수정할 User에 대한 User 객체를 받아서 데이터베이스를 업데이트하고 성공 여부를 반환하는 메서드

D. deleteUser(userID: Number): User ID를 받아서 데이터베이스에서 User를 삭제하고 성공 여부를 반환하는 메서드

E. createItem(item: Item): Item 객체를 받아서 데이터베이스에 저장하고 성공 여부를 반환하는 메서드

F. findItem(itemID: Number): Item ID를 받아서 데이터베이스에서 검색하여 Item 객체를 반환하는 메서드

G. findItems(condition: Condition): Item에 대한 조건을 받아서 해당 조건을 만족하는 Item들을 데이터베이스에서 검색하여 Item의 리스트를 반환하는 메서드

H. updateItem(item: Item): 수정할 Item에 대한 Item 객체를 받아서 데이터베이스를 업데이트하고 성공 여부를 반환하는 메서드

I. deleteItem(itemID: Number): Item ID를 받아서 데이터베이스에서 Item을 삭제하고 성공 여부를 반환하는 메서드

J. createTransaction(transaction: Transaction): Transaction 객체를 받아서 데이터베이스에 저장하고 성공 여부를 반환하는 메서드

K. findTransactions(condition: Condition): Transaction에 대한 조건을 받아서 해당 조건을 만족하는 Transaction들을 데이터베이스에서 검색하여 Transaction의 리스트를 반환하는 메서드

5.4. Backend Subsystem Architecture

5.4.1. Item Registration System

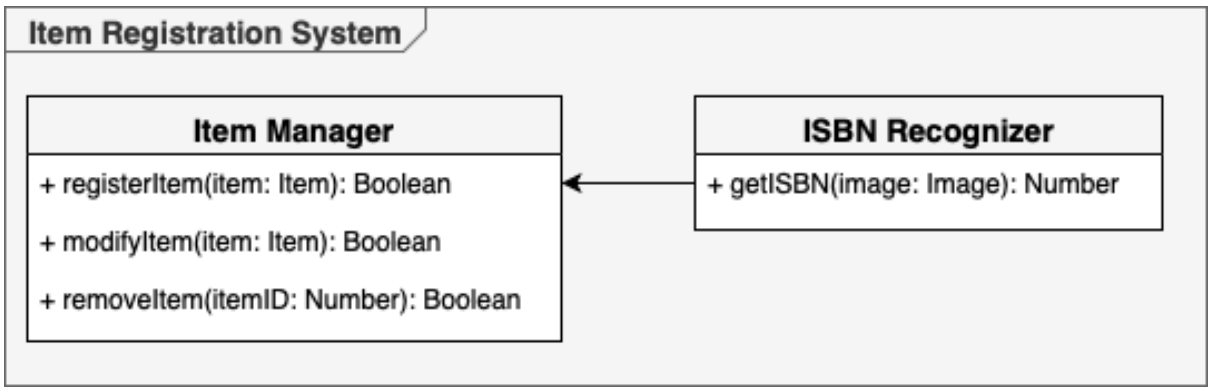


Diagram 11. Item Registration System

1. Item Manager

A. registerItem(item: Item): Item 객체를 받아서 카테고리에 따라 특화된 정보를 처리한 후 시스템에 저장하고 성공 여부를 반환하는 메서드

B. modifyItem(item: Item): 수정할 Item의 Item 객체를 받아서 시스템에 저장하고 성공 여부를 반환하는 메서드

C. removeItem(itemID: Number): 제거할 Item의 Item ID를 받아서 시스템에서 제거한 후 성공 여부를 반환하는 메서드

2. ISBN Recognizer

A. getISBN(image: Image): 사진에 있는 ISBN 번호를 검색한 후 숫자의 형태로 반환하는 메서드

5.4.2. Item Search System

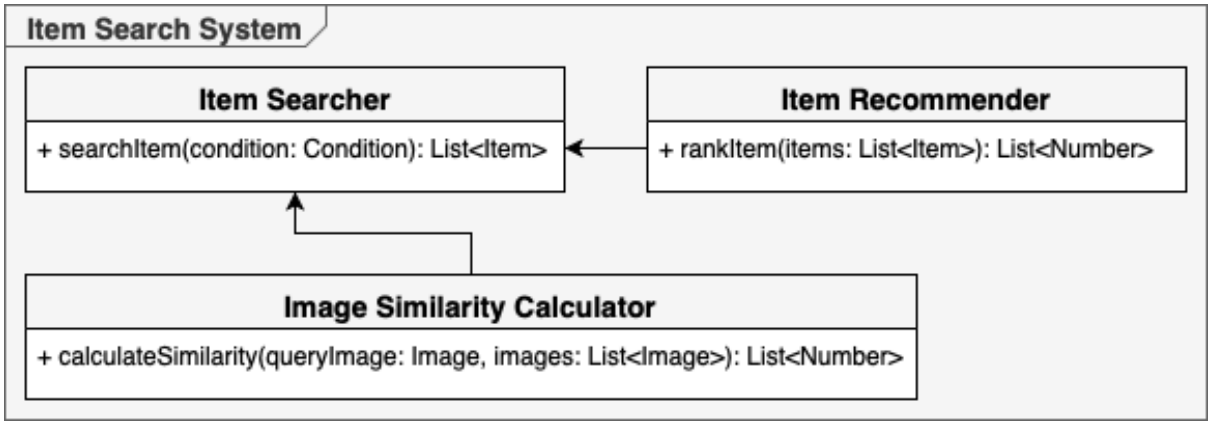


Diagram 12. Item Search System

1. Item Searcher

A. searchItem(condition: Condition): 주어진 조건에 따라 시스템에 등록되어 있는 Item을 검색하고 추천 점수가 높은 순으로 정렬해 반환하는 메서드

2. Item Recommender

A. rankItem(items: List<Item>): 특정 기준에 따라 Item에 점수를 매겨서 순위를 매긴 후 반환하는 메서드

3. Image Similarity Calculator

A. calculateSimilarity(queryImage: Image, images: List<Image>): 쿼리 이미지와 입력으로 들어온 이미지들간의 유사도를 계산한 후 유사하다고 판단되는 이미지들을 반환하는 메서드

