Team #1 To Dot

Design Specification

Contents

1. Preface

1.1. Objective

Preface에서는 본 문서의 독자를 정의하고, 문서의 목차를 간략하게 소개한다.

1.2. Readership

본 문서의 독자는 시스템의 설계와 개발, 유지 보수에 참여하는 모든 구성원으로 정의한다. 설계 명세서를 통해 To Dot(투닷)과 외부 시스템, 그리고 To Dot(투닷) 내부 구조간의 interaction 등 최종 시스템에 대한 대략적인 개요를 제공받을 수 있다.

1.3. Document Contents

A. Preface

Preface에서는 본 문서의 독자를 정의하고, 문서의 목차를 간략하게 소개한다.

B. Introduction

Introduction에서는 시스템 설계 시 사용하는 툴과 다이어그램, To Dot(투닷) 시스템을 이루는 7가지 시스템에 대해 개략적으로 소개한다.

C. System Architecture

System Architecture에서는 To Dot(투닷) 시스템의 전반적인 구조를 Block diagram, Package diagram, Deployment diagram을 사용하여 소개한다.

D. Sign up and Login System

Sign up and Login System은 To Dot(투닷) 서비스에 가입한 회원의 계정을 관리한다. Member DB와 연동되어 회원 가입과 로그인 기능을 제공한다. Class diagram, Sequence diagram, State Diagram을 통해 Sign up and Login System의 구조와 사용자 및 DB와의 상호 작용을 설계한다.

E. Post System

Post System은 사용자가 개인 공간에 글을 작성하는 시스템이다. 글을 업로드 할 때, 스킨 사용 여부와 커뮤니티 공개 여부를 어떻게 설정하는지에 따라 스킨 및 커뮤니티 시스템과의 상호 작용이 달라진다. 이 절에서는 Class diagram, Sequence Diagram, State Diagram을 통해 Post System의 구조와 사용자 및 DB와의 상호 작용을 설계한다.

F. Community System

Community System은 사용자들 간 커뮤니티를 통해 공감대를 형성하고 고민을 나눌 수 있는 기능이다. 본인의 글을 작성할 때 공유 여부를 선택하면, 선택된 글들에 한해서 Community Board로 업데이트가 된다. 커뮤니티의 모든 글은 익명으로 보여지고, 공유된 날짜와 시간 순서에 따라 최신의 글이 상위에 노출된다. 커뮤니티는 댓글 기능을 제공하지 않고, 쪽지(message)를 주고받을 수 있다. 아래는 Class diagram, Sequence diagram과 State Diagram을 통해 Community System의 구조를 표현하고 설명한다.

G. Send Message System

다른 사용자가 공유한 글(Post)에 답변을 보내는 작업을 말한다. 이 과정에서 ToDot 커뮤니티(Community)의 전체 공유된 글에 접근할 수 있어야 하며 원하는 글을 불러와 내용을 확인한 후 해당 글의 게시자에게 쪽지를 보낼 수 있어야 한다.

H. Evaluation and Block System

본 프로젝트에서는 게시한 글에 댓글을 달지 않고 메일이나 쪽지와 같은 형식의 답장을 보내는 것으로 글에 대한 의견을 표현하려고 한다. 이 때, 악의적인 사용자에 의한 시스템 악용을 막기 위해 사용자를 필터링하는 시스템이 필요한 것으로 판단되었고 이 과정이 사용자 간의 상호작용을 통해 자연스럽게 이뤄질 수 있는 구조를 구축하고자 하였다. 그에 따라 받은 답변에 대한 평가와 평가의 누적을 통한 차단 시스템을 구현하게 되었다.

I. Feature Analysis System

Feature Analysis System에서는 자연어 처리 및 감정 분석 API를 사용하여 게시한 글에 대한 키워드 및 감정 분석 서비스를 제공한다.

J. Cover System

Cover System을 통해 보유한 커버를 확인하고, 새로운 커버를 구매할 수 있다.

K. Protocol Design

Protocol Design에서는 하위 시스템들의 interaction에 대해 서술한다. 프로토콜의 기본 형식은 JSON으로 하며, 통신하는 메세지의 형식과 용도, 의미를 설명한다.

L. Database Design

요구사항 명세서에서 기술한 데이터베이스를 바탕으로 Database Design을 작성한다. ER Diagram과 Relational Schema의 단계를 거쳐 SQL을 작성한다.

M. Testing Plan

시스템을 테스팅하기 위한 plan을 설계 단계에서 미리 정의한다.

N. Development Environment

Development Environment에서는 사용한 프로그래밍 언어와 IDE와 같은 개발 환경에 대해 서술한다.

O. Development Plan

Development Plan에서는 Gantt chart를 통해 개발 계획에 대해 서술한다.

P. Index

Index에서는 본 문서에서 사용한 표, 다이어그램, 사진들의 인덱스를 표시한다.

2. Introduction

2.1. Objective

Introduction에서는 ToDot 시스템 설계에 사용되는 모든 종류의 diagram 및 tool에 관하여 서술한다.

2.2. Applied Diagram

A. UML

Figure 1 Unified Modeling Language(UML)

UML은 과거 개별적으로 존재하던 소프트웨어 설계 기법들을 하나로 모아 만든 통합된 소프트웨어 모델링 기법으로, 현재 객체 지향 소프트웨어 시스템을 개발하는 과정에서 각 단계의 결과를 도식적으로 시각화하고 문서화하기 위해 사용된다. 뿐만 아니라 시스템에 대한 시각적 정보를 제공함으로써 시스템 개발 과정에서 요구되는 사용자와 개발자 간의 의사소통에 크게 기여하며, 13개의 모델링 기법을 제공하기 때문에 프로젝트의 종류에 제한되지 않는다는 장점을 갖는다. 또, 각 모델이 사용하는 표기법은 해당 심벌(symbol) 마다 명확하고 고유한 정의를 가지고 있어 이를 이용하는 개발자들 간에 오류 없는 원활한 의사소통이 가능하다.

UML에서 제공하는 다이어그램(Diagram)의 종류는 총 13개로, 시스템의 구조(structure)를 도식화하는 Class Diagram, Object Diagram, Deployment Diagram, Composite Structure Diagram, Component Diagram, Package Diagram 등 6개, 시스템의 전개(flow)방식을 도식화하는 Activity Diagram, State Machine Diagram, Use Case Diagram 등 3개, 구성 요소(components) 간의 상호작용(interaction) 방식을 표현하는 Communication Diagram, Sequence Diagram, Timing Diagram, Interaction Overview Diagram 등 4개다이어그램으로 구성된다.

B. Class Diagram

Class Diagram은 클래스 내부의 정적 요소나 클래스 간의 관계를 나타내는 다이어그램으로 시스템의 부분 혹은 전체 구조를 파악하는 데 용이하다는 장점을 지닌다. 또한 Class Diagram은 각 클래스 사이의 의존 관계를 명확히 제시하는데, 만약 의존 관계 사이에 순환이 존재할 경우 이를 빠르게 파악하여 순환 관계를 파훼하고자 할 때 어떤 방법이 가장 효율적인지에 대한 판단 기준을 제공하기도 한다.

Class Diagram은 클래스와 클래스 간의 관계를 도식화하여 표기하는데, 각 클래스는 클래스 명, 클래스 속성(fields), 클래스 메소드(methods)로 구성되며 각 클래스 사이의 관계는 일반화(generalization), 의존(dependency), 연관(association) - 직접연관(directed association), 집합연관(aggregation association), 복합연관(composition) 등이 있다.

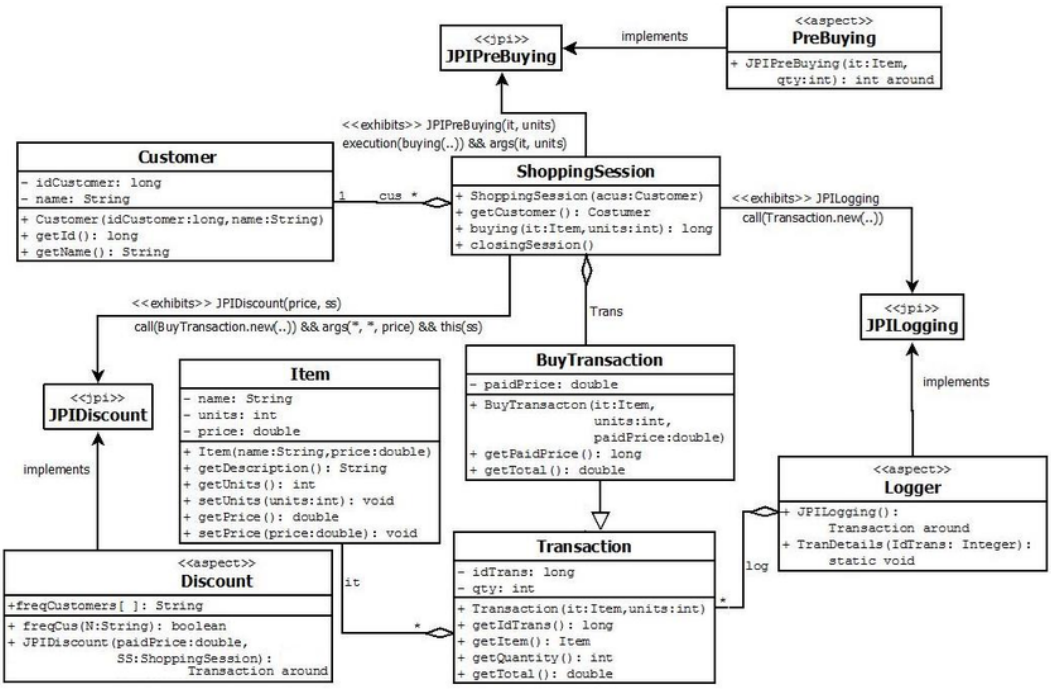

Figure 2 Example of Class Diagram

C. State Diagram

State Diagram은 한 객체의 현재 상태와 상태 변화를 도식화하여 표현하는 기법이다. 즉, 객체 하나를 대상으로 해당 객체가 생성되어 소멸되는 과정에서 다양하게 가질 수 있는 상태(state)를 분석한다. 객체의 상태는 부분적으로 매우 복잡한 상태로 변화할 수 있기 때문에 이를 분석하는 것은 중요한 의미를 지닌다고 할 수 있다. 또, State Diagram은 이러한 객체의 상태 변화를 유발하는 사건(event)를 분석하는 데 사용되기도 한다. 객체의 상태 변화는 스스로 이루어지는 것이 아니라 사건을 통해 발생하므로 그 원인이 되는 사건을 분석하고 정의하는 작업 역시 중요하다. 마지막으로 State Diagram은 각 객체가 가지는 속성(attributes)과 그 동작(operation)을 검증하는 데 사용된다. 분석 대상이 되는 객체의 상태는 속성의 값으로 정의되고, 사건은 대부분 객체의 동작으로 정의되기 때문에 Class Diagram에서 정의된 클래스의 속성과 동작 간의 적합성을 검증하는 데 용이하다.

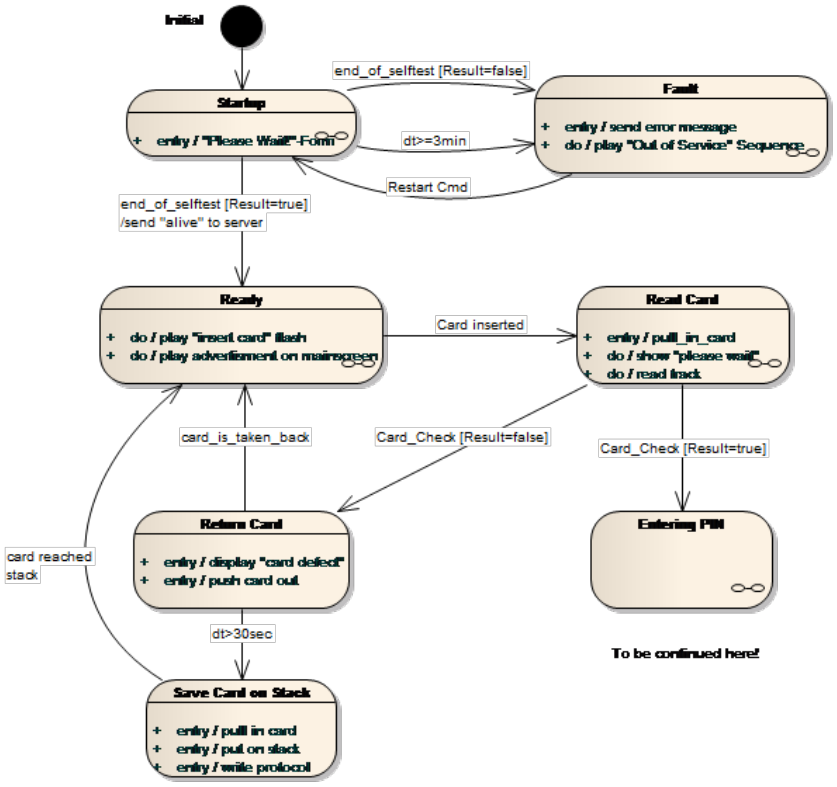

Figure 3 Example of State Diagram

D. Sequence Diagram

Sequence Diagram의 목적은 어떤 결과를 만들어내는 사건(event)의 시퀀스(sequence)를 정의하는 것이다. 즉, 어떤 메시지가 발생했는가 보다는 해당 메시지가 발생하는 과정 혹은 순서에 집중한다. 대부분의 Sequence Diagram은 시스템 내부 객체들 간에 어떤 메시지들이 전달되는지, 어떤 순서로 전달되는지를 수직 축(axis)과 수평 축을 기준으로 표기한다. 수직 축에서는 메시지가 발생한 시간 순서를 축의 상단에서 하단으로 진행하는 방향으로 나타내며, 수평 축에서는 화살표의 방향에 따라 메시지의 송신객체와 수신객체를 표기한다.

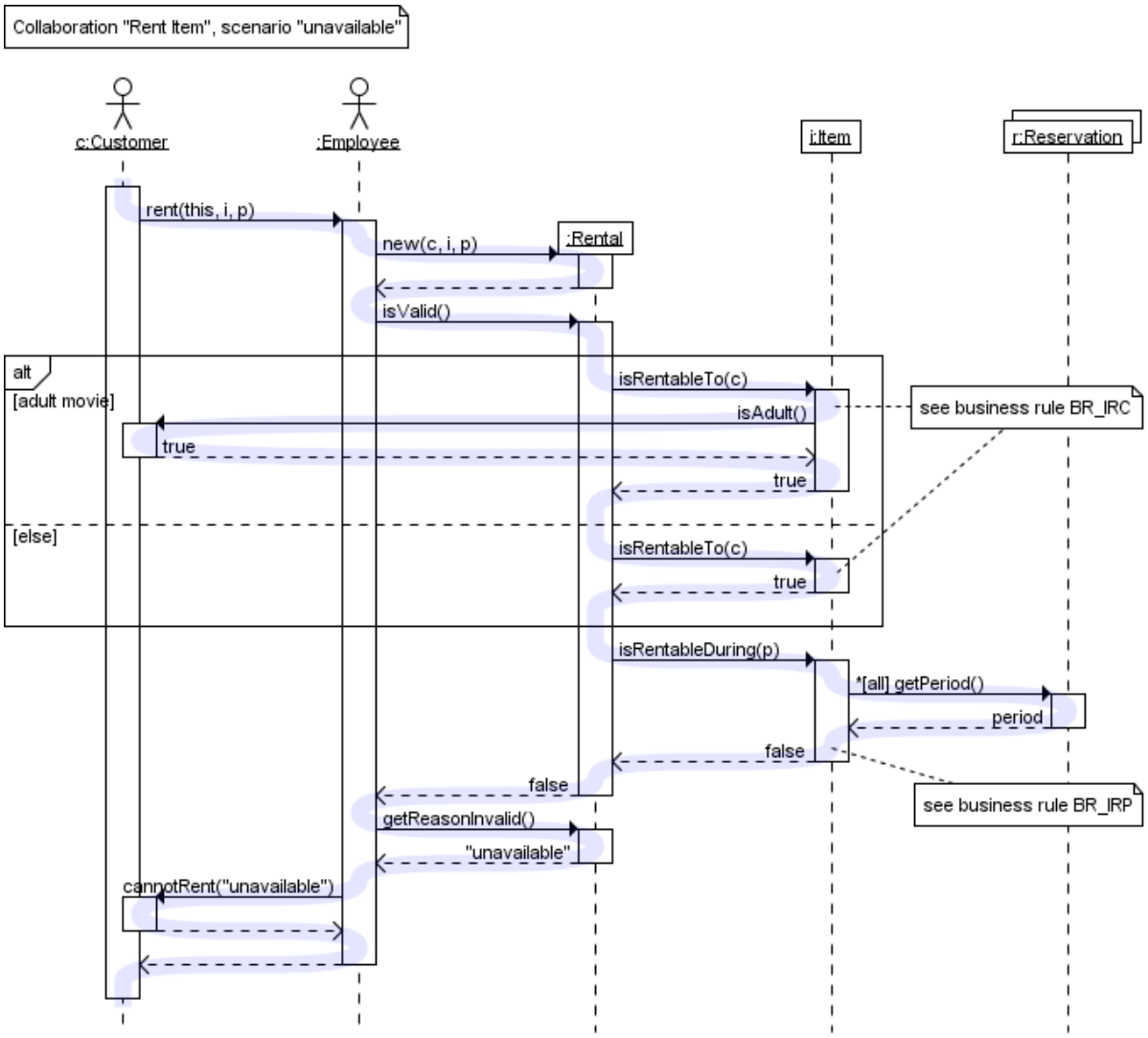

Figure 4 Example of Sequence Diagram

2.3. Applied Tool

A. Atom

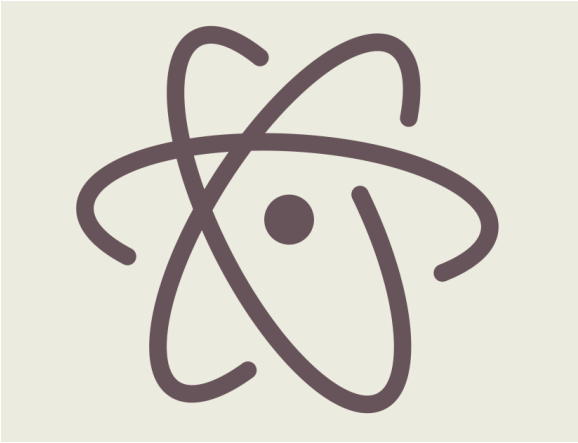

Figure 5 Atom Editor

개발 과정에서 에디터 프로그램으로 'Atom'을 이용하였다. Atom은 GitHub에서 만든 Electron 프레임워크 기반의 텍스트 에디터로, 다양한 프로그래밍 언어의 편집기로 사용할 수 있다. OS X, Windows, Linux 등 운영체제에 독립적으로 사용 가능하고, 다양한 패키지 및 테마를 설정할 수 있어 자유도가 높다.

