****

**Requirement**

**Specification**

**TEAM#4**

**2015310142 컴퓨터 공학과 한성주**

**김하영**

**전재영**

**탕치**

**로호연**

**Contents**

1. PREFACE 9

1.1 OBJECTIVE 9

1.2 READERSHIP 9

1.3 DOCUMENT STRUCTURE 9

A. PREFACE 9

B. INTRODUCTION 9

C. SYSTEM ARCHITECTURE 9

D. USER MANAGEMENT SYSTEM 9

E. COMMUNITY SYSTEM 10

F. SEARCH SYSTEM 10

G. SELLING SYSTEM 10

H. PURCHASE SYSTEM 10

I. PROTOCOL DESIGN 10

J. DATABASE DESIGN 10

K. TESTING PLAN 10

1.4 VERSION HISTORY 11

2. INTRODUCTION 11

2.1 OBJECTIVES 11

2.2 APPLIED DIAGRAM 11

A. UML 12

B. CLASS DIAGRAM 13

C. SEQUENCE DIAGRAM 14

D. STATE DIAGRAM 15

E. E-R DIAGRAM 16

2.3 APPLIED TOOL 16

A. ATOM 17

B. DRAW.IO 18

C. TENSORFLOW 19

3. SYSTEM ARCHITECTURE 19

3.1 OBJECTIVE 19

3.2 SYSTEM ORGANIZATION 19

A. USER MANAGEMENT SYSTEM 20

B. ORDER MANAGEMENT SYSTEM 21

C. DISTANCE COMPUTING SYSTEM 22

D. PURCHASING SYSTEM 22

3.3 PACKAGE DIAGRAM 23

3.4 DEPLOYMENT DIAGRAM 24

4. USER MANAGEMENT SYSTEM 25

4.1. OBJECTIVES 25

4.2. COMMUNICATION DIAGRAM 25

A. ACCOUNT 26

B. BUSINESS 26

C. DB HANDLER 27

D. COMMUNITY SYSTEM 27

4.3. SEQUENCE DIAGRAM 27

A. SIGN IN 27

B. LOGIN 27

4.4 STATE DIAGRAM 28

A. SIGN IN 28

B. LOGIN 29

5. COMMUNITY SYSTEM 29

5.1. OBJECTION 29

5.2. COMMUNICATION DIAGRAM 29

A. ACCOUNT 30

B. BUSINESS 30

C. DB HANDLER 31

D. COMMUNITY SYSTEM 31

5.3 SEQUENCE DIAGRAM 31

A. VIEW 31

B. WRITE 32

C. EDIT 34

5.4 STATE DIAGRAM 35

A. VIEW 35

B. WRITE 36

6. SEARCH SYSTEM 39

6.1 OBJECTION 39

6.2 SEARCH DIAGRAM 39

A. ACCOUNT 39

B. BUSINESS 40

C. DB HANDLER 40

D. SEARCH SYSTEM 40

6.3 SEQUENCE DIAGRAM 41

A. VIEW 41

B. WRITE 42

C. EDIT 43

6.4 STATE DIAGRAM 44

A. VIEW 44

B. WRITE 45

7. SELLING SYSTEM 45

7.1. OBJECTION 46

7.2. SELLING DIAGRAM 46

A. ACCOUNT 46

B. BUSINESS 47

C. DB HANDLER 47

D. SELLING SYSTEM 47

7.3 SEQUENCE DIAGRAM 47

A. VIEW 47

B. WRITE 48

C. EDIT 50

7.4 STATE DIAGRAM 52

A. VIEW 52

B. WRITE 53

8. PURCHASE SYSTEM 54

8.1 OBJECTIVES 54

8.2 CLASS DIAGRAM 55

A. DATABASE HANDLER 55

A.1 ATTRIBUTES 55

A.2 METHODS 55

B. PURCHASING USER 55

B.1 ATTRIBUTES 55

B.2 METHODS 56

C. FOOD 56

C.1 ATTRIBUTES 56

C.2 METHODS 56

D. PAYMENT 56

D.1 ATTRIBUTES 56

D.2 METHODS 56

8.3 SEQUENCE DIAGRAM 56