5.4.3. Transaction System

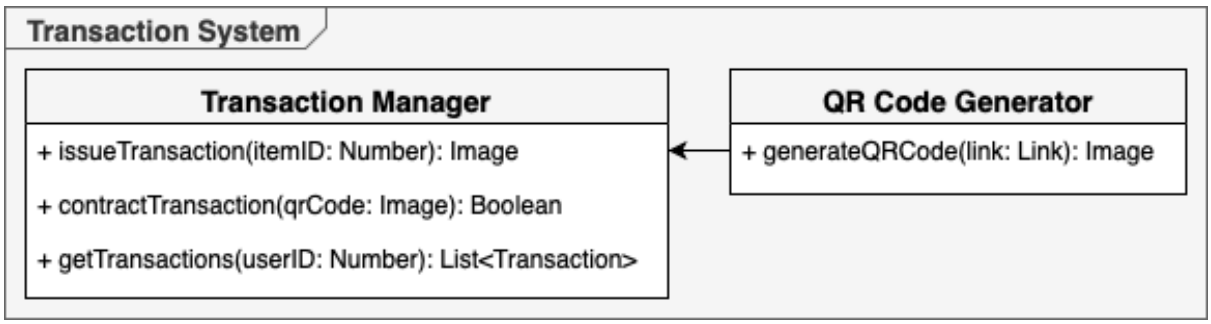


Diagram 13. Transaction System

1. Transaction Manager

A. issueTransaction(itemID: Number): 지정한 Item ID에 대한 거래를 발급하고 거래 정보를 담은 QR Code를 생성해 반환하는 메서드

B. contractTransaction(qrCode: Image): 거래 정보가 담긴 QR Code를 인식하고 거래를 체결해 시스템에 저장하고 성공 여부를 반환하는 메서드

C. getTransactions(userID: Number): 시스템에서 지정한 User ID가 참여한 거래 목록을 검색해 반환하는 메서드

5.4.4. User Information System

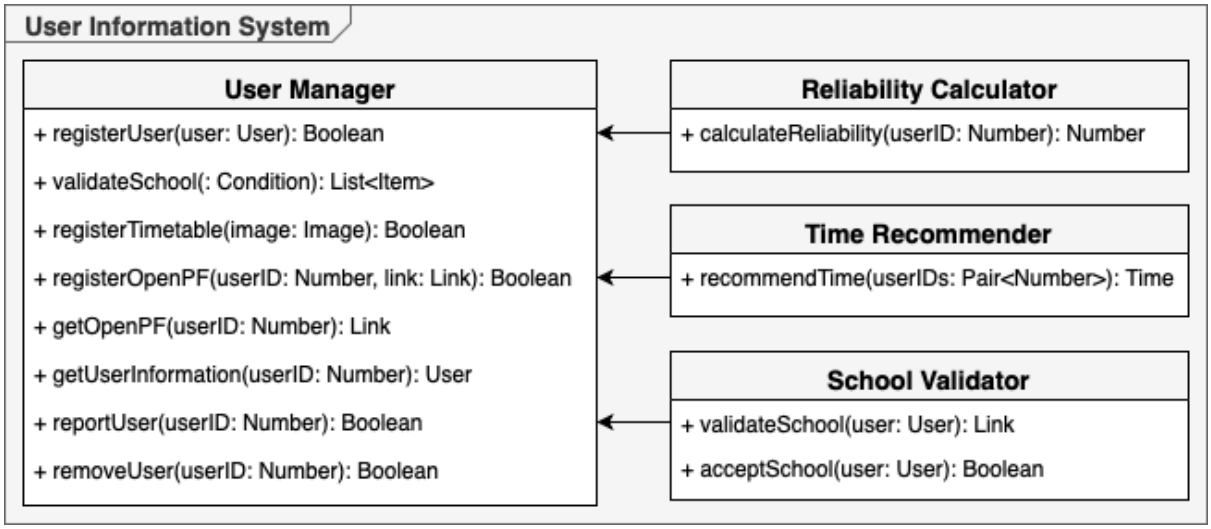


Diagram 14. User Information System

1. User Manager

A. registerUser(user: User): User 객체를 받아서 시스템에 등록하고 성공 여부를 반환하는 메서드

B. registerTimetable(image: Image): 에브리타임 시간표 이미지를 받아서 User의 시간표를 분석한 후 시간표를 텍스트 형태로 시스템에 등록하고 성공 여부를 반환하는 메서드

C. registerOpenPF(userID: Number, link: Link): User의 카카오톡 오픈프로필 링크를 시스템에 등록하고 성공 여부를 반환하는 메서드

D. getOpenPF(userID: Number): User ID에 따라 시스템에 등록되어있는 카카오톡 오픈프로필 링크를 반환하는 메서드

E. getUserInformation(userID: Number): User ID에 따라 시스템에 등록되어있는 User 정보를 반환하는 메서드

F. reportUser(userID: Number): User ID에 해당하는 User를 신고처리하고 성공 여부를 반환하는 메서드

G. removeUser(userID: Number): User ID에 해당하는 User를 시스템에서 제거하고 성공 여부를 반환하는 메서드

2. Reliability Calculator

A. calculateReliability(userID: Number): User ID에 해당하는 User의 신뢰도 점수를 계산해 반환하는 메서드

3. Time Recommender

A. recommendTime(userIDs: Pair<Number>): 두 User간의 시스템에 등록된 시간표 정보에 따라 거래 시간을 추천한 후 반환하는 메서드

4. School Validator

A. validateSchool(user: User): User의 대학교 정보에 따라 인증 링크를 생성한 후 반환하는 메서드

B. acceptSchool(user: User): 인증 링크에 접속한 유저의 대학교 정보를 인증하고 성공 여부를 반환하는 메서드

6. Protocol Design

6.1. Objective

Protocol Design에서는 서브시스템들이 상호작용하는 프로토콜에 대해 서술한다. 통신하는 메시지의 형식 및 용도, 의미를 설명한다.