B. Draw.io

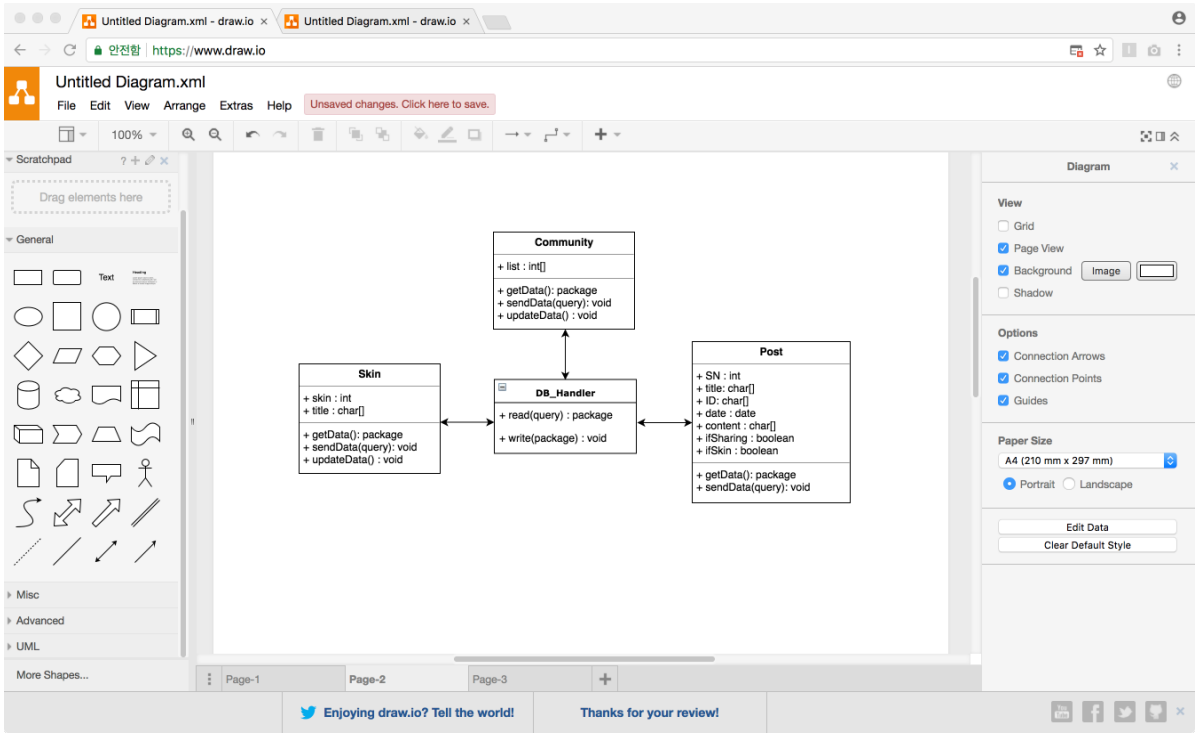

Figure 6 Draw.io

본 문서에 첨부되어 있는 sequence diagram, state diagram 등은 draw.io를 사용하여 작성했다. 웹 기반 서비스로 특별한 가입 없이 이용할 수 있고, 다양한 UML을 그릴 수 있어 편리하다. 또한 GitHub, Dropbox, OneDrive와도 연동이 되고, 공유 기능을 사용하여 협업도 가능하다.

2.4. Project Scope

ToDot(투닷)은 기존 SNS(페이스북, 인스타그램 등)나 블로그, 커뮤니티 등의 글(혹은 게시물) 공유 플랫폼의 부정적인 측면을 최소화하고 개인의 생각을 정리하여 관리할 수 있는 사적인 비밀 공간을 제공하고자 기획된 프로젝트이다. ToDot은 크게 Sign up and Login system, Post system, Community system, Send message system, Evaluation and Block system, Feature analysis system, Cover system 의 7개 시스템으로 구성되어 있다.

A. Sign up and Login system

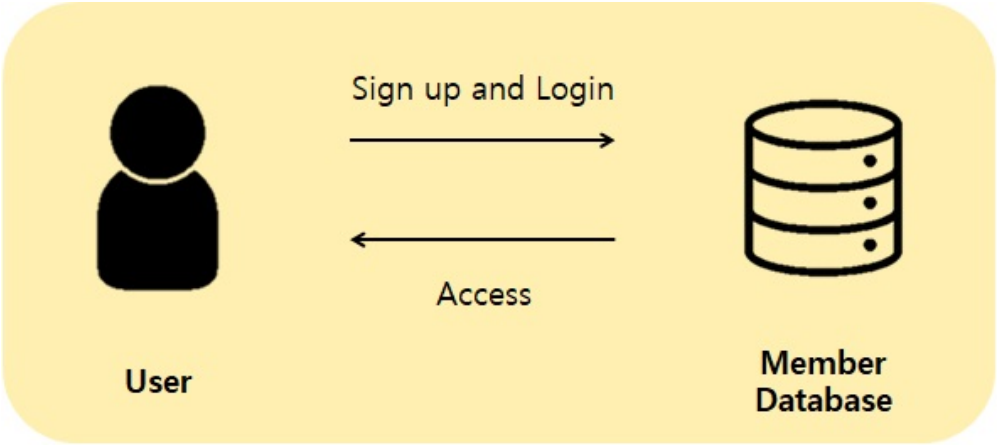

Diagram 1 Structure of Sign up and Login system

Sign up and Login system은 사용자의 회원 정보를 관리하는 Sign up system과 ToDot 시스템에 접근하게 하는 Login system으로 구성된다.

B. Post system

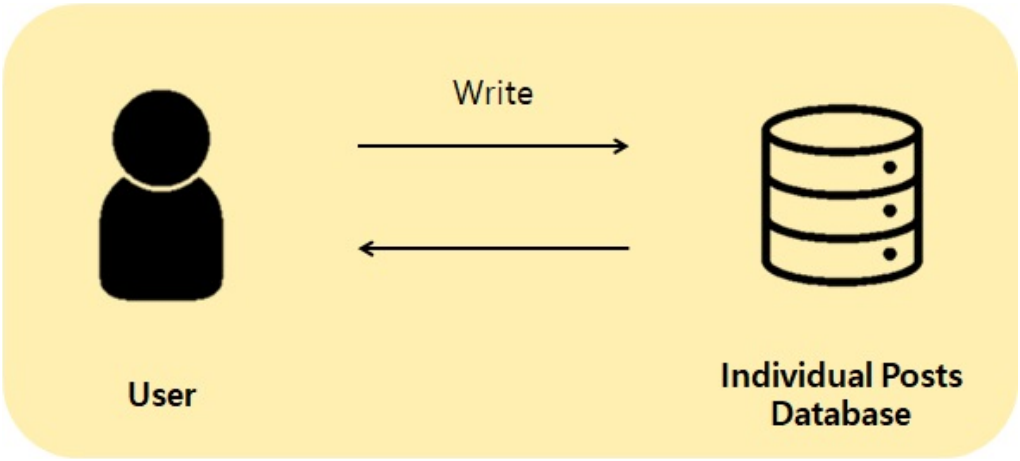

Diagram 2 Structure of Post system

Post system은 개인 사용자가 글을 작성하여 개인 데이터베이스에 저장하는 기능을 지원한다. 게시물이 자동으로 공유되어 전체 사용자가 이를 열람할 수 있도록 하는 것이 아니라 작성자 고유의 저장소에 존재하도록 하며, 공유 여부를 별도 선택할 수 있는 기능으로 지원한다.

C. Community system

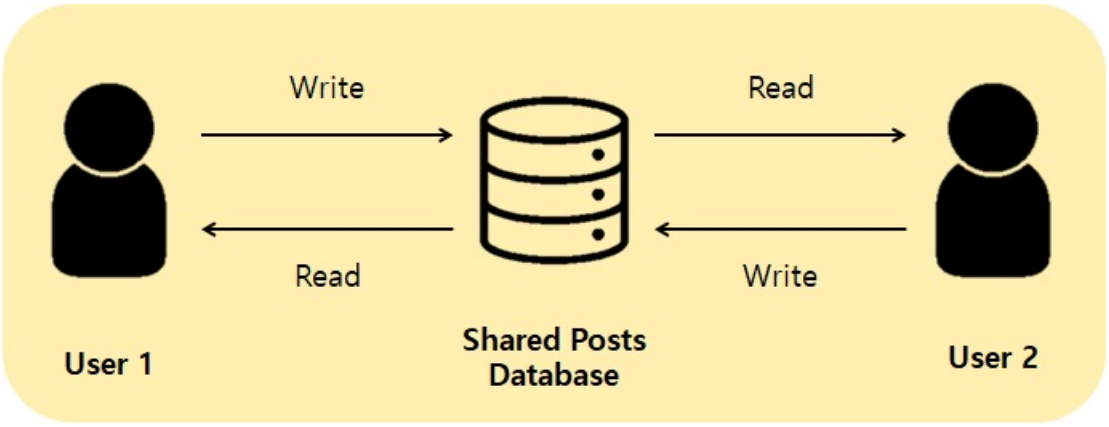

Diagram 3 Structure of Community system

Community system은 각 사용자가 자신의 게시물을 공유하여 이를 다른 사용자가 열람할 수 있도록 공개된 공간에 게시할 수 있게 하는 기능이다.

D. Send message system

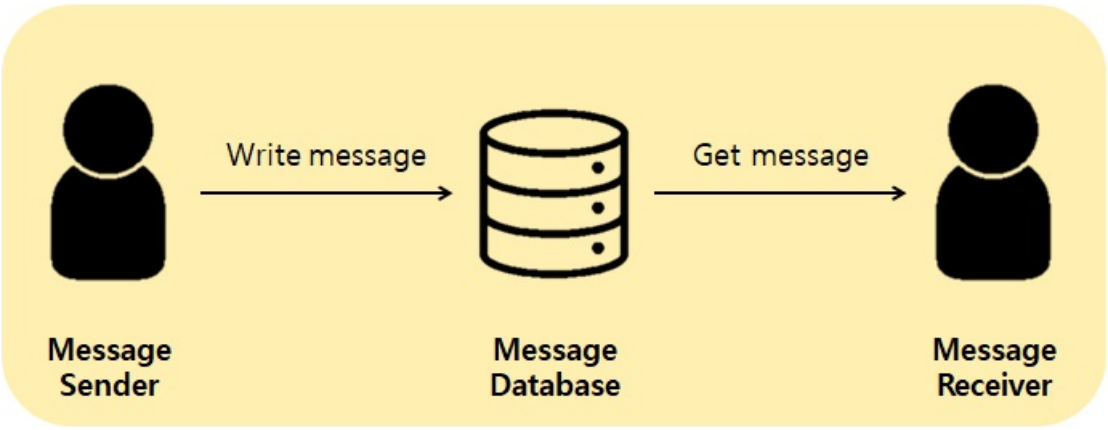

Diagram 4 Structure of Send Message system

Send message system에서는 공유된 글에 대해 글 열람자가 작성자에게 답변을 보내는 기능을 제공한다.

E. Evaluation and Block system

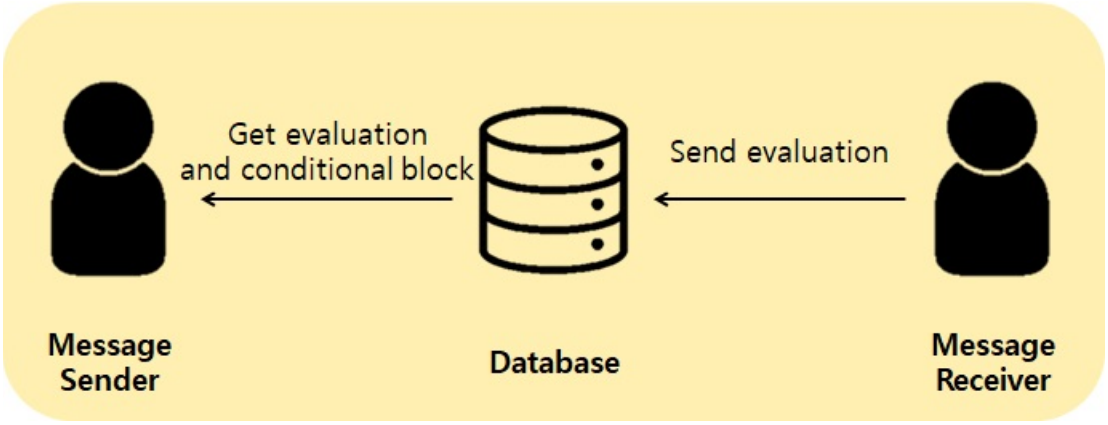

Diagram 5 Structure of Evaluation and Block system

Evaluation and Block system에서는 답변을 받은 수신자(글 작성자)가 답변의 내용을 토대로 답변자를 평가(피드백)하는 기능을 제공한다. 나쁜 평가를 받은 경우 나쁜 평가의 수가 일정 기준을 넘어가게 되면 해당 사용자(답변자)는 시스템에서 차단될 수 있다.

F. Feature Analysis system

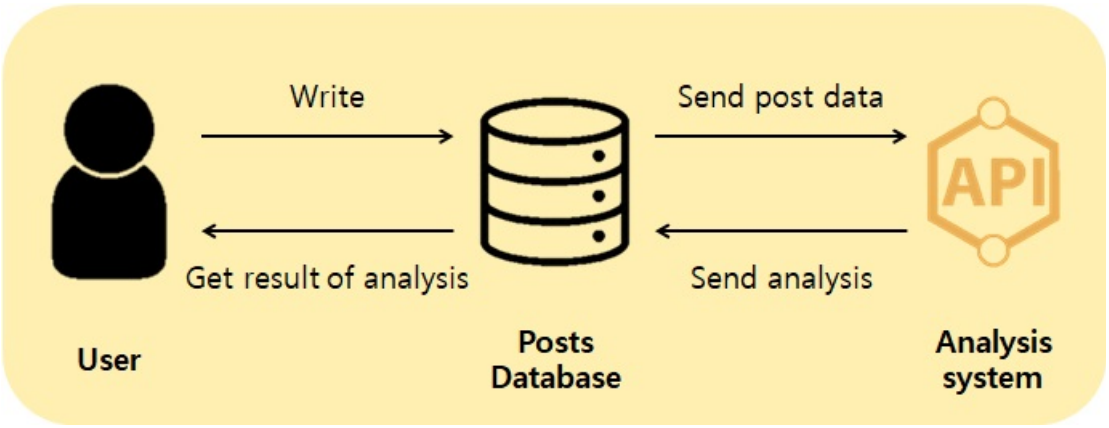

Diagram 6 Structure of Feature analysis system

Feature analysis system은 게시물의 내용을 자연어 분석 기법을 이용한 감성분석 과정을 거쳐 정량화된 수치로 환산해주는 기능을 제공한다.

G. Cover system

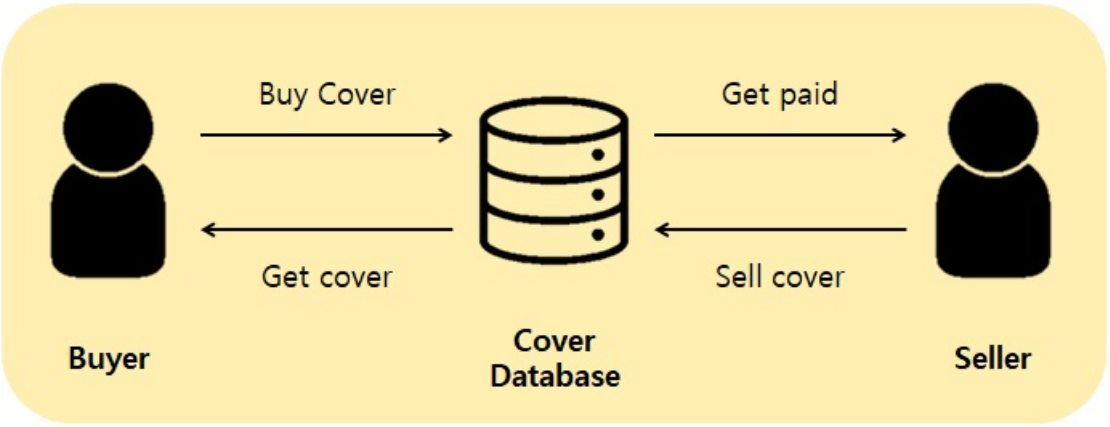

Diagram 7 Structure of Cover system

Cover system은 원하는 커버를 사고 팔 수 있도록 지원한다.

3. System Architecture

3.1. Objective

System Architecture에서는 To Dot(투닷) 시스템의 전체적인 구조를 Block diagram으로 나타낸다. 구조들의 관계와 실제 어떻게 사용되는지를 Package diagram과 Deployment diagram을 사용하여 설명한다.