8.4 STATE DIAGRAM 57

9. PROTOCOL DESIGN 58

9.1 OBJECTIVES 58

9.2 JSON 58

9.3 PROTOCOL DESCRIPTION 59

A. OVERVIEW 59

B. SIGN UP PROTOCOL 59

B.1 REQUEST 59

B.2 RESPONSE 60

C. LOGIN PROTOCOL 60

C.1 REQUEST 60

C.2 RESPONSE 60

D. FOOD SALES REGISTRATION / EDIT PROTOCOL 61

D.1 REQUEST 61

D.2 RESPONSE 61

E. FOOD RECOMMENDATION PROTOCOL 61

E.1 REQUEST 61

E.2 RESPONSE 61

F. SEARCH FOOD PROTOCOL 62

F.1 REQUEST 62

F.2 RESPONSE 62

G. PURCHASE PROTOCOL 63

G.1 REQUEST 63

G.2 RESPONSE 63

H. REVIEWS AND RATINGS PROTOCOL 63

H.1 REQUEST 63

H.2 RESPONSE 63

I. RECIPE POST / EDIT PROTOCOL 64

I.1 REQUEST 64

I.2 RESPONSE 64

J. RECIPE CONFIRMATION PROGOCOL 64

J.1 REQUEST 64

J.2 RESPONSE 64

K. COMMUNITY POST / EDIT PROTOCOL 65

K.1 REQUEST 65

K.2 RESPONSE 65

L. COMMUNITY SEARCH PROTOCOL 65

L.1 REQUEST 65

L.2 RESPONSE 65

10. DATABASE DESIGN 66

10.1 OBJECTIVES 66

10.2 E-R DIAGRAM 66

A. ENTITY 67

A.1 USER 67

A.2 TRANSACTION 67

A.3 FOOD 68

A.4 RATE 69

A.5 COMMUNITY 69

A.6 RECIPE 70

B. RELATIONSHIP 70

B.1 COMMUNITY READ/WRITE 70

B.2 FOOD REVIEW 71

B.3 RECIPE READ/WRITE 72

B.4 PAYMENT 74

B.5 FOOD REGISTRATION 74

10.3 RELATIONAL SCHEMA 75

A. USER 75

B. FOOD 75

C. TRANSACTION 75

D. RATE 76

E. COMMUNITY 76

F. RECIPE 77

10.4 NORMALIZATION 77

10.5 SQL DDL 78

A. USER 78

B. FOOD 78

C. TRANSACTION 79

D. RATE 79

E. COMMUNITY 80

F. RECIPE 80

11. TESTING PLAN 81

11.1 OBJECTIVE 81

11.2 TESTING POLICY 81

A. DEVELOPMENT TESTING 81

B. RELEASE TESTING 82

C. USER TESTING 82

11.3 TEST CASE 82

A. USER MANAGEMENT SYSTEM 82

A.1 SIGN UP FOR USER 82

A.2 LOG IN 82

B. COMMUNITY SYSTEM 83

B.1 COMMUNITY POST WRITE 83

B.2 COMMUNITY POST EDIT 83

C. SEARCH SYSTEM 83

C.1 NEAR FOOD SEARCH 83

D. SELLING SYSTEM 84

D.1 FOOD UPLOAD 84

D.2 NOTIFICATION OF PURCHASING 84

E. PURCHASE SYSTEM 84

E.1 PURCHASE FOOD 84

12. DEVELOPMENT PLAN 85

12.1 OBJECTIVES 85

12.2 GANTT CHART 85

13. REFERENCE 86

**1. Preface**

* 1. **Objective**

Preface에서는 본 문서의 독자를 정의한 후 문서의 전체 구조와 각 목차를 소개한다. 또한 Version의 변화를 기록하여 변경 사항을 기술한다.

* 1. **Readership**

본 문서의 독자는 ‘For Single Life’ 시스템의 설계와 개발, 유지 보수에 참가하는 모든 구성원들로 정의한다. 서브 시스템(Subsystem)과 시스템을 설계하는 시스템 엔지니어, 시스템을 개발하는 소프트웨어 엔지니어, Stakeholder 등 시스템 개발과 설계에 관련된 구성원들을 대상으로 한다.