6.2. JSON

JSON(JavaScript Object Notation)은 속성-값 쌍으로 이루어진 데이터 오브젝트를 전달할 수 있는 개방형 표준 포맷으로, 자료를 주고받을 때 그 정보를 표현하기 적합한 방식이다. 다양한 프로그래밍 언어에서 쉽게 다룰 수 있으며, 대부분의 자료형을 표현할 수 있다. Rest API Handler를 통해 JSON을 Request로, Response를 JSON으로 변환할 수 있다.

6.3. protocol Description

A. Overview

클라이언트와 서버 사이 전송되는 메시지의 형태를 정의한다. 클라이언트에서의 request와 서버에서의 response 메시지로 구분한다.

A. 회원가입

a. Request

Attribute	Value
Nickname	사용자의 별명
KakaoID	사용자의 카카오톡 고유번호
UnivName	대학교 이름
TermAgree	약관 동의 여부

표2. 회원가입 request

b. Response

Attribute	Value
Subscribe_success	회원가입 성공 여부

표3. 회원가입 response

B. 로그인

a. Request

Attribute	Value
Nickname	사용자의 별명
KakaoID	사용자의 카카오톡 고유번호

표4. 로그인 request

b. Response

Attribute	Value
Login_success	로그인 성공 여부

표5. 로그인 response

C. 시간표 등록

a. Request

Attribute	Value
KakaoID	사용자의 카카오톡 고유번호
Timetable	사용자의 시간표

표6. 시간표 등록 request

b. Response

Attribute	Value
Timetable_success	시간표 등록 성공 여부

표7 시간표 등록 response

D. 상품 등록

a. Request

Attribute	Value
KakaoID	사용자의 카카오톡 고유번호
ItemCategory	상품 분류

ItemName	상품 이름
ItemPrice	상품 가격
ItemImages	상품 사진
ItemDetail	상품 상세설명
ItemISBN	상품 ISBN (도서)
itemLocation	상품 위치 (원룸)

표8 상품 등록 request

b. Response

Attribute	Value
Regist_success	시간표 등록 성공 여부

표9 상품 등록 reponse

E. 도서 인식

a. Request

Attribute	Value
ItemISBNImage	도서 ISBN 이미지

표10 도서 인식 request

b. Response

Attribute	Value
ISBN_success	ISBN 인식 성공 여부
itemISBN	도서 ISBN

표11 도서 인식 response

F. 상품 조회

a. Request

Attribute	Value
Category	카테고리
Name	제품명

ISBN	ISBN
Location	위치

표12 상품 조회 request

b. Response

Attribute	Value
SN	게시글 고유번호
Title	게시글 제목
SellerID	판매자 카카오톡 고유번호
Date	작성일자

표13 상품 조회 response

G. 이미지 검색

a. Request

Attribute	Value
Image	검색 대상 이미지

표14 이미지 검색 request

b. Response

Attribute	Value
SN	게시글 고유번호
Title	게시글 제목
SellerID	판매자 카카오톡 고유번호
Date	작성일자

표15 이미지 검색 response

H. 상품 상세 정보 보기

a. Request

Attribute	Value
SN	게시글 고유번호

표16 상품 상세 정보 보기 request

b. Response

Attribute	Value
KakaoID	사용자의 카카오톡 고유번호
ItemCategory	상품 분류
ItemName	상품 이름
ItemPrice	상품 가격
ItemImages	상품 사진
ItemDetail	상품 상세설명
ItemISBN	상품 ISBN (도서)
itemLocation	상품 위치 (원룸)

표17 상품 상세 정보 보기 response

I. 판매자 신뢰도 검색

a. Request

Attribute	Value
KakaoID	판매자 카카오톡 고유번호
ItemID	상품 고유번호

표18 판매자 신뢰도 검색 request

b. Response

Attribute	Value
IDSearch_success	검색 성공 여부
IDScore	판매자 신뢰도 점수

표19 판매자 신뢰도 검색 response

J. 거래 시간 추천

a. Request

Attribute	Value
KakaoID_1	판매자 카카오톡 고유번호

KakaoID_2	구매자 카카오톡 고유번호

표20 거래 시간 추천 request

b. Response

Attribute	Value
Avail_time	추천된 거래시간

표21 거래 시간 추천 response

K. 판매자 거래 확인 QR 발급

a. Request

Attribute	Value
KakaoID	판매자 카카오톡 고유번호
ItemID	상품 고유번호

표22 판매자 거래 확인 QR 발급 request

b. Response

Attribute	Value
QRCode	거래 확인 토큰이 포함된 QR코드

표23 판매자 거래 확인 QR 발급 response

L. 구매자 거래 확인 QR 인식

a. Request

Attribute	Value
KakaoID	구매자 카카오톡 고유번호
TransactionID	거래 확인 토큰

표24 구매자 거래 확인 QR 인식 request

b. Response

Attribute	Value
Success	판매 완료 처리 여부

표25 구매자 거래 확인 QR 인식 response

M. 사용자 신고

a. Request

Attribute	Value
KakaoID	구매자 카카오톡 고유번호
TransactionID	거래 확인 토큰

표26 사용자 신고 request

b. Response

Attribute	Value
KakaoID	구매자 카카오톡 고유번호
TransactionID	거래 확인 토큰

표27 사용자 신고 response

7. Database Design: ER Diagram, SQL

7.1 Objective

데이터베이스를 디자인하기 위해, 구현하고자 하는 시스템의 특성에 따라 필요한 Entity를 구상한다. Entity들간의 관계를 ER Diagram을 통해 모델링해보고, 각 Entity의 특성을 명시한다. 모델링한 ER Diagram을 통해 Relational Schema를 디자인한 후, 이를 DB Table로 생성하기 위해 DDL Syntax를 구상해 각 Entity마다 attributes들의 column화를 구현해본다.