3.2. System Organization

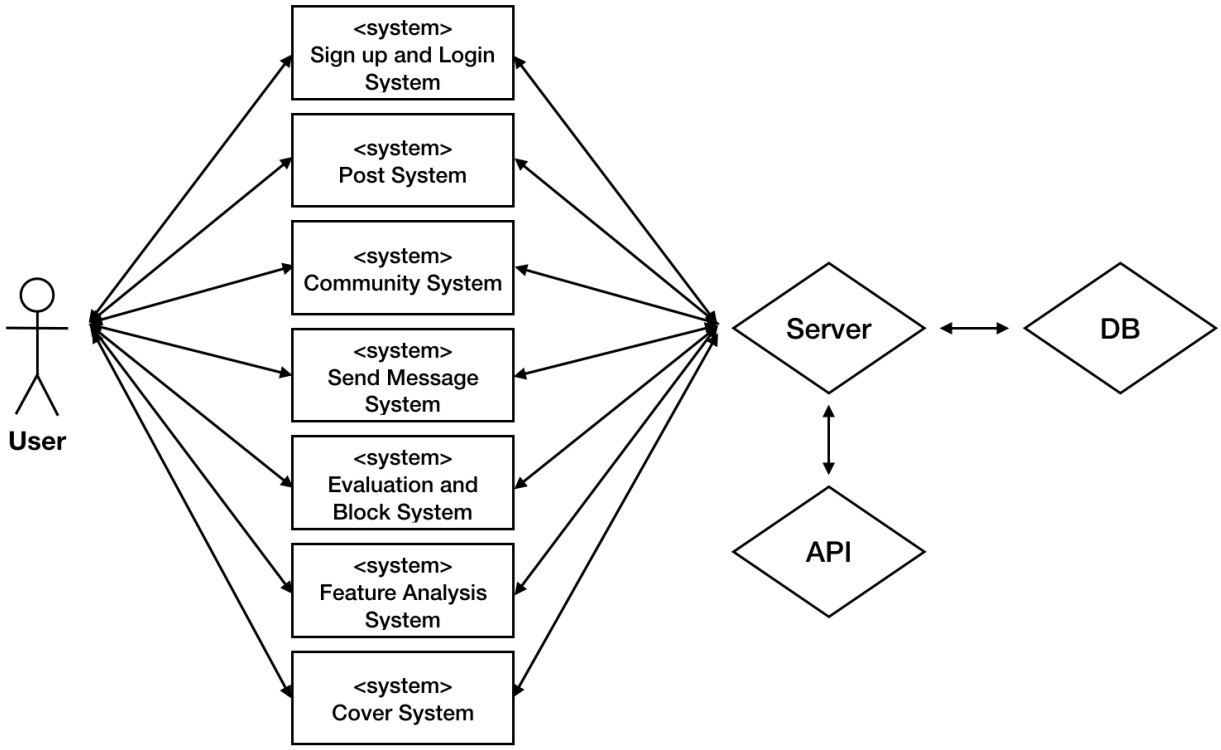

Diagram 8 System Organization Block System

To Dot(투닷) 시스템은 Client-server Model을 사용하여 구현된다. 사용자는 To Dot(투닷)의 7개의 시스템을 직/간접적으로 사용하고, 각 시스템은 사용자의 요청을 서버로 전송한다. 서버는 사용자의 요청에 맞게 내부 데이터베이스를 변경하거나, 적절한 데이터베이스를 사용자에게 전송하는 역할을 수행한다. 사용자 및 서버와 상호작용하는 7개의 시스템은 Sign up and Login System, Post System, Community System, Send Message System, Evaluation and Block System, Feature Analysis System, Cover System이다.

A. Sign up and Login System

Sign up and Login System은 회원 가입 기능을 제공하는 Sign-up sub-system, 로그인 기능을 제공하는 Log-in sub-system의 두 가지 하위 시스템으로 이루어진다.

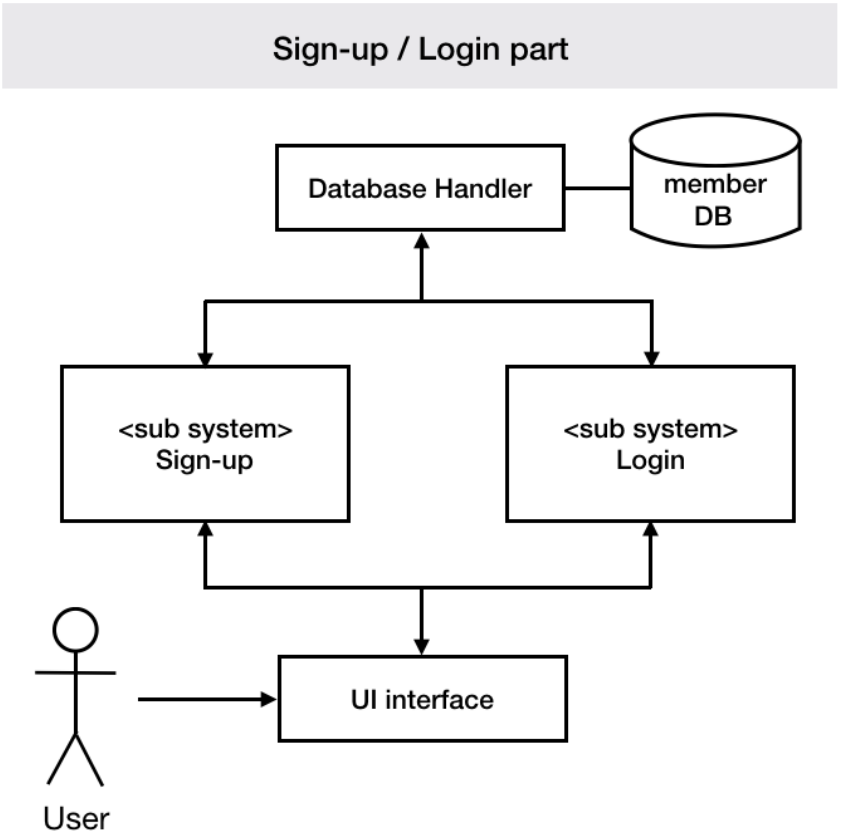

Diagram 9 User Management System Architecture

B. Post System

Post System은 사용자가 개인 공간에 글을 작성하고, 스킨 사용 여부와 커뮤니티 공개 여부를 설정할 수 있는 시스템이다.

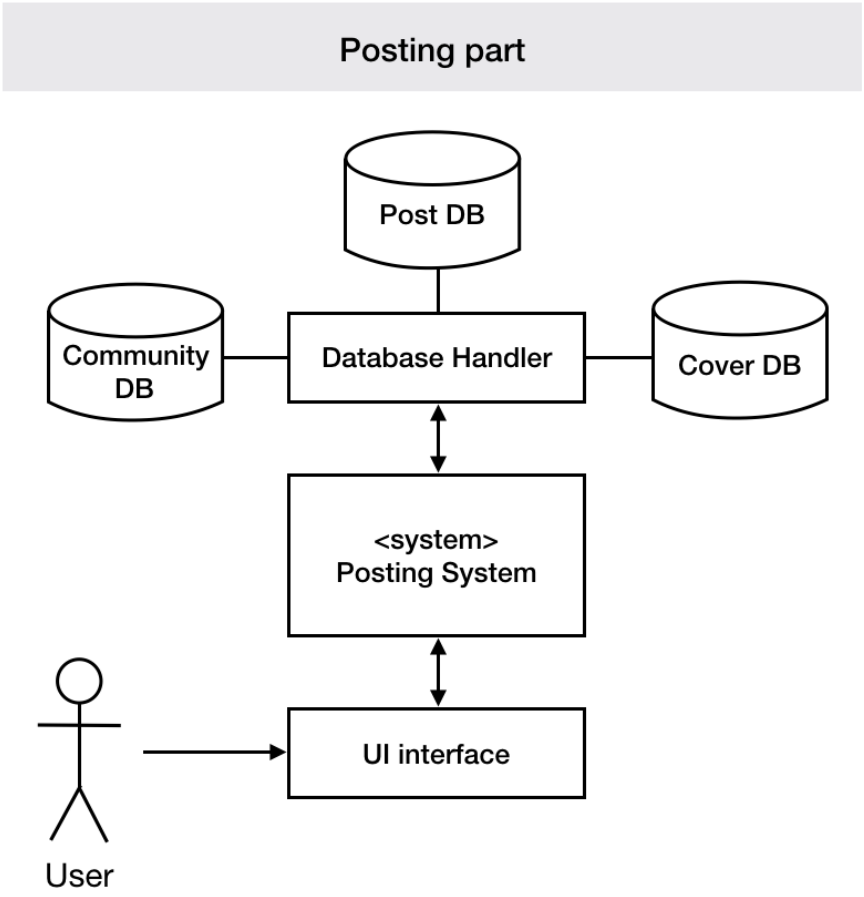

Diagram 10 Post System Architecture

C. Community System

Community System을 통해 사용자들은 공유 설정된 글을 읽을 수 있다.

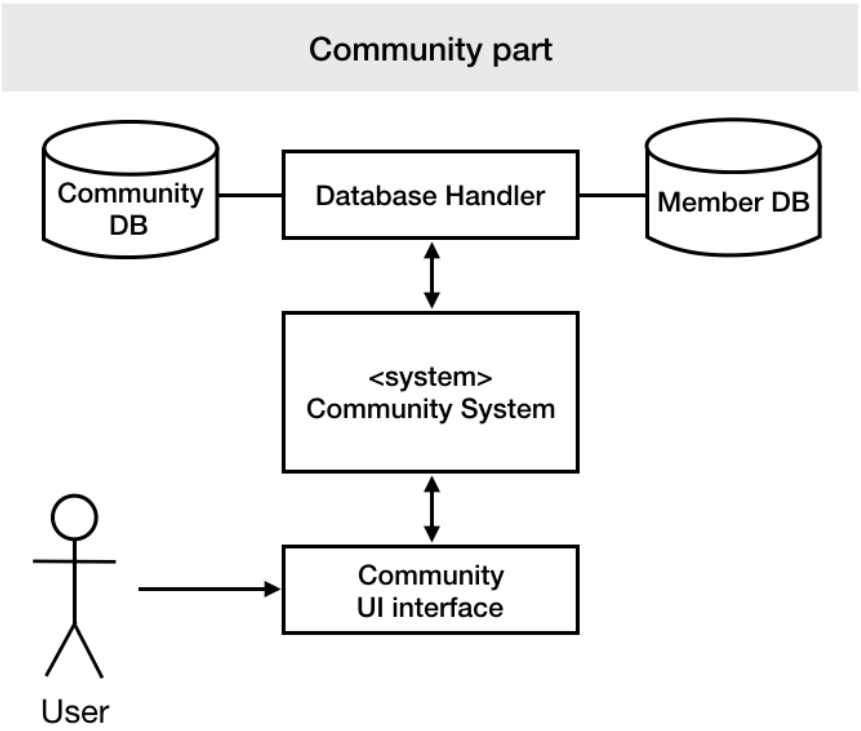

Diagram 11 Community System Architecture

D. Send Message System

커뮤니티의 Send Message System을 통해 사용자는 다른 사용자에게 쪽지를 보낼 수 있다.

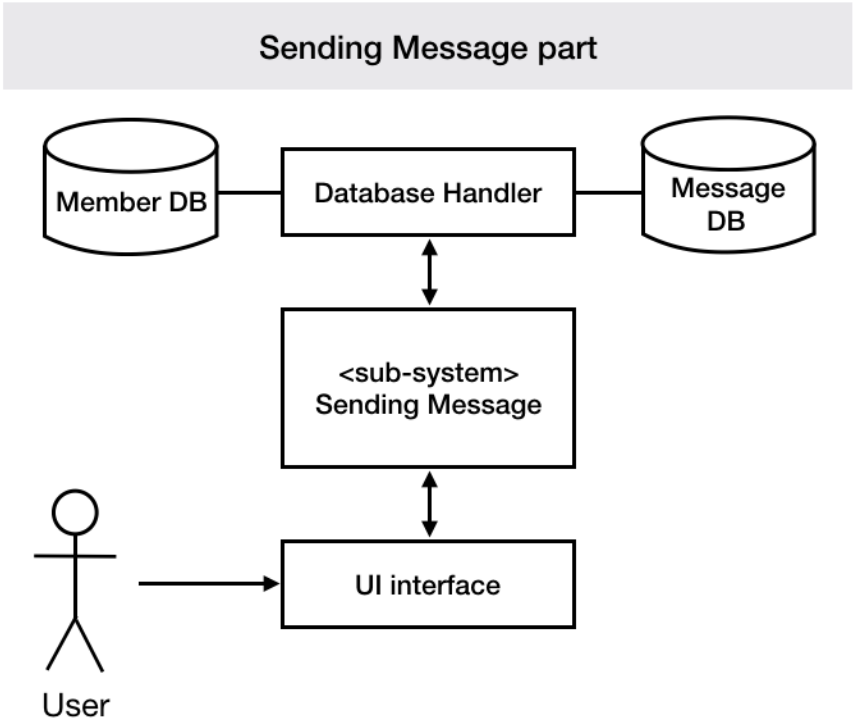

Diagram 12 Send Message System Architecture

E. Evaluation and Block System

Evaluation and Block System을 통해 사용자들은 받은 쪽지에 대해 평가하고, 시스템은 이를 토대로 악의적인 사용자를 차단할 수 있다.

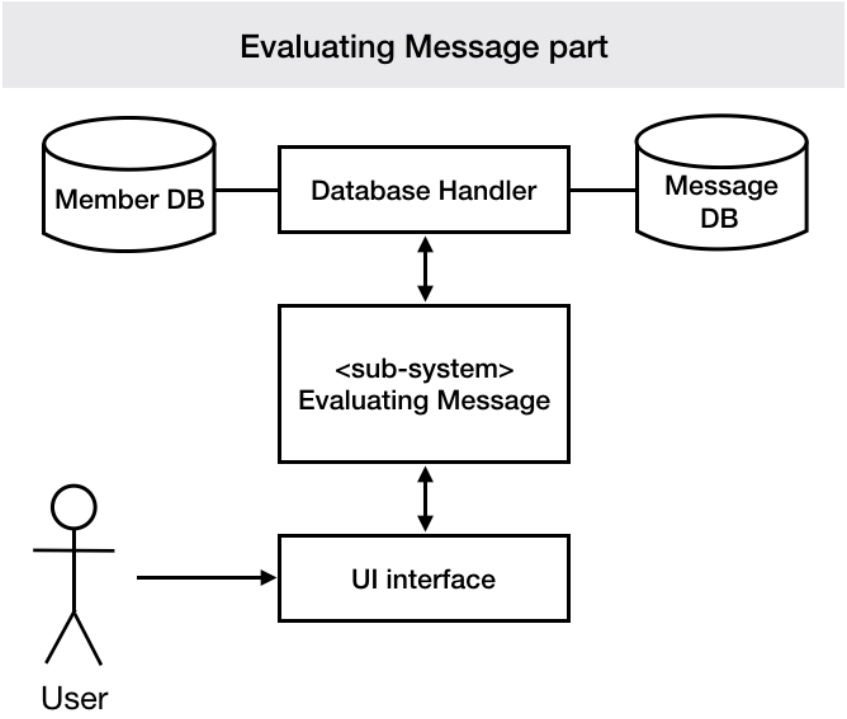

Diagram 13 Message and Evaluation System Architecture

F. Feature Analysis System

Feature Analysis System은 사용자가 작성한 글을 Google API를 통해 분석하고, 커뮤니티에 있는 글 중 비슷한 키워드의 글을 사용자에게 추천해준다.

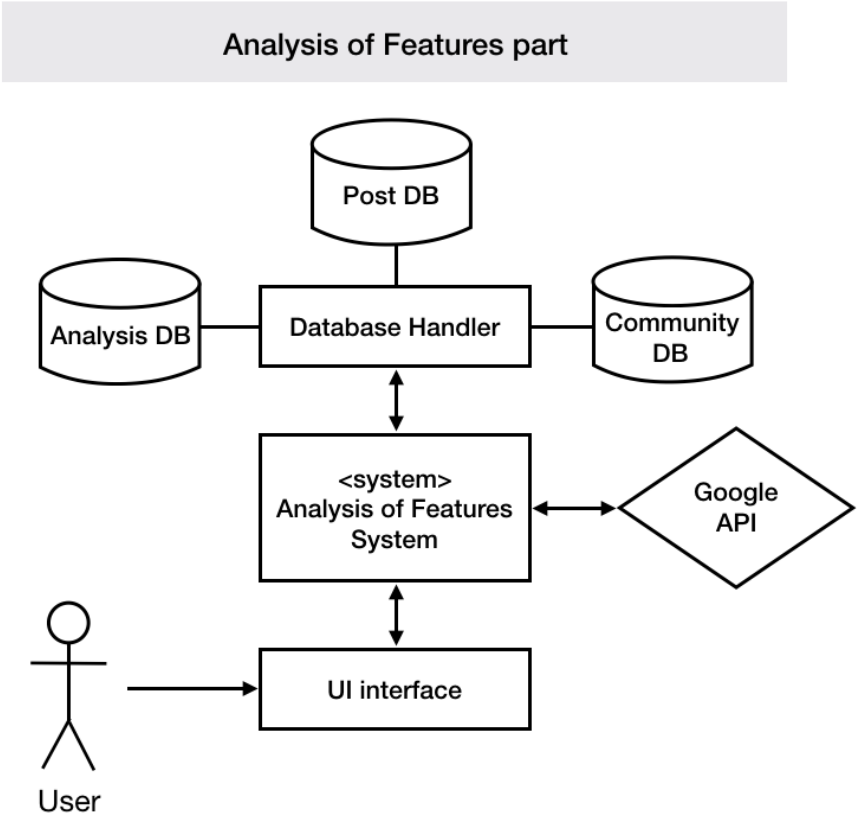

Diagram 14 Feature Analysis System Architecture

G. Cover System

Cover System의 경우, 사용자가 글에 적용할 스킨을 구매하는 기능인 Purchasing Cover sub-system, 보유한 스킨을 글에 적용하는 기능인 Adapting Cover sub-system의 두 가지 하위 시스템으로 이루어진다.