7.2 Diagrams

A. ER Diagram

A-1) Overall ER Diagram

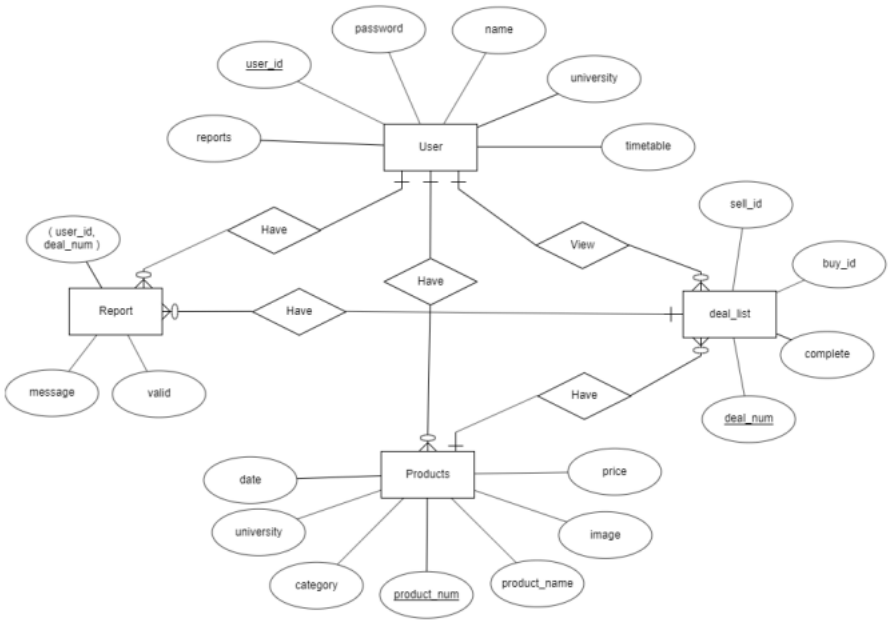


Diagram15 ER Diagram - overall

DB는 총 4개의 Entity로 분류한다, 가입 사용자의 정보를 담고 있는 User Entity, 거래 상품들의 정보를 담는 Products Entity, 거래 목록들을 갖고 있는 deal_list Entity, 그리고 사용자 간의 신고 기능을 구현할 Report Entity가 있다.

User Entity는 구현할 데이터베이스의 중심이 되는 Entity로 다른 모든 Entity들과 관계가 있다. Products와 Report Entity는 User Entity의 user_id 없이는 존재할 수 없으며, deal_list Entity 역시 sell_id와 buy_id의 명시를 위해 user_id가 참조되어야만 한다.

Products Entity는 거래 목록을 가지는 deal_list Entity에서 참조한다. product_num 중 거래 목록에 오르지 않은 것이 존재할 순 있으나, 거래 목록에선 product_num이 필수이다.

deal_list Entity는 Report Entity에서 User Entity와 함께 참조해야만 하는 Entity로, 신고 사항이 없는 deal_num이 존재할 수 있지만 모든 Report는 deal_num을 가져야만 한다.

A-2) Entities

a) User

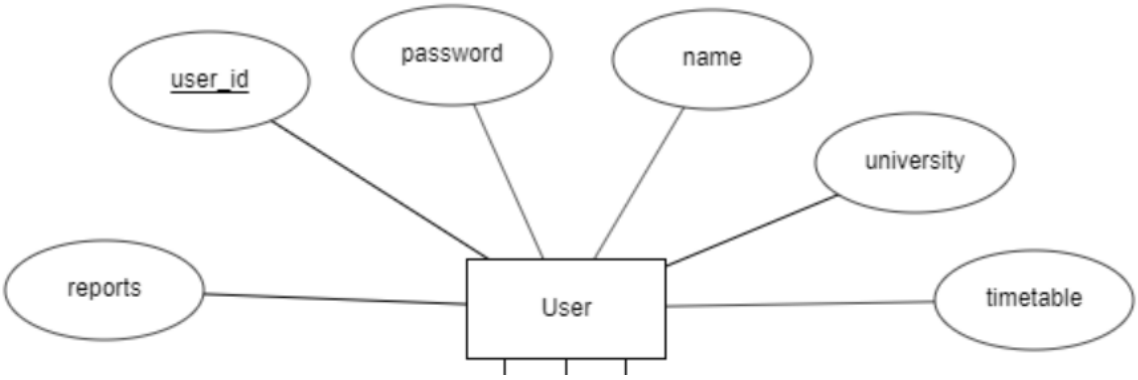


Diagram16 Entities

User Entity는 사용자의 정보를 담고 있으며 사용자의 id인 user_id를 primary key로 가진다. 사용자의 닉네임, 비밀번호, 대학, 시간표, 그리고 신고당한 횟수(reports)를 속성으로 가진다.