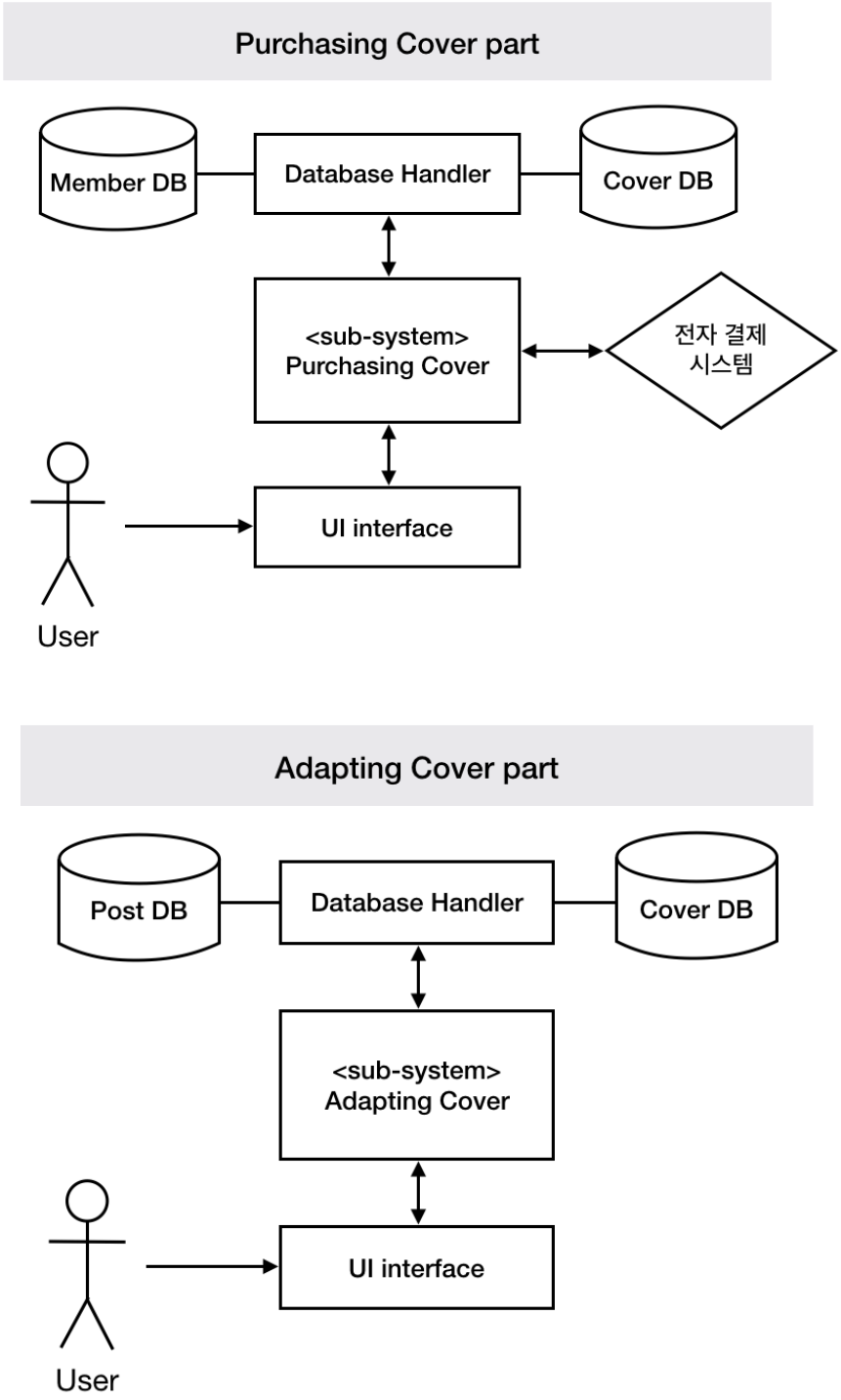

Diagram 15 Cover System Architecture

4. Sign up and Login System

4.1. Objectives

Sign up and Login System은 To Dot(투닷) 서비스에 가입한 회원의 계정을 관리한다. Member DB와 연동되어 회원 가입과 로그인 기능을 제공한다. Class diagram, Sequence diagram, State Diagram을 통해 Sign up and Login System의 구조와 사용자 및 DB와의 상호 작용을 설계한다.

4.2. Class Diagram

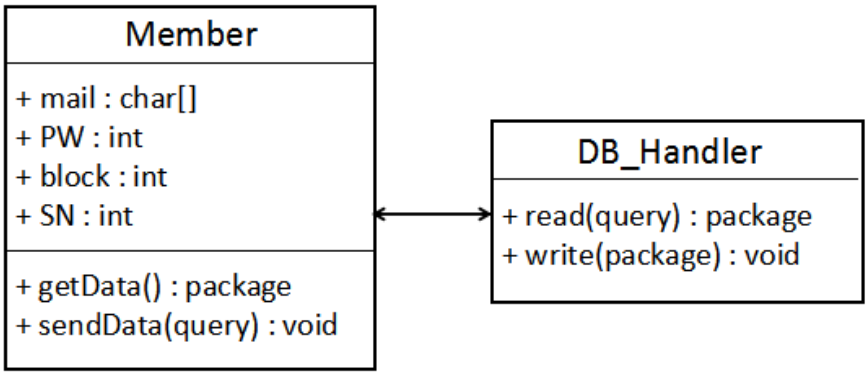

Diagram 16 Sign up and Login System Class Diagram

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+ package read(query) : 해당되는 DB에서 원하는 데이터를 읽어온다.

+ void write(package) : 해당되는 DB에 데이터를 저장한다.

B. Member

B.1. Attributes

+ SN : 계정 고유번호 값

+ block : 차단 받은 횟수

+ PW : 계정 비밀번호 정보

+ mail : 계정 이메일 주소 정보

B.2. Methods

+ package getData() : DB로부터 데이터를 받는다.

+ void sendData(query) : DB로 데이터를 보낸다.

4.3. Sequence Diagram

A. Sign up

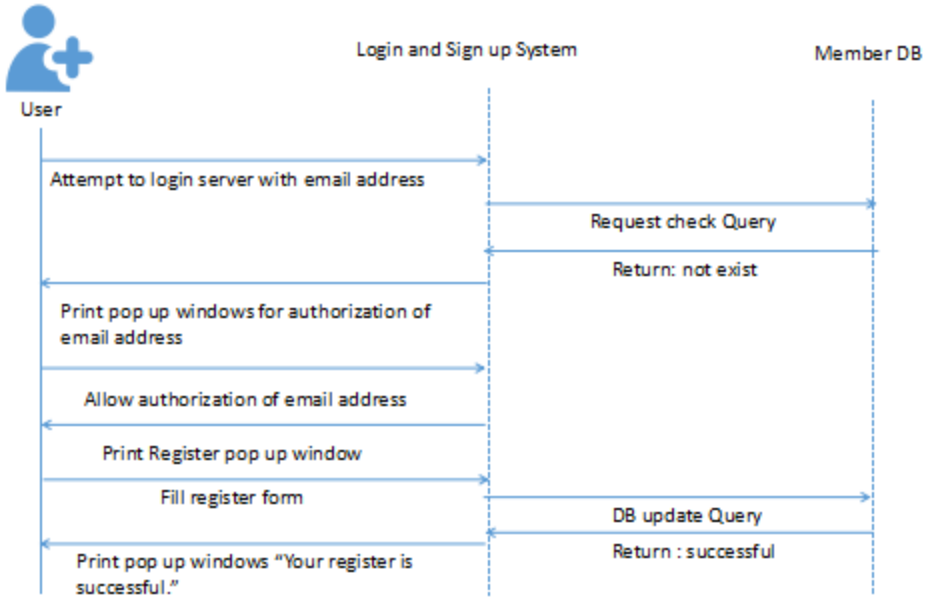

Diagram 17 Sign up Sequence Diagram

B. Login

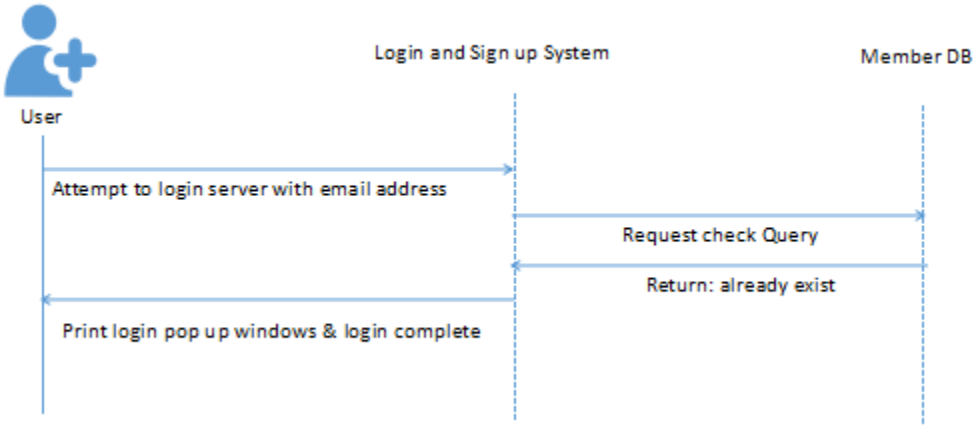

Diagram 18 Sign up Sequence Diagram

5. Post System

5.1. Objectives

Post System은 사용자가 개인 공간에 글을 작성하는 시스템이다. 글을 업로드 할 때, 스킨 사용 여부와 커뮤니티 공개 여부를 어떻게 설정하는지에 따라 스킨 및 커뮤니티 시스템과의 상호 작용이 달라진다. 이 절에서는 Class diagram, Sequence Diagram, State Diagram을 통해 Post System의 구조와 사용자 및 DB와의 상호 작용을 설계한다.

5.2. Class Diagram

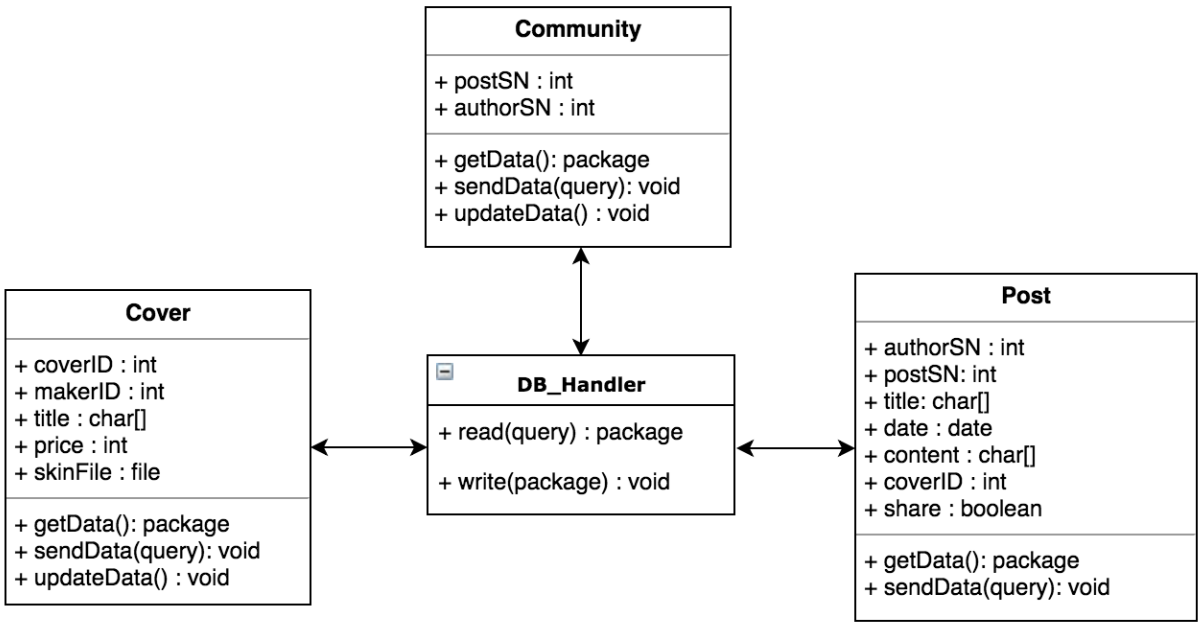

Diagram 19 Posting System Class Diagram

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

package read(query) : 해당되는 DB에서 원하는 데이터를 읽어온다.

void write(package) : 해당되는 DB에 데이터를 저장한다.

B. Post

B.1. Attributes

authorSN : 글 작성자의 고유번호

postSN : 글의 고유번호

title : 글 제목 정보

date : 글 작성 날짜 정보

content : 글 내용 정보

coverID : 사용한 커버 테마 정보

share : 커뮤니티 공유 여부

B.2. Methods

package getData() : DB로부터 데이터를 받는다.

void sendData(query) : DB로 데이터를 보낸다.

C. Community

C.1. Attributes

authorSN : 글 작성자의 고유번호

postSN : 글의 고유번호

C.2. Methods

package getData() : DB로부터 데이터를 받는다.

void sendData(query) : DB로 데이터를 보낸다.

void updateData() : DB를 업데이트한다.

D. Cover

D.1. Attributes

coverID : 커버의 고유번호

makerID : 커버 작성자 고유번호

title : 커버의 이름

price : 커버의 가격 정보

skinFile : 이미지 정보

D.2. Methods

package getData() : DB로부터 데이터를 받는다.

void sendData(query) : DB로 데이터를 보낸다.

void updateData() : DB를 업데이트한다.

5.3. Sequence Diagram

A. Posting

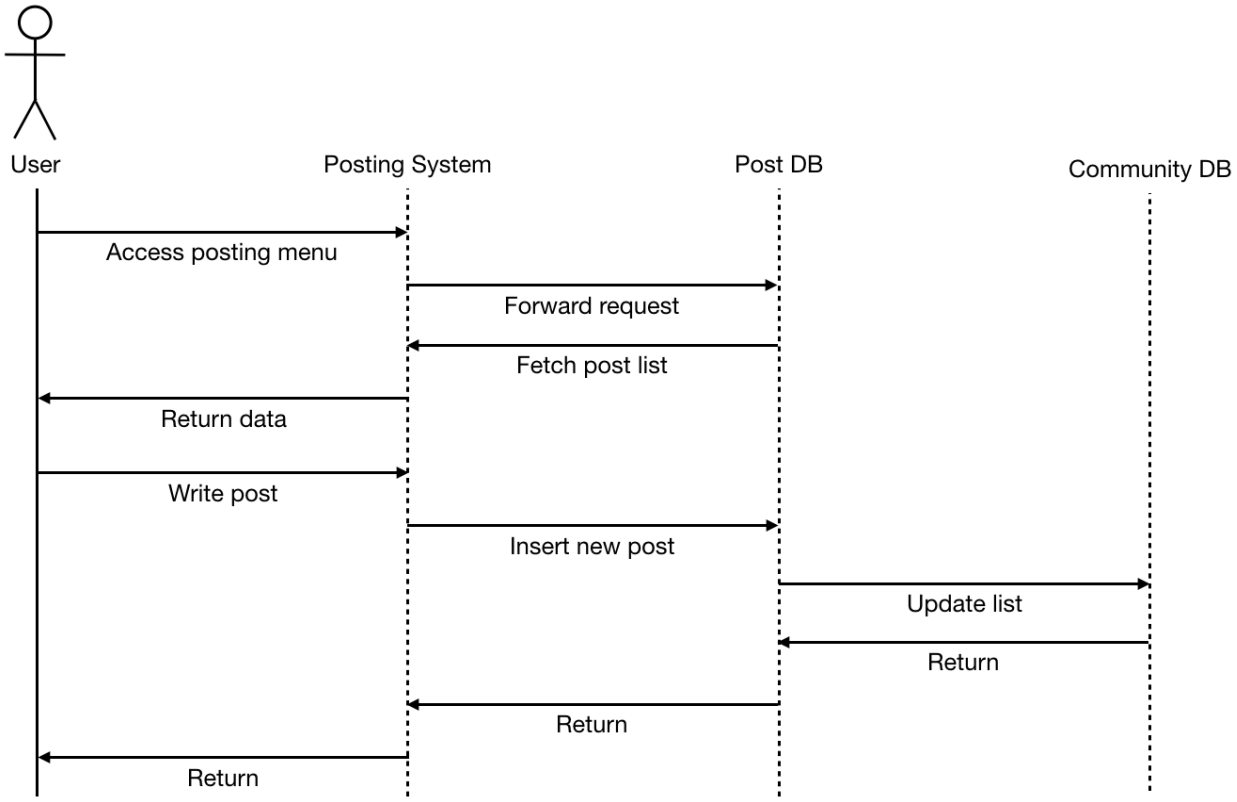

Diagram 20 Posting System Sequence Diagram

B. Adapting Cover

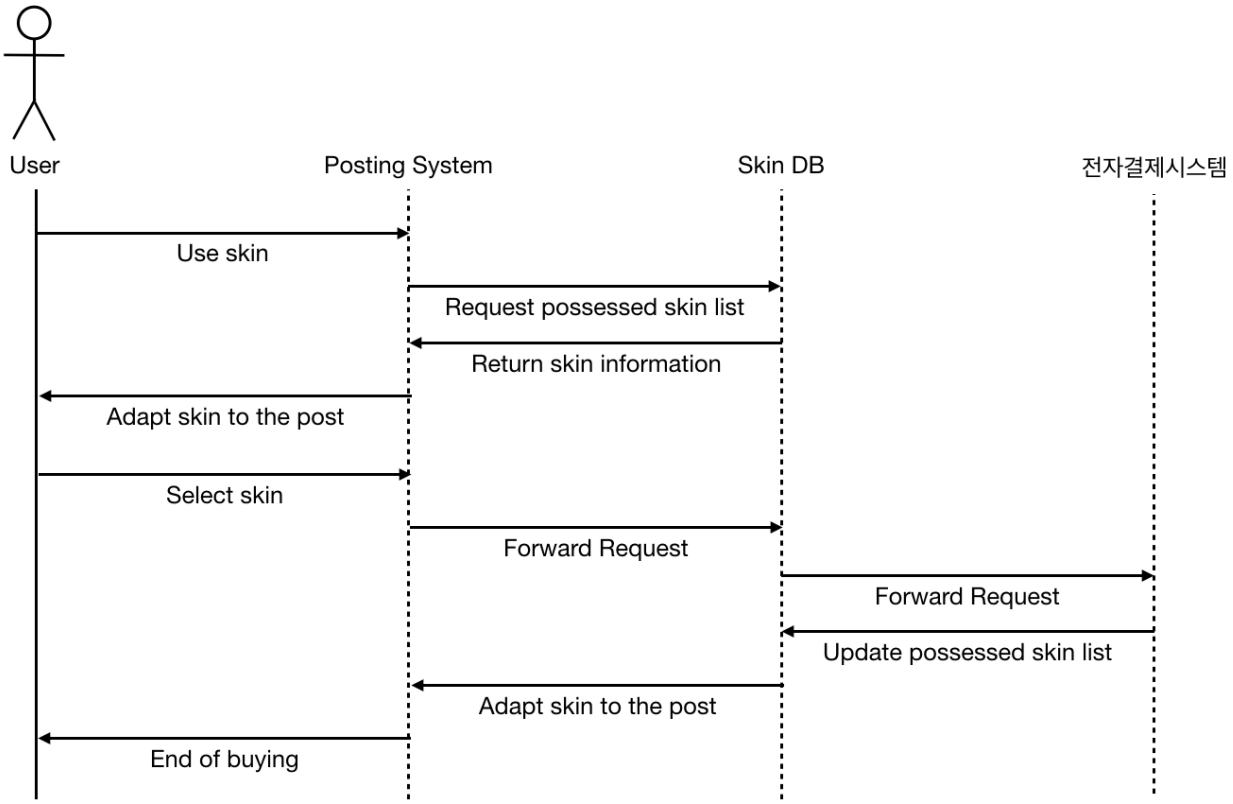

Diagram 21 Posting System-Adapting Cover- Sequence Diagram

5.4. State Diagram

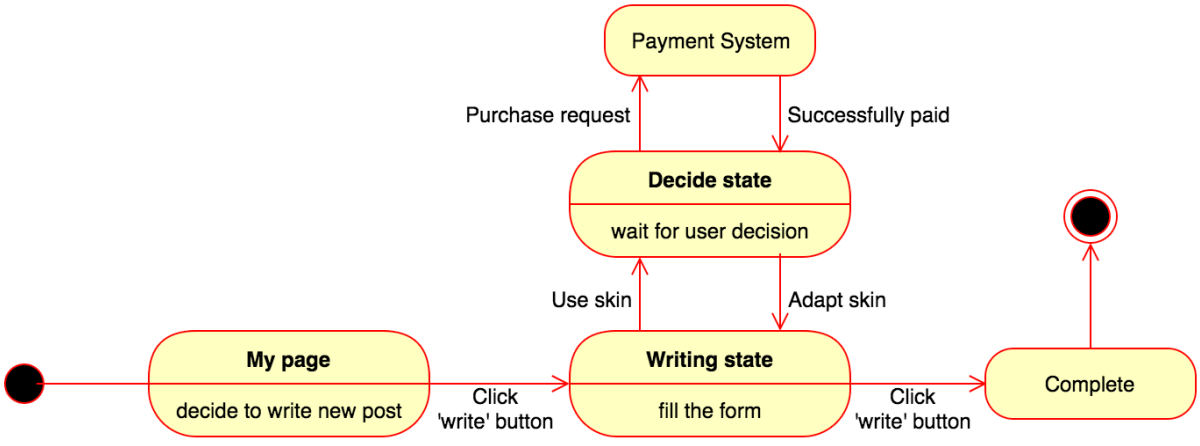

Diagram 22 Posting System State Diagram

6. Community System

6.1. Objectives

Community System은 사용자들 간 커뮤니티를 통해 공감대를 형성하고 고민을 나눌 수 있는 기능이다. 본인의 글을 작성할 때 공유 여부를 선택하면, 선택된 글들에 한해서 Community Board로 업데이트가 된다. 커뮤니티의 모든 글은 익명으로 보여지고, 공유된 날짜와 시간 순서에 따라 최신의 글이 상위에 노출된다. 커뮤니티는 댓글 기능을 제공하지 않고, 쪽지(message)를 주고받을 수 있다. 아래는 Class diagram, Sequence diagram과 State Diagram을 통해 Community System의 구조를 표현하고 설명한다.

6.2. Class-Diagram

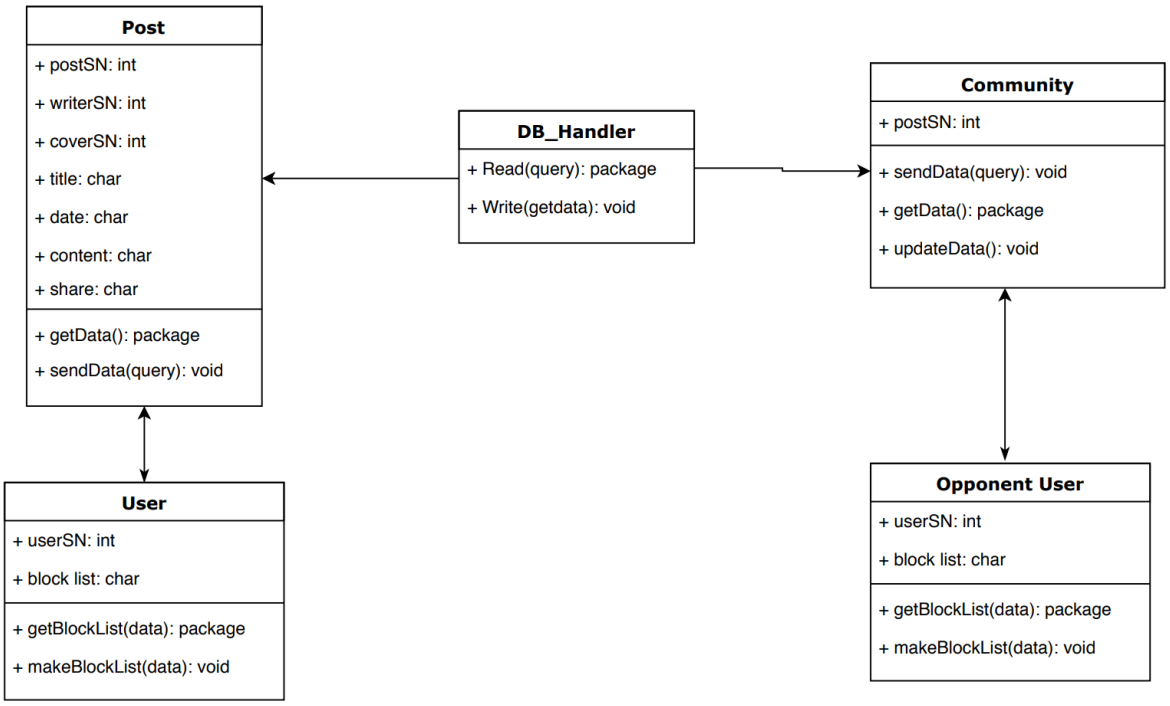

Diagram 23 Community System Class Diagram

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+ package read(query) : 해당되는 DB에서 원하는 데이터를 읽어온다.
+ void write(package) : 해당되는 DB에 데이터를 저장한다.

B. Post

B.1. Attributes

+ postSN : 글의 고유번호
+ writerSN: 작성자 고유번호
+ title : 글 제목 정보
+ coverSN : 표지 고유번호
+ date : 글 작성 날짜
+ content : 글 내용
+ share : 커뮤니티 공유 여부

B.2. Methods

+ package getData() : DB로부터 데이터를 받는다.
+ void sendData(query) : DB로 데이터를 보낸다.

C. Community

C.1. Attributes

+ postSN : 글의 고유번호

C.2. Methods

+ package getData() : DB로부터 데이터를 받는다.
+ void sendData(query) : DB로 데이터를 보낸다.
+ void updateData() : DB를 업데이트한다.

D. User

+ userSN : 유저 고유번호
+ block list: 차단 여부
+ package getBlockList(data): 차단 리스트를 받는다
+ void makeBlockList(data): 차단 리스트를 보낸다.

E. Opponent User

+ userSN : 유저 고유번호
+ block list: 차단 여부
+ package getBlockList(data): 차단 리스트를 받는다
+ void makeBlockList(data): 차단 리스트를 보낸다.

6.3. Sequence-Diagram

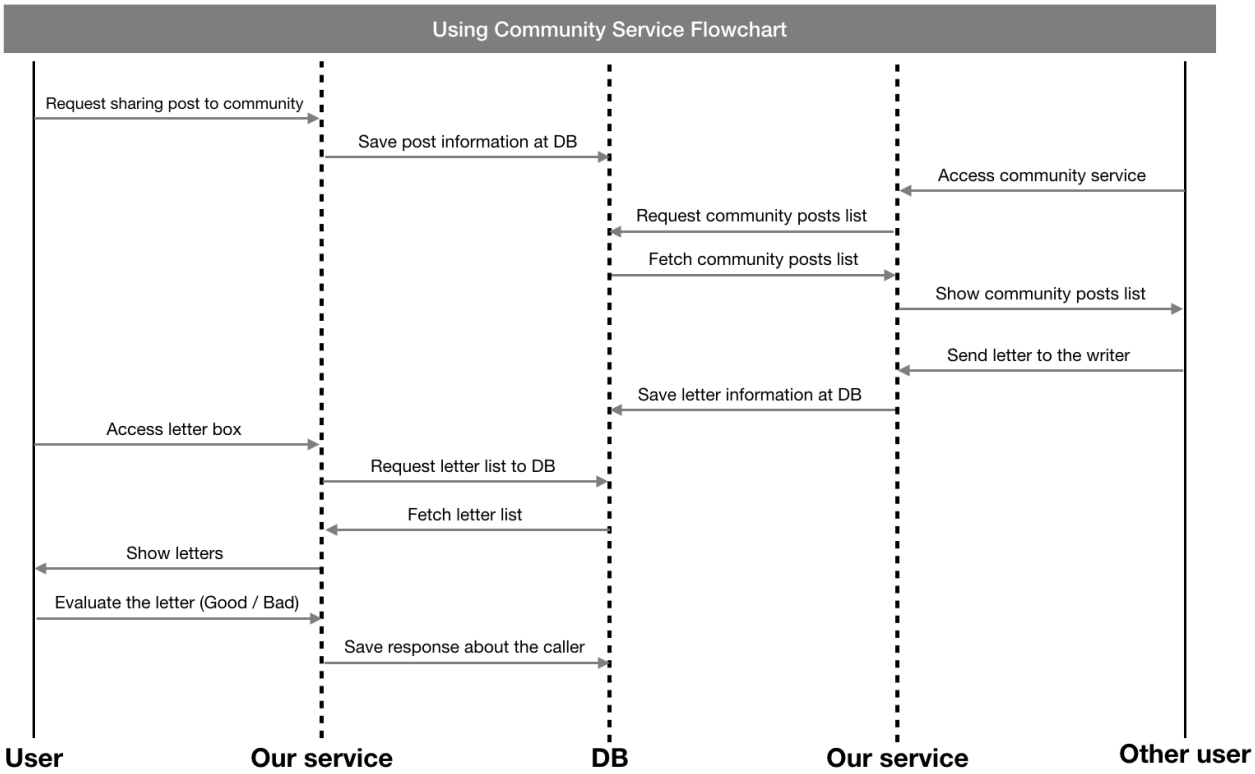

Diagram 24 Community System Sequence Diagram

6.4. State-Diagram

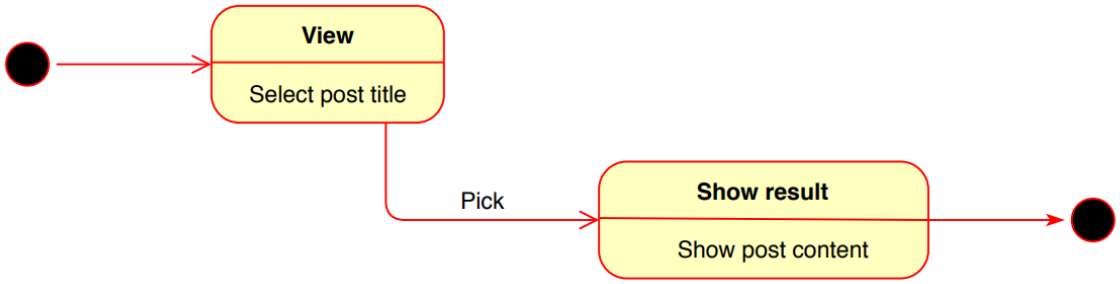

Diagram 25 Community System State Diagram

7. Send Message System

7.1. Objectives

다른 사용자가 공유한 글(Post)에 답변을 보내는 작업을 말한다. 이 과정에서 ToDot 커뮤니티(Community)의 전체 공유된 글에 접근할 수 있어야 하며 원하는 글을 불러와 내용을 확인한 후 해당 글의 게시자에게 쪽지를 보낼 수 있어야 한다.

7.2. Class Diagram

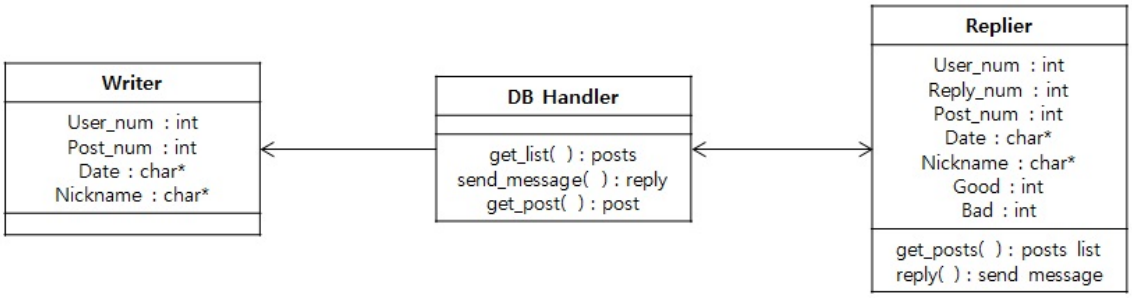

Diagram 26 Class Diagram of Send Message system

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+post get_list( ) : community DB에 저장된 모든 post list를 읽어 온다.

+void send_message( ) : 작성한 쪽지를 글 작성자에게 보낸다.

+post get_post( ) : post list에서 원하는 글을 불러온다.

B. Replier

B.1. Attributes

+User_num : 사용자 고유번호

+Reply_num : 답변 글의 고유번호

+Post_num : 답변할 글의 고유번호

+Date : 각 게시글의 작성 일자 및 시간

+Good : 사용자가 받은 좋은 평가 수

+Bad : 사용자가 받은 나쁜 평가 수

B.2. Methods

+post get_posts( ) : 공유 글 목록을 읽어 온다.

+void reply( ) : 작성한 답변을 보낸다.

C. Writer

C.1. Attributes

+User_num : 사용자 고유번호

+Post_num : 게시한 글의 고유번호

+Date : 각 게시글의 작성 일자 및 시간

C.2. Methods

해당 사항 없음.