b) Products

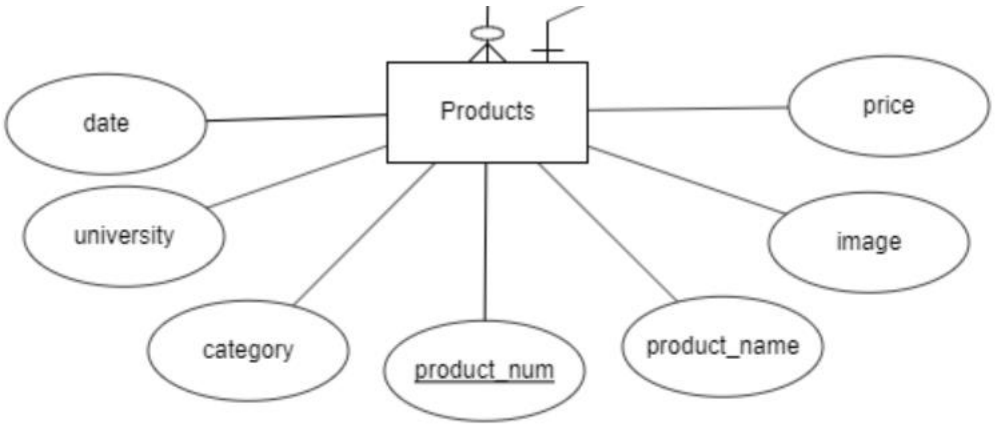


Diagram17 Products

Products Entity는 등록된 상품들을 담고 있다. 상품번호인 product_num을 primary key로 하며, 각 상품마다 판매자 id(User TABLE 참조), 상품명, 상품 카테고리, 등록 날짜, 소속 대학, 상품의 가격과 이미지를 속성으로 갖는다.

c) deal_list

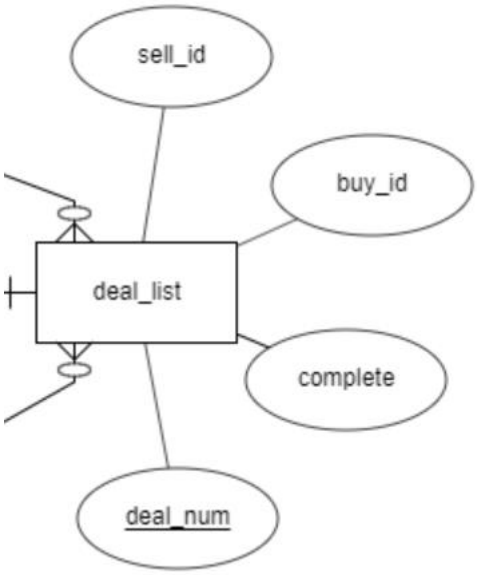


Diagram18 deal_list

deal_list Entity는 거래 목록을 담고 있는 Entity이다. deal_num을 primary key로 가지며 판매자 id, 구매자 id, 상품번호(Products TABLE 참조), 거래완료유무 정보를 속성으로 갖는다.

d) Report

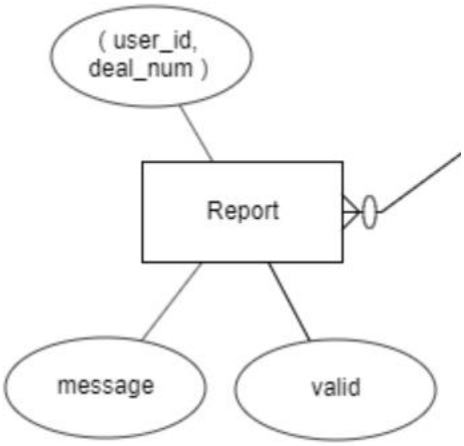


Diagram19 report

Report Entity는 신고 정보를 담고 있는 Entity로, User Entity의 user_id와 deal_list의 deal_num을 조합한 composite key를 갖는다.

B. Relational Schema

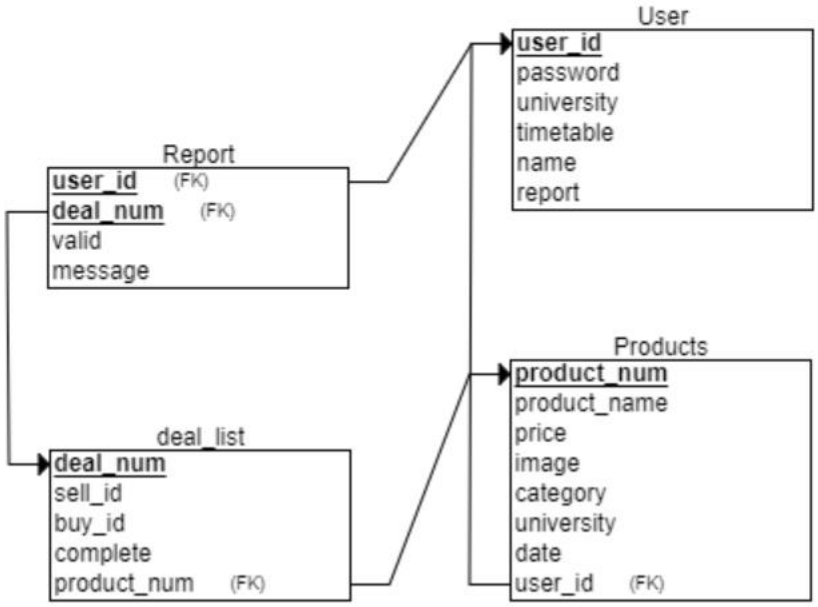


Diagram20 relational schema

작성한 ER Diagram을 바탕으로 Relational Schema를 모델링하였다. 각 Entity가 TABLE로서 구현되며 Entity의 속성들은 고유 속성과 다른 Entity를 참조하는 속성들이 합쳐져 TABLE의 column으로 구현된다. 이들 중 PRIMARY KEY와 FOREIGN KEY 역시 명시하였다.

C. SQL DDL Syntax

a) User

```
CREATE TABLE User (
    user_id VARCHAR(10) NOT NULL,
    password INT NOT NULL,
    university VARCHAR(10) NOT NULL,
    timetable VARCHAR(100) NULL DEFAULT NULL,
    name VARCHAR(10) NOT NULL,
    reports INT NULL DEFAULT '0',
    PRIMARY KEY (user_id)
)
```

그림5 user sql ddl syntax

User TABLE을 생성하는 DDL이다. timetable의 등록과 reports 횟수를 제외한 column들은 NULL값이 들어갈 수 없는 필수 입력값이고, user_id를 PRIMARY KEY로 한다.

b) Products

```
CREATE TABLE Products (
    product_num INT NOT NULL,
    product_name VARCHAR(30) NOT NULL,
    price INT NOT NULL,
    image VARCHAR(100) NOT NULL,
    category VARCHAR(20) NOT NULL,
    university VARCHAR(10) NOT NULL,
    date DATE NOT NULL,
    user_id VARCHAR(10) NOT NULL,
    PRIMARY KEY (product_num),
    FOREIGN KEY (user_id) REFERENCES User(user_id)
)
```

그림6 products sql ddl syntax

Products TABLE은 모든 column들이 값을 가져야 한다. PRIMARY KEY로 product_num, User TABLE의 user_id를 외래키로 참조한다.

c) deal_list

```
CREATE TABLE deal_list (
   deal_num INT NOT NULL,
   sell_id VARCHAR(10) NOT NULL,
   buy_id VARCHAR(10) NOT NULL,
   complete INT NULL DEFAULT '0',
   product_num INT NOT NULL,
   PRIMARY KEY (deal_num),
   FOREIGN KEY (product_num) REFERENCES Products(product_num)
   )
```

그림7 deal_list sql ddl syntax

deal_list TABLE의 생성이다. complete은 거래의 완료 여부를 나타내는 column으로 기본값이 0이고 거래 완료 시에 1로 변경된다. deal_num을 순차적인 PRIMARY KEY로 가질 것이며 Products TABLE의 product_num을 외래키로 참조한다.

d) Report

```
CREATE TABLE Report (
   message VARCHAR(200) NULL DEFAULT NULL,
   valid INT NULL DEFAULT '0',
   user_id VARCHAR(10) NOT NULL,
   deal_num INT NOT NULL,
   PRIMARY KEY (user_id, deal_num),
   FOREIGN KEY (user_id) REFERENCES User(user_id),
   FOREIGN KEY (deal_num) REFERENCES deal_list(deal_num)
   )
```

그림8 report sql ddl syntax

Report TABLE은 user_id와 deal_num을 COMPOSITE KEY로 PRIMARY KEY가 된다. 각각 User TABLE, deal_list TABLE을 참조한다. message column은 사용자의 신고 내용이 비어있을 수 있기에 기본값으로 NULL을 갖는다. valid는 신고의 유효 여부를 나타낼 것으로 기본값

0을 갖고 관리자의 판단으로 결정된다.

8. Testing Plan

8.1. Objectives

시스템의 의도한 대로 실행되는지 확인하며, 시스템 내부의 오류를 바로잡기 위해 테스팅을 수행한다. 이를 위해서는 설계 단계에서부터 계획하는 것이 필요하며, testing plan에서는 testing policy와 test case에 대해 기술한다.

8.2. testing policy

시스템의 개발에 있어서 크게 세 단계로 나누어 테스팅을 실행한다. Developing testing, Release testing, User Testing 단계로 나뉘며, developing testing은 각 과정에 따라 component testing, integrating testing, system testing, acceptance testing의 단계로 나뉜다.

1) Developing Testing
   개발 과정에서 수행된다. Component testing은 각 컴포넌트 단위로 개발이 진행되었을 때 각 요소들이 개발 후에 목표한 대로 작동하는지 확인하는 절차이다. Integrating testing은 컴포넌트 요소들을 하나씩 점진적으로 합치면서 수행하는 테스팅 절차이다. System testing은 subsystem을 합한 후 전체가 잘 동작하는지 확인하는 과정이며, Acceptance testing은 사용자 정보를 이용하여 시스템에 대한 사용자의 요구사항을 테스팅한다.

2) Release Testing
   릴리즈 전 최종 시스템을 테스팅한다. 요구사항 명세서에 작성된 요구사항이 제대로 반영되었는지 확인 과정을 거친다.

3) User Testing
   사용자가 사용자 환경에서 시스템을 테스팅한다.

8.3. Test Case

A. User Subscription System

A.1. 가입

1) 사용자 : 카카오톡 계정을 통해 가입을 시도한다.

2) 시스템 동작 : DB에 저장되어 있는 데이터와 사용자 정보를 비교하여 중복 여부를 확인한다.

2-1) 가입 성공

시스템 동작 : 해당 아이디로 가입이 진행된다.

2-2) 가입 실패

시스템 동작 : 가입이 진행되지 않는다.

시스템 알림 : '이미 가입한 계정입니다'

A.2. 로그인

1) 사용자 : 카카오톡 계정을 통해 로그인을 시도한다.

2) 시스템 동작 : DB에 저장되어 있는 사용자 데이터와 로그인한 계정 정보의 일치 여부를 확인한다.

2-1) 로그인 성공

시스템 동작 : 해당 계정으로 로그인이 진행된다.

시스템 알림 : '로그인에 성공하였습니다'

2-2) 로그인 실패

시스템 동작 로그인 실패 알림을 띄운 후, 회원가입을 진행할지 묻는다.

A.3. 시간표 등록

1) 사용자 : 에브리타임 시간표 사진을 업로드한다.

2) 시스템 동작 : 시간표 사진을 이미지 처리하여 여유 시간을 텍스트 형태로 추출한 후 DB에 저장한다.

B. Seller System

B.1. 상품 등록

1) Seller : 상품 등록 버튼을 선택한다.