7.3. Sequence Diagram

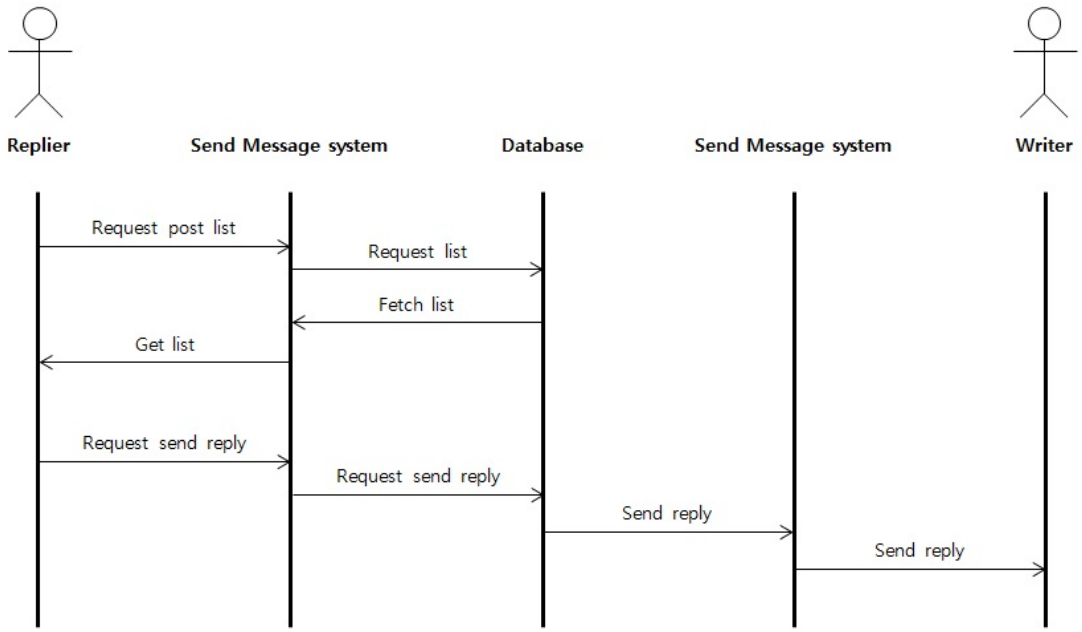

Diagram 27 Sequence Diagram of Send Message system

7.4. State Diagram

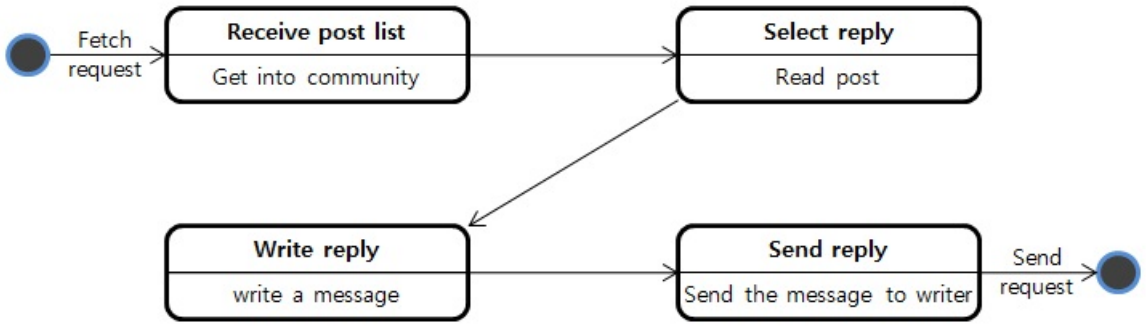

Diagram 28 State Diagram of Send Message system 1

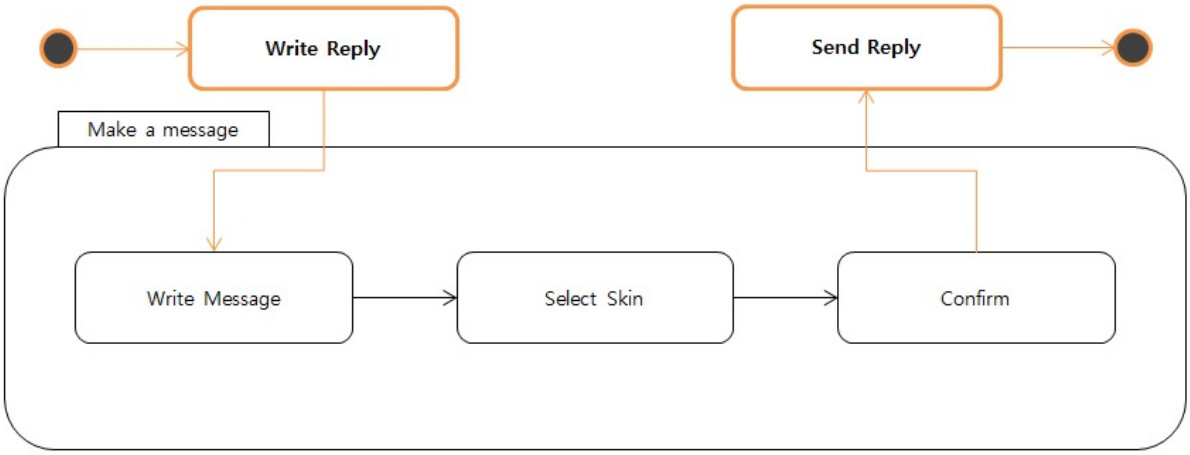

Diagram 29 State Diagram of Send Message system 2

8. Evaluation and Block System

8.1. Objectives

본 프로젝트에서는 게시한 글에 댓글을 달지 않고 메일이나 쪽지와 같은 형식의 답장을 보내는 것으로 글에 대한 의견을 표현하려고 한다. 이 때, 악의적인 사용자에 의한 시스템 악용을 막기 위해 사용자를 필터링하는 시스템이 필요한 것으로 판단되었고 이 과정이 사용자 간의 상호작용을 통해 자연스럽게 이뤄질 수 있는 구조를 구축하고자 하였다. 그에 따라 받은 답변에 대한 평가와 평가의 누적을 통한 차단 시스템을 구현하게 되었다.

8.2. Class Diagram

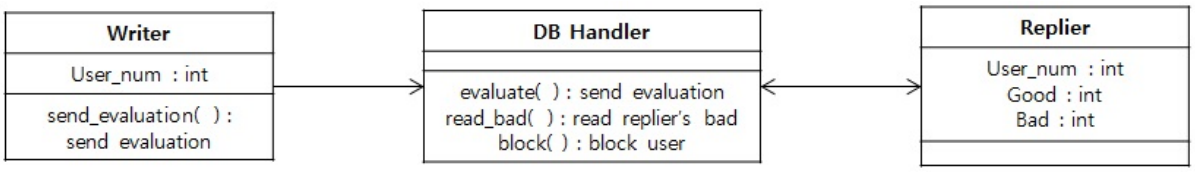

Diagram 30 Class Diagram of Evaluation and Block system

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+evaluate( ) : 글 작성자의 평가를 답변자에게 보낸다.

+read_bad( ) : replier의 bad number를 읽어 온다.

+block( ) : 읽어 온 bad number를 기준으로 사용자를 차단한다.

B. Writer

B.1. Attributes

+User_num : 사용자 고유번호

B.2. Methods

+send_evaluation( ) : 평가 사항을 답변자에게 보낸다.

C. Replier

C.1. Attributes

+User_num : 사용자 고유번호

+Good : 사용자가 받은 좋은 평가 수

+Bad : 사용자가 받은 나쁜 평가 수

C.2. Methods

해당 사항 없음.

8.3. Sequence Diagram

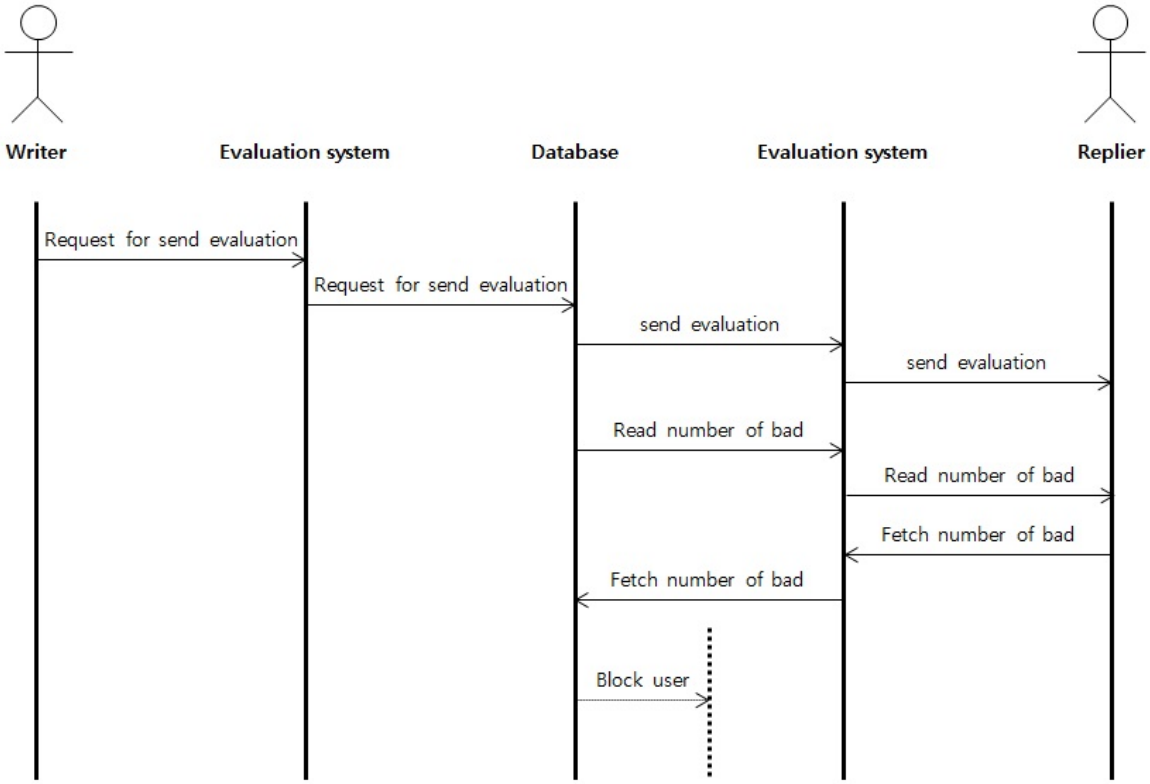

Diagram 31 Sequence Diagram of Evaluation and Block system

8.4. State Diagram

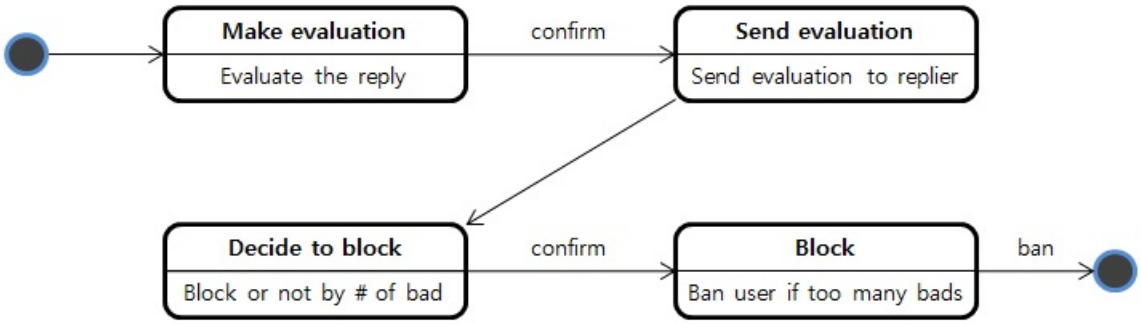

Diagram 32 State Diagram of Evaluation and Block system

9. Feature Analysis System

9.1. Objectives

Feature Analysis System에서는 자연어 처리 및 감정 분석 API를 사용하여 게시한 글에 대한 키워드 및 감정 분석 서비스를 제공한다.

9.2. Class Diagram

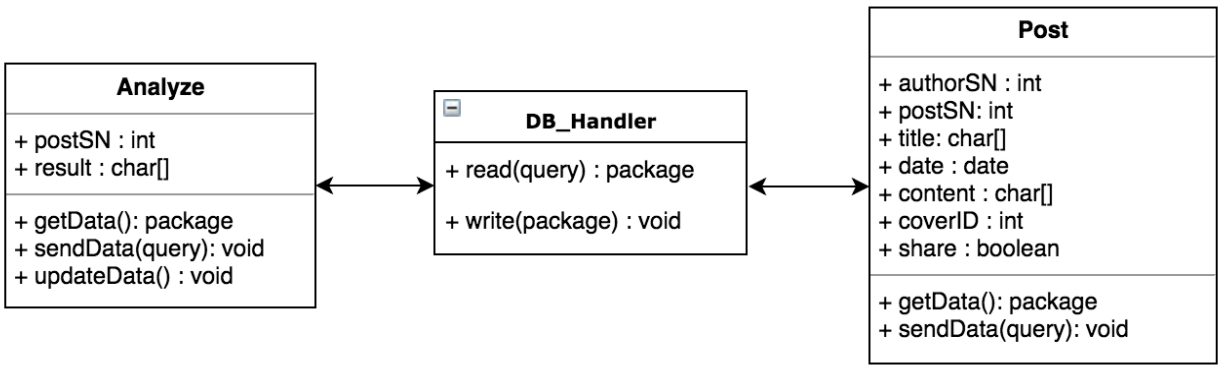

Diagram 33 Feature Analysis System Class Diagram

10. Cover System

10.1. Objectives

Cover System을 통해 보유한 커버를 확인하고, 새로운 커버를 구매할 수 있다.

10.2. Class Diagram

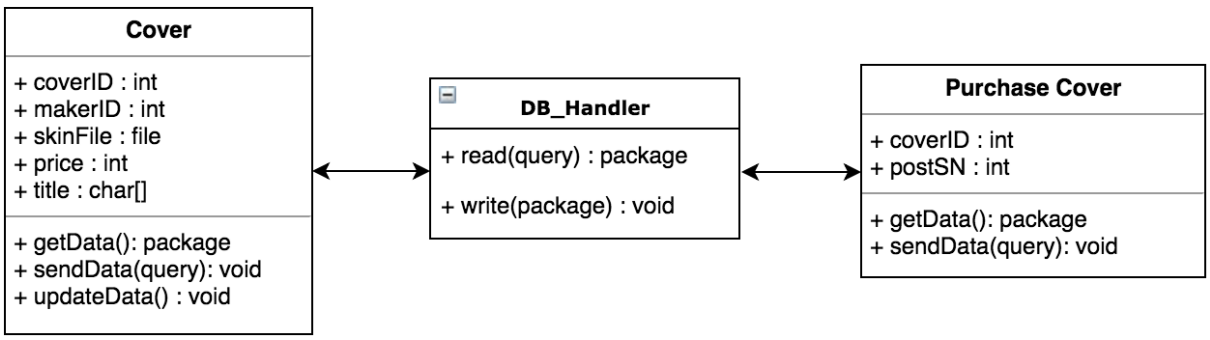

Diagram 34 Cover System Class Diagram

11. Protocol Design

11.1. Objectives

Protocol Design에서는 서브시스템들이 상호작용하는 프로토콜에 대해 서술한다. 프로토콜의 기본 형식은 JSON을 기본으로 하며, 통신하는 메시지(Message)의 형식과 용도, 의미를 설명한다.

Figure 7 JSON symbol

11.2. JSON

JSON(JavaScript Object Notation)은 보다 가볍고 편리하게 만들어진 텍스트 기반의 데이터 교환 방식이다. JavaScript에 기반하여 만들어진 데이터 표현형식이지만 프로그래밍언어나 플랫폼에 독립적인 특성을 갖고 있어, 다양한 언어에서 JSON을 활용할 수 있다. Attribute-Value의 쌍으로 표현되며, 거의 대부분의 자료형을 사용할 수 있다. JSON의 공식 인터넷 미디어 타입은 application/json이며, 확장자는 .json이다.

11.3. Protocol Description

A. Overview

HTTP 통신에서 client와 server 사이에서 전송되는 메시지의 형태를 용도 별로 정의한다. Client에서의 요청(request) 메시지와 server에서의 응답 (response) 메시지로 구분한다. 해당 절의 표는 캡션을 생략한다.

B. Login Protocol

a. Request

Attribute	Value
Email ID	사용자의 이메일
Password	사용자의 비밀번호

b. Response

Attribute	Value
login_success	로그인 성공 여부

C. Registration Protocol

a. Request

Attribute	Value
Email ID	사용자의 이메일
Password	사용자의 비밀번호

b. Response

Attribute	Value
register_success	가입 성공 여부

D. Post/Edit Protocol

a. Request

Attribute	Value
title	글 제목
date	글 작성날짜
emotion	글 감정
content	글 내용
cover	글 표지
share	글 공유 상태

b. Response

Attribute	Value
post_success	글 작성/수정 성공 여부

E. Post Search Protocol

a. Request

Attribute	Value
search_word	검색할 단어
search_date	검색할 날짜

b. Response

Attribute	Value	
search_result	검색된 글의 목록	
	Attribute	**Value**
	title	글 제목
	date	글 작성날짜
	emotion	글 감정
	content	글 내용
	cover	글 표지
	shareability	글 공유 상태

F. Community Send Protocol

a. Request

Attribute	Value
title	글 제목
date	글 작성날짜
emotion	글 감정
content	글 내용
cover	글 표지
shareability	글 공유 상태

b. Response

Attribute	Value
share_success	글 공유 성공 여부

G. Community View Protocol

a. Request

Attribute	Value
share_SN	공유된 게시글 고유번호

b. Response

Attribute	Value
title	글 제목
date	글 작성날짜
emotion	글 감정
content	글 내용
cover	글 표지
shareability	글 공유 상태
send	쪽지 전송

H. Send message Protocol

a. Request

Attribute	Value
share_SN	공유된 게시글 고유번호
sender_username	보내는 유저 이름
recv_username	받는 유저 이름
date	보내는 날짜
content	보내는 내용

b. Response

Attribute	Value
message_success	메세지 전송 성공 여부

I. Receive message Protocol

a. Request

Attribute	Value
share_SN	공유된 게시글 고유번호

b. Response

Attribute	Value
message_success	메세지 전송 성공 여부
sender_username	보내는 유저 이름
recv_username	받는 유저 이름
date	보내는 날짜
content	보내는 내용

J. Block User Protocol

a. Request

Attribute	Value
share_SN	공유된 게시글 고유번호
date	보내는 날짜
sender_username	보내는 유저 이름
recv_username	받는 유저 이름

b. Response

Attribute	Value
block_success	차단 성공 여부

K. Cover Register Protocol

a. Request

Attribute	Value
user_SN	유저 고유번호
cover_SN	표지 고유번호
image	표지 이미지
price	표지 가격

b. Response

Attribute	Value
cover_reg_success	표지 등록 성공 여부

L. Cover Buying Protocol

a. Request

Attribute	Value
cover_SN	표지 고유번호

b. Response

Attribute	Value
cover_buy_success	표지 구입 성공 여부

M. Post Analysis Protocol

a. Request

Attribute	Value
user_SN	유저 고유번호

b. Response

Attribute	Value	
keyword	키워드	
post_count	포스트 갯수	
emotion	감정 빈도	
recommend_post	추천된 포스트 목록	
	Attribute	**Value**
	title	글 제목
	content	글 내용
	emotion	글 감정

12. Database Design

12.1. Objectives

Database Design은 요구사항 명세서의 데이터베이스 요구사항을 토대로 작성하였다. ER Diagram을 작성하고, 이를 통해 schema를 작성하고 sql DDl을 작성했다.

12.2. ER Diagram

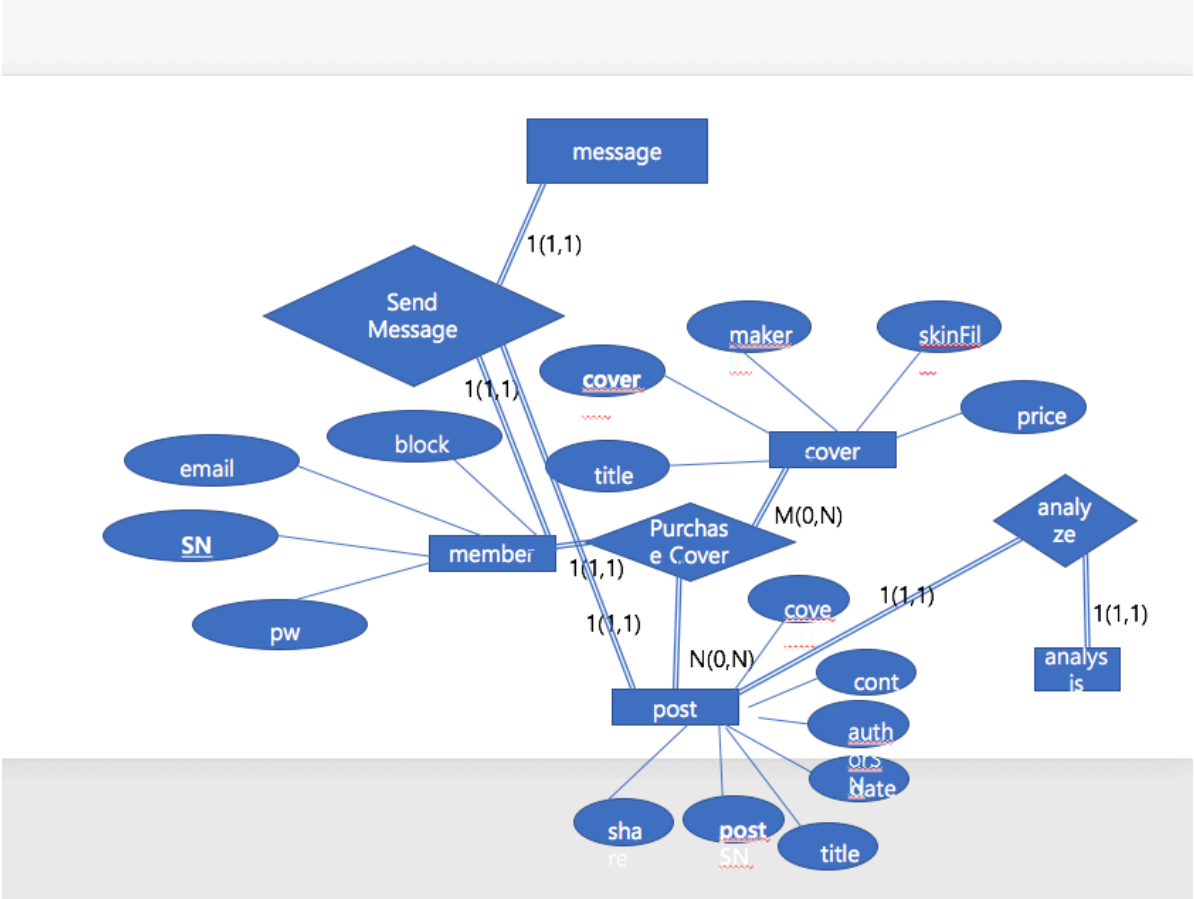

Diagram 35 Overall ER Diagram

개체는 사각형이며, relation은 다이아몬드이다. 개체와 relation은 특성을 가질 수 있으며 원으로 표시된다. 각 개체나 relation은 primary key나 foreign key를 가지고 있다.

A. Entity

A.1. Member

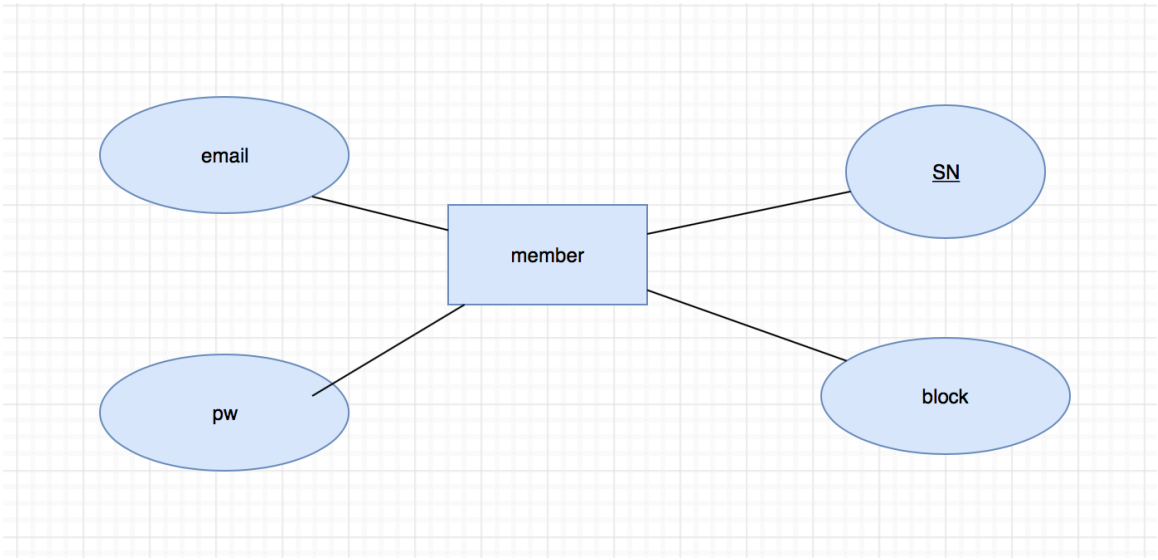

Diagram 36 Member Entity

회원을 관리하는 데이터베이스 이다. SN, email, pw을 저장하고, 신고를 받은 회원을 block값을 통해 식별한다.

A.2. Post

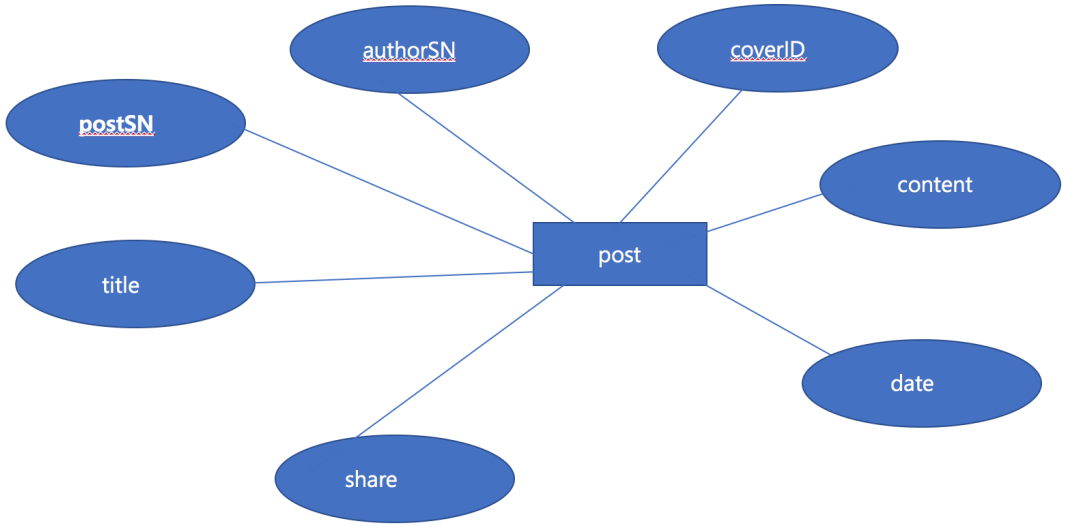

Diagram 37 Post Entity

게시물을 관리하는 데이터베이스이다. postSN으로 구별하며 작성자는 authorSN, 커버테마는 coverID로 저장한다. 공유할지 자신만 볼지는 sharer값을 통해서 구별한다.

A.3. Analyze

Diagram 38 Analyze Entity

감정분석 결과를 저장하는 데이터베이스이다. 게시물 postSN으로 식별하며 분석 결과를 result에 저장한다.

A.4. Message

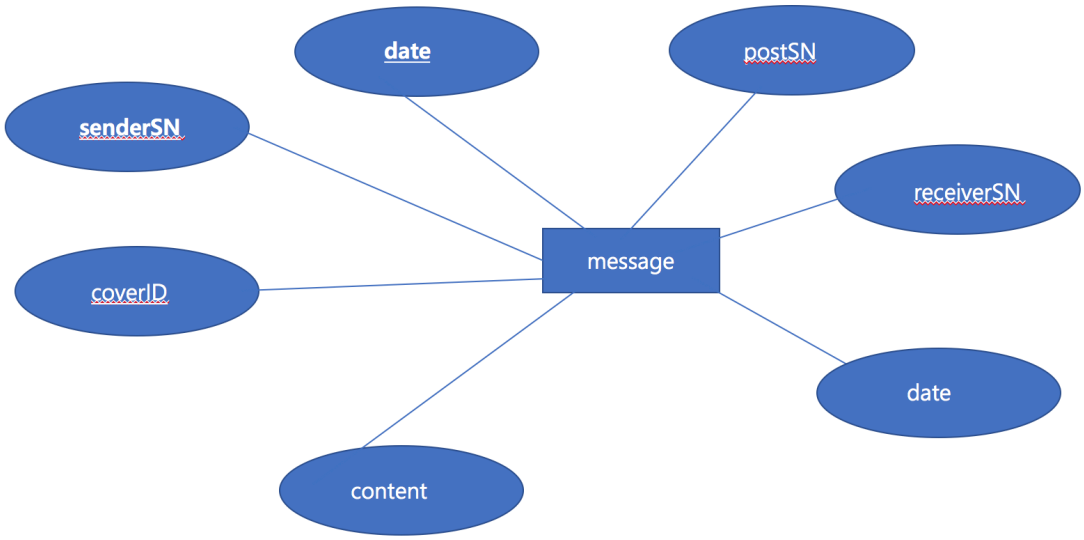

Diagram 39 Message Entity

쪽지를 관리하는 데이터베이스이다. 보낸이 senderSN과 보낸 시각 date로 구별한다. 쪽지에도 coverID로 커버 스킨을 구별한다.

A.5. Community

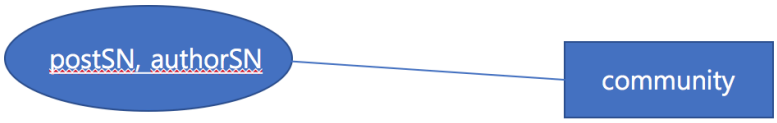

Diagram 40 Community Entity

커뮤니티를 관리하는 데이터베이스이다. postSN과 authorSN으로 게시물을 가져온다.

A.6. Cover

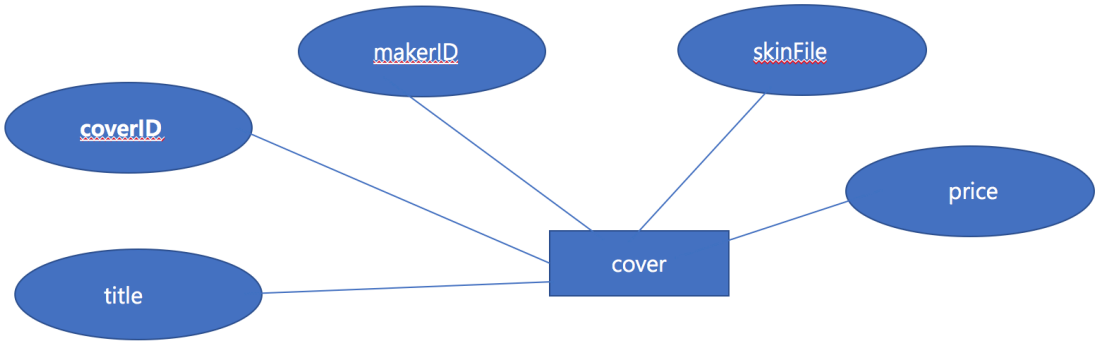

Diagram 41 Cover Entity

커버테마를 저장하는 데이터베이스이다. coverID로 식별하며 만든이 makerID를 저장하며 가격, 이름, 커버파일을 저장한다.

B. Relation

B.1. Write Post

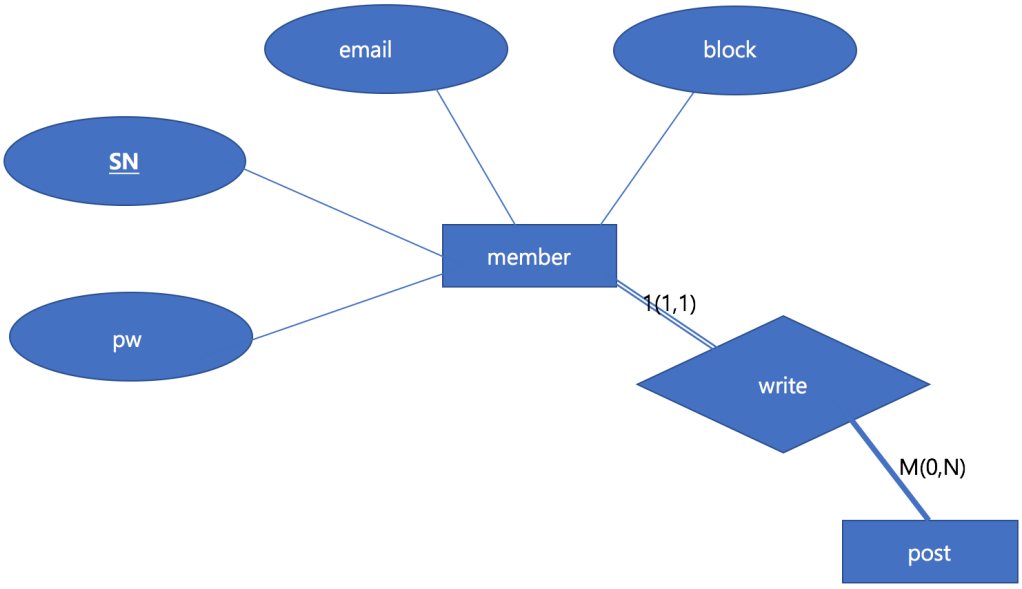

Diagram 42 Write Post Relation

글을 쓰는 relation이다. post와 member테이블에 접근하며 해당 정보를 추가한다.

B.2. Send Message

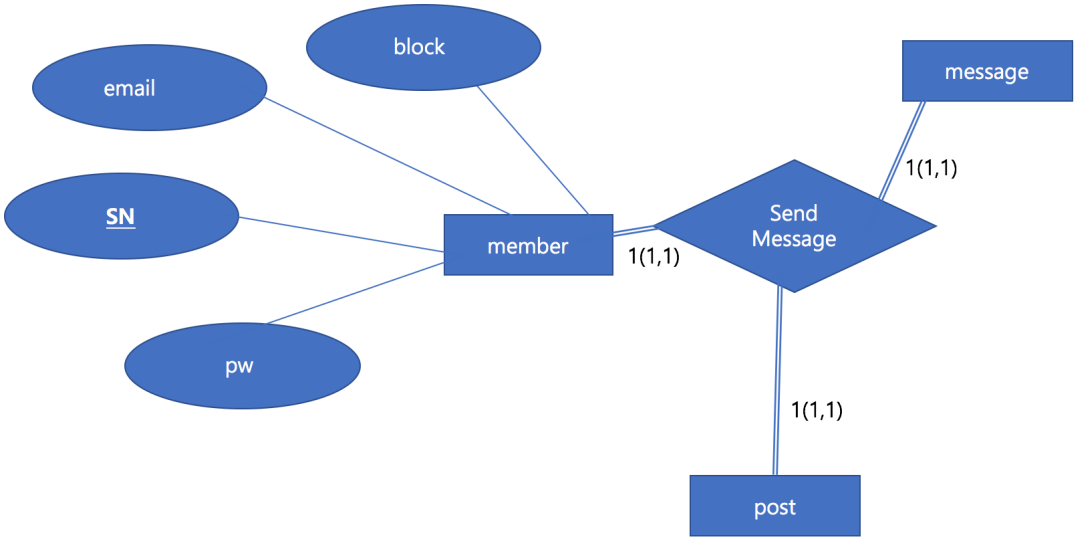

Diagram 43 Send Message Relation

쪽지를 보내는 relation이다. member, message, post 테이블에 접근한다. 쪽지를 보내고 해당 정보를 추가한다.

B.3. Purchase Cover

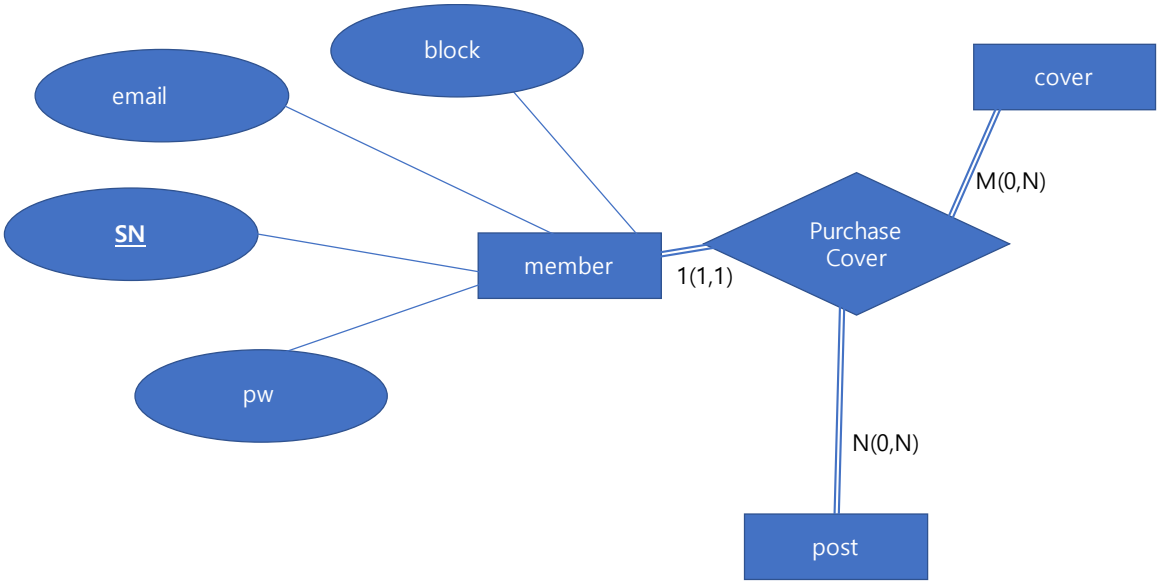

Diagram 44 Purchase Cover Relation

커버를 구입하는 relation이다. member, post, cover테이블에 접근하여 구매 정보를 추가한다.