2) 시스템 동작 : 등록하고자 하는 제품의 정보를 요청한다.

3) Seller : 상품 카테고리를 입력한다.

4-1) 도서인 경우 시스템 동작 : ISBN 자동입력 지원을 묻는다.

4-2) 원룸인 경우 시스템 동작 : 위치 정보를 묻는다.

5) Seller : 추가 정보를 기입한다. (일반 정보/ISBN 이미지/위치 정보)

6) 시스템 동작 : 입력 정보를 검증한 후, DB에 저장한다.

B.2. 상품 수정

1) Seller : 상품 정보 수정을 요청한다.

2) 시스템 동작 : 변경 정보를 DB에 갱신한다.

C. Buyer System

C.1. 상품 조회 및 검색

1) Buyer : 상품 정보를 입력한다.

2) 시스템 동작 : 해당 상품과 비슷한 상품의 리스트를 보여준다

3) Buyer : 특정 상품을 선택한다.

4) 시스템 동작 : 상품의 상세 정보를 보여준다.

C.2. 판매자 신뢰도 조회

1) Buyer : 신뢰도를 알아보고자 하는 판매자의 정보를 입력한다.

2-1) 조회 성공

시스템 동작 : 판매자의 신뢰도를 보여준다.

2-2) 조회 실패

시스템 알림 : '유효하지 않은 사용자입니다'

D. 거래 System

D.1. 거래 시간 추천

1) 거래 당사자(Buyer, Seller) : 당사자가 거래 상대의 정보를 입력한다.

2) 시스템 동작 : 등록된 시간표 정보를 바탕으로 공통의 여유시간을 보여준다.

D.2. 판매자 거래 확인 QR 발급

1) Seller : 상품에 대한 거래 확인 일회용 토큰 발급을 요청한다.

2) 시스템 동작 : DB에서 상품에 대한 정보가 유효한지 확인한다.

3-1) 성공

시스템 동작 : 거래 확인 일회용 토큰을 QR코드로 만들어 반환한다.

3-2) 실패

시스템 동작 : 거래 확인 일회용 토큰을 반환하지 않는다.

시스템 알림 : '유효하지 않은 거래입니다'

D.3. 구매자 거래 확인 QR 인식

1) Buyer : 거래 확인 일회용 토큰을 인식한다.

2) 시스템 동작 : 판매 완료 처리하며, 판매자와 구매자의 거래 기록을 DB에 추가한다.

9. Development Environment

9.1 Objective

이 장에서는 실제 개발 단계에서 사용하는 기술과 개발 환경을 설명하고, 개발 일정을 기술한다.

9.2 Frontend Environment

이 장에서는 실제 개발 단계에서 사용하는 기술과 개발 환경을 설명하고, 개발 일정을 기술한다.

A. Kakao i open builder

그림 . kakao i open builder

카카오 i 오픈빌더는 카카오 인공지능 기술을 이용하여 카카오톡 채널 챗봇과 카카오미니 보이스봇을 동시에 설계할 수 있는 카카오 AI 설계 플랫폼이다. 그 중 카카오톡 채널 챗봇을 사용할 예정이다. 카카오톡 채널 챗봇이란 카카오톡 채널을 통해 제공되는 대화형 인터페이스의 챗봇(Chatbot)을 의미한다. 오픈빌더를 통해 제작한 챗봇은 카카오톡 채널 관리자센터에서 개설한 카카오톡 채널과 연결하여 이용자에게 챗봇 서비스를 제공할 수 있다.

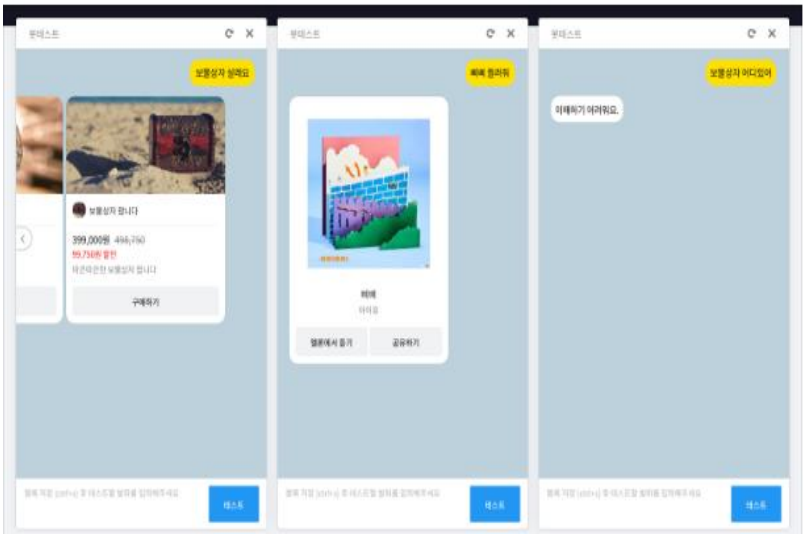
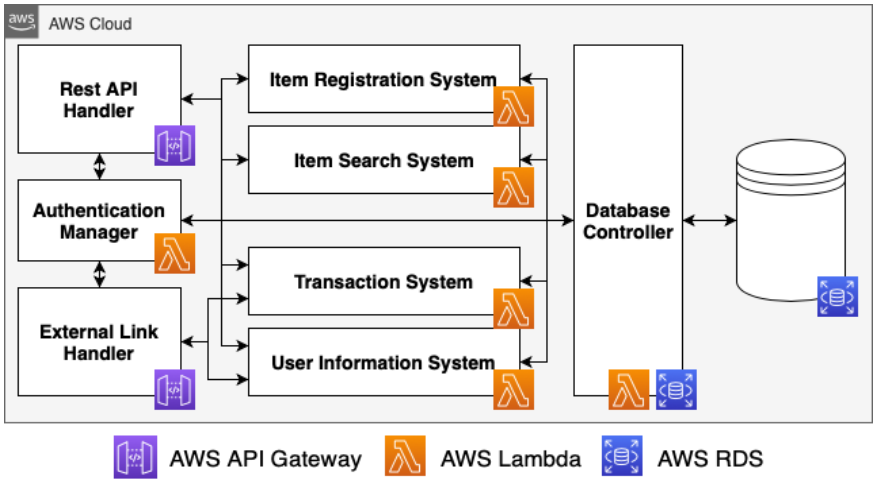