B.4. Analyze

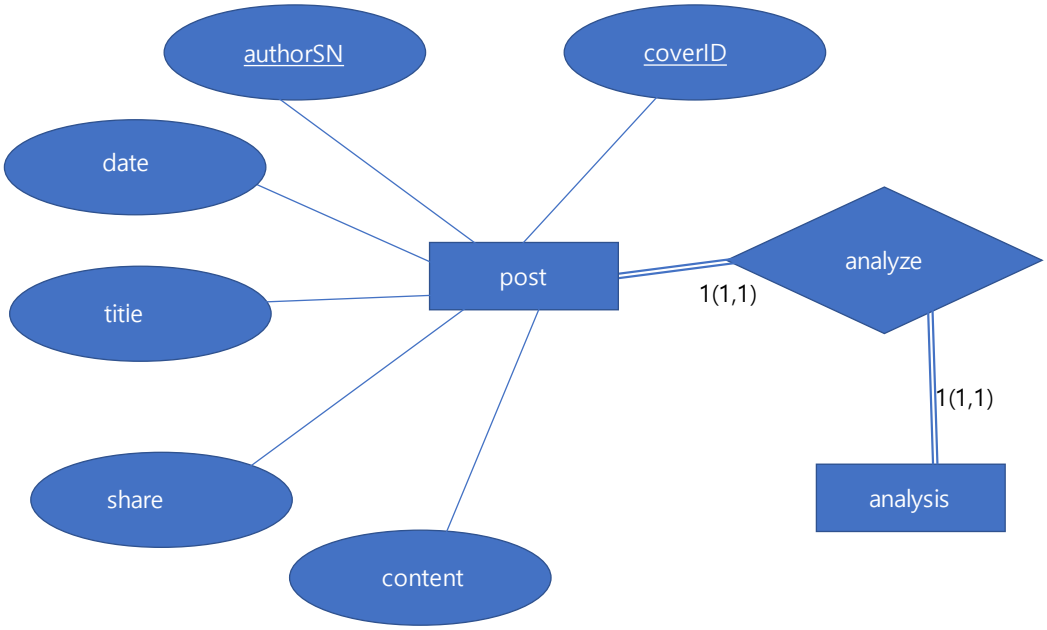

Diagram 45 Analyze Relation

감정분석을 하는 relation이다. post, analyze 테이블에 접근하며 분석결과를 추가한다.

12.3. Relational Schema

A. Member

SN	pw	block	email

Primary Key (PK) : SN

Foreign Key (FK) : 없음

FUNCT DEPT (FD) :

SN-> {pw, block, email}

Description : 기본 회원 관련 정보이므로, 모든 값에 null을 허용하지 않는다.

B. Post

postSN	authorSN	coverID	date
title		share	
content			

Primary Key (PK) : postSN

Foreign Key (FK) : authorSN, coverID,

FUNCT DEPT (FD) :

postSN -> {authorSN, coverID, date, title, share, content}

Description : 게시물에 대한 정보이다. 모든 정보는 null을 허용하지 않는다. 작성자 정보는 authorSN으로, 커버 정보는 coverID로 참조한다.

C. Analyze

postSN	result

Primary Key (PK) : postSN

Foreign Key (FK) : 없음

FUNCT DEPT (FD) :

postSN -> {result}

Description : 게시글의 감정 분석에 대한 정보이다. postSN으로 구별한다.

D. Message

senderSN	**date**	postSN	receiverSN
coverID	content		

Primary Key (PK) : {senderSN, date}
Foreign Key (FK) : postSN, senderSN, receiverSN, coverID
FUNCT DEPT (FD) :
{senderSN, date} -> {postSN, receiverSN, coverID, content}
Description : 쪽지에 관한 정보이다. 쪽지를 보낸이의 SN값과 쪽지를 보낸 시간과 날짜로 구분한다.

E. Community

postSN	**authorSN**		

Primary Key (PK) : {postSN, authorSN}
Foreign Key (FK) : 없음
FUNCT DEPT (FD) :
{postSN, authorSN}
Description : 게시판의 게시글 관련 테이블이다. 게시글 SN과 글쓴이 SN으로 구분한다.

F. Cover

coverID	makerID	skinFIle	price
title			

Primary Key (PK) : coverID
Foreign Key (FK) : makerID
FUNCT DEPT (FD) :
coverID -> {makerID, skinFile, price, title}
Description : coverID로 커버를 구분하고 커버를 만든이를 makerID로 참조한다.

G. Send message

memberSN	postSN	senderSN, date

Primary Key (PK) : {memberSN, postSN, senderSN, date}

Foreign Key (FK) : memberSN, postSN, senderSN, date

Description : memberSN, postSN, senderSN, date로 참조하고 메세지를 생성한다.

H. Purchase Cover

coverID	postSN		

Primary Key (PK) : coverID, postSN

Foreign Key (FK) : coverID, postSN

Description : 커버를 구매하면 coverID, postSN를 참조하여 추가한다.

12.4. SQL DDL

A. Member

```
CREATE TABLE 'Member' (
  'SN' INT(11) NOT NULL AUTO_INCREMENT,
  'pw' VARCHAR(15) NOT NULL,
  'block' INT(3) DEFAULT 0,
  'email' VARCHAR(30) NOT NULL UNIQUE,
  PRIMARY KEY('SN'),
);
```

B. Post

```
CREATE TABLE 'Post' (
    'postSN' INT(15) NOT NULL AUTO_INCREMENT,
    'authorSN' INT(11) NOT NULL,
    'coverID' INT(11) DEFAULT 0,
    'date' DATETIME NOT NULL,
    'title' VARCHAR(50) NOT NULL,
    'share' INT(1) DEFAULT 0,
    'content' VARCHAR(3000) NOT NULL,
    PRIMARY KEY('postSN'),
    FOREIGN KEY ('authorSN') REFERENCES Member('SN') ON DELETE CASCADE
   FOREIGN KEY ('coverID') REFERENCES Cover('coverID') ON DELETE CASCADE ON
     UPDATE CASCADE
);
```

C. Analyze

```
CREATE TABLE 'Analyze' (
    'postSN' INT(15) NOT NULL,
    'result' VARCHAR(100),
    PRIMARY KEY('postSN'),
    FOREIGN KEY ('postSN') REFERENCES Post('postSN') ON DELETE CASCADE ON
     UPDATE CASCADE
);
```

D. Message

```
CREATE TABLE 'Message' (
      'senderSN' INT(11) NOT NULL,
      'date' DATETIME NOT NULL,
      'postSN' INT(15) NOT NULL,
      'receiver' INT(11) NOT NULL,
      'coverID' INT(11) DEFAULT 0,
      'content' VARCHAR(1000) NOT NULL,
      PRIMARY KEY('senderSN', 'date'),
      FOREIGN KEY('postSN') REFERENCES Post('postSN') ON DELETE SET NULL,
      FOREIGN KEY('senderSN') REFERENCES Member('SN') ON DELETE CASCADE,
      FOREIGN KEY('receiverSN') REFERENCES Member('SN') ON DELETE CASCADE,
      FOREIGN KEY('coverID') REFERENCES Cover('coverID') ON DELETE CASCADE ON
        UPDATE CASCADE
);
```

E. Community

```
CREATE TABLE 'Community' (
  'postSN' INT(15) NOT NULL,
  'authorSN' INT(11) NOT NULL,
  PRIMARY KEY('postSN', 'authorSN'),
  FOREIGN KEY('postSN') REFERENCES Post('postSN') ON DELETE CASCADE ON
        UPDATE CASCADE,
  FOREIGN KEY('authorSN') REFERENCES Member('SN') ON DELETE CASCADE
);
```

F. Cover

```
CREATE TABLE 'Cover' (
    'coverID' INT(11) NOT NULL AUTO_INCREMENT,
    'makerID' INT(11) NOT NULL,
    'skinFile' VARCHAR(45) NOT NULL,
    'price' INT(5) DEFAULT 0,
    'title' VARCHAR(50) NOT NULL,
    PRIMARY KEY('coverID'),
    FOREIGN KEY('makerID') REFERENCES Member('SN') ON DELETE CASCADE
);
```

G. Send Message

```
CREATE TABLE 'Send message' (
    'memberSN' INT(11) NOT NULL,
    'postSN' INT(15) NOT NULL,
    'senderSN' INT(11) NOT NULL,
    'date' DATETIME NOT NULL,
    PRIMARY KEY('memberSN', 'postSN', 'senderSN', 'date'),
    FOREIGN KEY('makerID') REFERENCES Member('SN'),
    FOREIGN KEY('postSN') REFERENCES Post('postSN'),
    FOREIGN KEY('senderSN') REFERENCES Member('SN'),
    FOREIGN KEY('date') REFERENCES Message('date')
);
```

H. Purchase Cover

```
CREATE TABLE 'Purchase cover' (
    'coverID' INT(11) NOT NULL,
    'postSN' INT(15) NOT NULL,
    PRIMARY KEY('coverID', 'postSN'),
    FOREIGN KEY('coverID') REFERENCES Cover('coverID'),
    FOREIGN KEY('postSN') REFERENCES Post('postSN')
);
```

13. Testing Plan

13.1. Objectives

시스템의 testing을 설계 단계에 미리 계획함으로써 testing 단계의 overhead를 줄인다. testing은 크게 development testing, release testing, user testing으로 나눠지며, 본 목차에서는 development testing의 plan에 대해 자세하게 기술한다.

13.2. Test Case

A. Sign up and Login System

A.1. Sign up for new users

1) User : 메인 화면에서 회원 가입 버튼을 클릭한다.
2) 시스템 동작 : 회원 가입 페이지로 이동한다.
3) User : 회원 가입 양식(이메일, 비밀번호, 비밀번호 확인)을 작성한 후 제출한다.
4) 시스템 동작 : MemberDB에 중복된 이메일이 있는지, 비밀번호를 바르게 입력했는지를 확인한다.
4-1) (회원 가입 성공) 시스템 알림 : “welcome! you’re successfully registered to To Dot.”
시스템 동작 : 만들어진 계정으로 로그인 된다.
4-2) (회원 가입 실패) 시스템 알림 : “sorry, fill in the form correctly.”

A.2. Login

1) User : 메인 화면에서 메일과 비밀번호를 입력하고, 로그인 버튼을 클릭한다.
2) 시스템 동작 : MemberDB의 정보와 입력받은 정보를 대조한다.
2-1) (로그인 성공) 시스템 알림 : “welcome to To Dot.”
시스템 동작 : 개인 페이지로 이동한다.
2-2) (로그인 실패) 시스템 알림 : “wrong ID or PW. plz try again.”
시스템 동작 : 로그인 화면으로 돌아간다.

B. Post System

B.1. Upload new post

1) User : 글 작성 버튼을 클릭한다.
2) 시스템 동작 : 글 작성 페이지로 이동한다.
3) User : 글의 제목, 내용 등을 입력하고, 커버 테마 및 공유 여부를 선택한다.
3-1) (커버 테마 사용) 시스템 동작 : 보유한 커버 테마를 보여주는 창을 띄우고, 사용자가 구매를 원할 시 구매를 진행한다.
4) User : 업로드 버튼을 클릭한다.
4-1) (업로드 성공) 시스템 동작 : 개인 페이지로 이동한다. postDB를 업데이트한다. 커버를 사용했다면 coverDB를, 그리고 커뮤니티에 공유하기를 설정했다면 communityDB를 업데이트한다.
4-2) (업로드 실패) 시스템 동작 : “the title and contents must have value.”

B.2. Edit existing post

1) User : 글 수정 버튼을 클릭한다.
2) 시스템 동작 : 글 수정 페이지로 이동한다.
3) User : 글 제목, 내용, 커버, 커뮤니티 공유 여부 등을 수정한 후 업로드 버튼을 클릭한다.
4-1) (업로드 성공) 시스템 동작 : 개인 페이지로 이동한다. postDB를 업데이트한다. 커버를 변경했다면 coverDB를, 그리고 커뮤니티 공유 여부를 변경했다면 communityDB를 업데이트한다.
4-2) (업로드 실패) 시스템 동작 : “the title and contents must have value.”

B.3. Delete existing post

1) User : 글 삭제 버튼을 클릭한다.
2) 시스템 알림 : “are you sure to delete this post?”
3) User : 확인 버튼을 클릭한다.
4) 시스템 알림 : “delete post.”
시스템 동작 : postDB에서 해당 글의 DB를 삭제한다. 만약 커뮤니티에 공유된 글이라면 communityDB도 갱신한다.

C. Community System

1) User : 커뮤니티 탭을 클릭한다.
2) 시스템 동작 : 커뮤니티 페이지로 이동한 후 communityDB를 조회하여 등록된 날짜 순으로 글 목록을 보여준다.
3) User : 원하는 글을 클릭하거나, 스크롤을 내려 더 많은 글 목록을 불러온다.

D. Send Message System

1) User : 커뮤니티의 글을 통해 쪽지 보내기 버튼을 클릭한다.
2) 시스템 동작 : 쪽지를 전송할 수 있는 팝업창을 띄운다.
3) User : 쪽지의 내용을 입력하고, 원한다면 커버 설정을 마친 후 전송 버튼을 클릭한다.
4-1) (전송 성공) 시스템 알림 : “message is successfully send.”
시스템 동작 : 수신자와 발신자의 messageDB를 갱신한다.
4-2) (전송 실패) 시스템 알림 : “content must have value.”

E. Evaluation and Block System

E.1. Block Specific User

1) User : 받은 쪽지에 대해 차단하기 버튼을 클릭한다.
2) 시스템 알림 : “the sender is successfully blocked.”
시스템 동작 : 해당 쪽지는 더 이상 보여지지 않는다. 쪽지를 보낸 사람의 block 스택이 1 증가한다. 일정 횟수 이상 차단을 받은 사용자는 관리자가 판단한 후 쪽지 기능이 작동하지 않도록 한다.

F. Feature Analysis System

1) User : 글 분석 탭을 클릭한다.
2) 시스템 동작 : 분석 페이지로 이동한다. AnalysisDB를 조회하여 분석 결과를 보여준다.

G. Cover System

1) User : 글 또는 쪽지 작성 중 커버 사용을 클릭한다.
2) 시스템 동작 : 현재 보유한 커버에 대한 정보, 구매 가능한 커버에 대한 정보 등을 보여준다.
3) User : 구매를 원하는 커버를 클릭한다.
4) 시스템 동작 : 전자결제시스템이 작동하여 결제를 진행한다.
5) 시스템 동작 : 사용자가 새로운 커버를 구매하거나 사용할 때마다 coverDB 및 postDB, message DB가 갱신된다.

14. Development Environment

14.1. Objectives

Development Environment에서는 To Dot(투닷)의 개발 환경에 대해 서술한다.

14.2. Programming Language & Tool

A. Programming Language

Figure 8 Node.js express symbol

Express is a simple, flexible web application development framework based on the Node.js platform that provides a series of powerful features that help you create a variety of Web and mobile device applications. Express does not abstract the characteristics of Node.js for the two time. We just extend the basic functions of Web application on it.

In general, one or more programs designed to carry out operations for a specific purpose. In the context of Express, a program that uses the Express API running on the Node.js platform. May also refer to an app object. Software platform used to build scalable network applications. Node.js uses JavaScript as its scripting language, and achieves high throughput via non-blocking I/O and a single-threaded event loop. This makes it easy to implement a real-time processing web server suitable for the To.Dot system.

Figure 9 MySQL symbol

MySQL is the most popular relational database management system, and in WEB applications MySQL is a very good RDBMS (Relational Database Management System) application software.

Figure 10 Bootstrap symbol

Bootstrap is a free and open-source front-end library for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only.

B. Version Management Tool

Figure 11 GitHub symbol

GitHub is a web-based hosting service for version control using git. It is mostly used for computer code. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

15. Development Plan

15.1. Objectives

Develop plan에서는 Gantt chart를 사용하여 개발 계획에 대해 서술한다.

15.2. Gantt chart

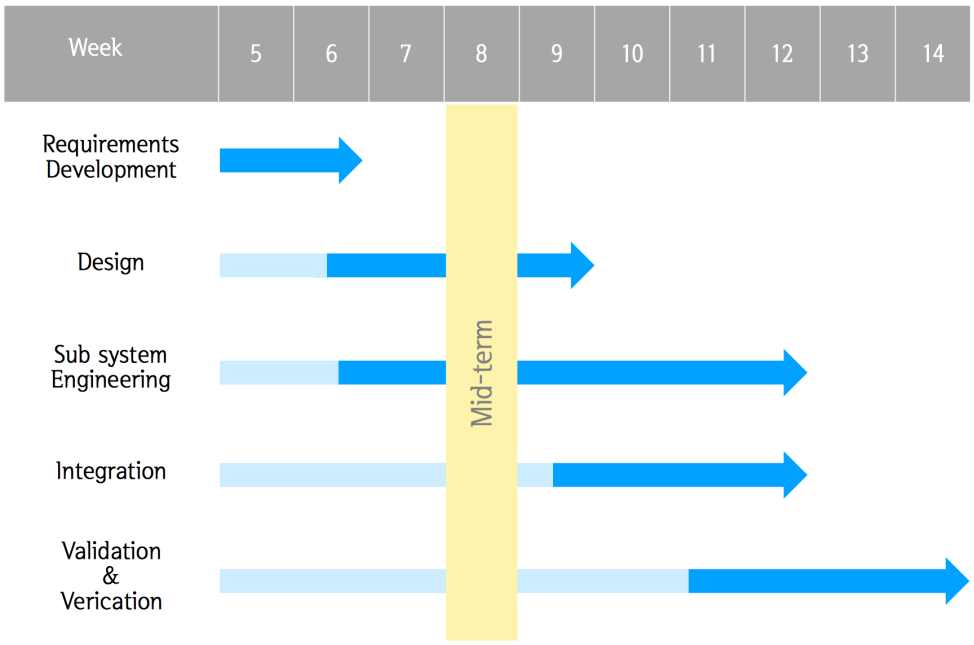

Figure 12 Gantt chart

The development plan and actual development situation are the same as the above Gantt chart. At the initial stage, we have a lot of ideas and investigations and discussions about the project to be studied, and we finally determine the theme, which is a little more than expected in this part.

When the prerequisites are ready, we have to test, design and develop, we spend more time than planned in the design and development stages, which is because the level of knowledge that the team members know is different, and it takes some time to communicate to achieve a certain purpose.

Although some parts are somewhat later than planned, they have no effect on the overall results.

16. Index

16.1. Diagram Index

16.2. Figure Index