그림 2. 카카오 챗봇 구현 예시

오픈빌더에서 제공하는 기능 단위인 시나리오의 블록에 따라 사용자의 발화 분석으로 의도 파악할 수 있고, 봇이 수행할 액션과 대답을 지정할 수 있다. 또한 기본 플랫폼은 카카오톡이기 때문에 따로 UI 를 구성하지 않아도 위 그림처럼 깔끔한 디자인을 구성할 수 있어 효율적이다. 외부 API 를 이용하여 부가 서비스를 연결할 수도 있으며, 카카오 측에서 제공하는 기능을 최대로 활용할 예정이다.

9.3 Backend Environment

이 장에서는 실제 개발 단계에서 사용하는 기술과 개발 환경을 설명하고, 개발 일정을 기술한다.



아마존에서 제공하는 데이터베이스(AWS RDS)와 서버리스(AWS Lambda)를 이용하여 백엔드를 구축할 예정이다.

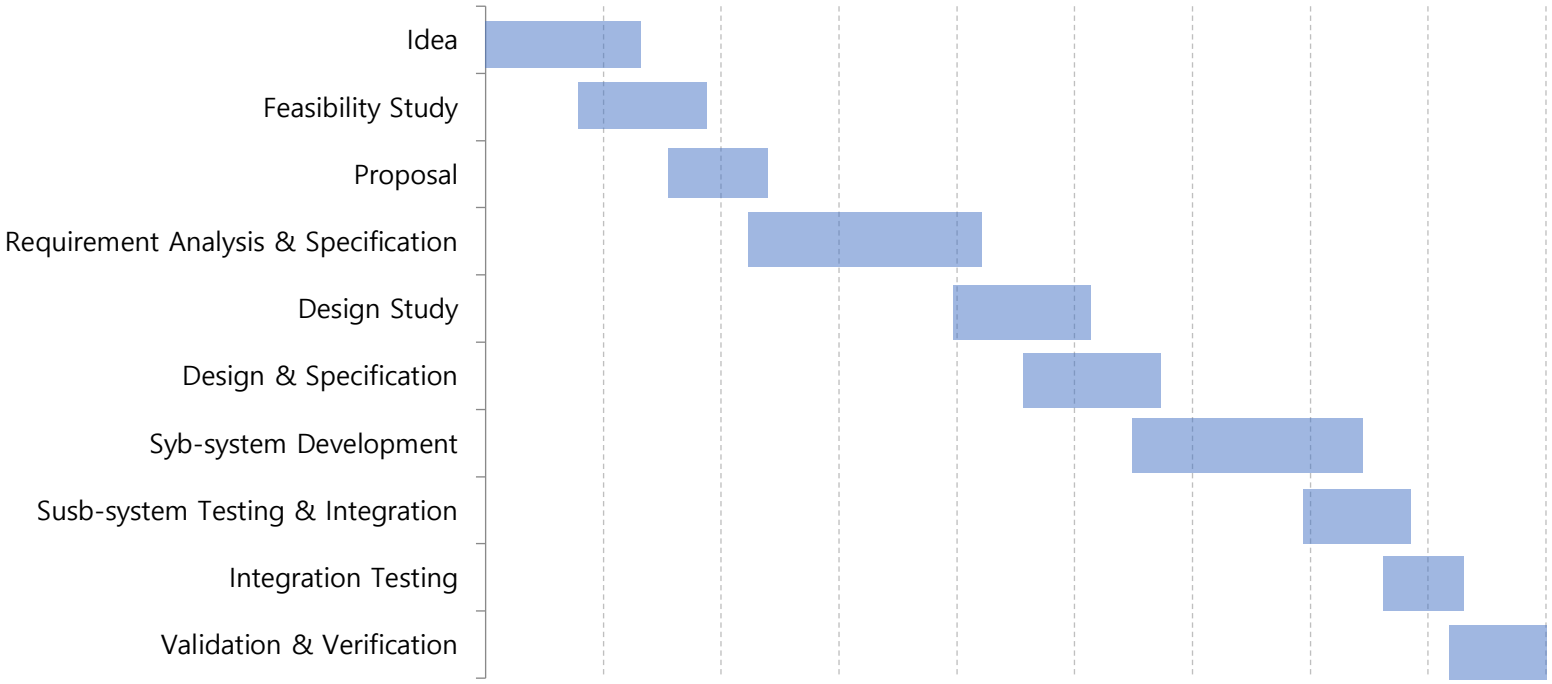


Figure 9. Gantt chart

개발 계획과 실제 상황은 위의 Gantt chart와 같다. 기존에 계획했던 것보다 요구사항 명세서 작성과 아이디어 회의에 많은 시간이 소요되었지만, 전반적인 계획에서 크게 벗어나지는 않았다.

명세화를 최대한 상세하게 작성했기 때문에, 앞으로의 개발일정은 빠르게 이루어 질 수 있을 것이다. 위의 정해진 일정 외에도 지속적으로 변경사항을 반영하도록 하고 이후 개발 일정 역시 계획에 맞게 최대한 진행할 수 있도록 노력할 예정이다.

10. Index

1.1 Objective

이 장에서는 본문에서 사용된 그림, 표, 다이어그램 등의 색인을 기술한다.