* 1. **Document structure**

1. **Preface**

Preface에서는 본 문서의 독자를 정의한다. 또 문서를 이루는 세부 사항들과 각 사항들의 목적을 기술한다.

1. **Introduction**

Introduction에서는 본 시스템에 대한 개략적인 설명과, 사용한 다이어그램 및 툴에 대해서 설명한다.

1. **System Architecture**

System Architecture에서는 ‘For Single Life’ 시스템의 구조를 UML, Class Diagram, State Diagram, Sequence Diagram등을 이용해서 소개한다.

1. **User Management System**

User Management System에서는 회원가입, 로그인 같은 ‘For Single Life’를 이용하는 사용자들을 관리하는 시스템에 대한 설계를 설명한다.

1. **Community System**

Community System에서는 ‘For Single Life’ 시스템을 이용하는 유저 들이 사용하는 커뮤니티 시스템에 대한 설계를 설명한다. 게시글 및 댓글 작성 같은 기능을 포함한다.

1. **Search System**

Search System에서는 원하는 음식을 찾을 수 있는 기능을 제공한 다. 검색창에서 단어를 입력하면 레시피나 커뮤니티에 게시된 글에서 해당 단어가 포함된 글들을 순서대로 보여줄 수 있어야 한다.

1. **Selling System**

Selling System에서는 판매자가 음식을 등록하고 판매신청이 들어오면 배달을 보내고 돈을 받는 과정을 포함하는 기능들을 설명한다. 음식의 등록, 등록된 계좌로의 거래 등의 기능들이 있다.

1. **Purchase System**

Purchase System에서는 구매자가 음식을 검색한 다음 원하는 음식을 선택하면 결제를 할 수 있게 해야 한다. 결제는 모바일 지원이 되는 페이 시스템을 사용한다.

1. **Protocol Design**

Protocol Design에서는 Software에서 어떤 방식으로 각 Sub-System간의 데이터를 주고받는지 Setting하고 Communicate하는 세부적인 사항에 대해 설명한다.

1. **Database Design**

Database Design에서는 데이터 베이스 요구사항을 바탕으로 E-R Diagram을 작성한 후 Relation Schema 과정을 거쳐서 SQL을 작성한다.

1. **Testing Plan**

Testing Plan 에서는 시스템이 설계한 대로 작동하는지를 평가한다. 작동 과정에서의 문제가 생기는 상황을 방지하기 위해 시나리오 등을 작성하도록 한다.

* 1. **Version History**

|  |  |  |
| --- | --- | --- |
| Version | Change Date | Update History |
| 1.0 | 2019.05.16 | 문서 생성 |
|  |  |  |
|  |  |  |

1. **Introduction**
   1. **Objective**

Introduction에서는 본 문서에서 시스템을 설계할 때 어떤 Diagram과 Tool을 사용할 것인지를 기술한다.

* 1. **Applied Diagram**

1. **UML (Unified Modeling Language)**

Figure 2.1 UML

UML은 소프트웨어 공학에서 사용되는 국제적으로 표준화된 모델링 기법이다. UML은 객체 지향 분석 설계를 위해 각 단계별 결과를 도식적으로 시각화한 Diagram들의 통합이다. 시스템에 대한 시각적 정보를 제공해주기 때문에 사용자와 개발자 간의 Communication을 도와준다. 객체 지향에 큰 공헌을 한 그래디 부치(Grady Booch), 제임스 럼버(James Rumbaugh), 이바 야콥슨(Ivar Jacobson)이 고안하고 설계한 UML의 초안은 1997년 UML 컨소시엄에서 Oracle, Microsoft, HP등의 기업들이 중심이 되어 Ver 1.0이 탄생하였다. 이후 꾸준히 관리, 개선이 되었다. UML의 다이어그램은 현재 총 13가지가 있으며, 우리는 그 중에서 4가지를 사용한다.

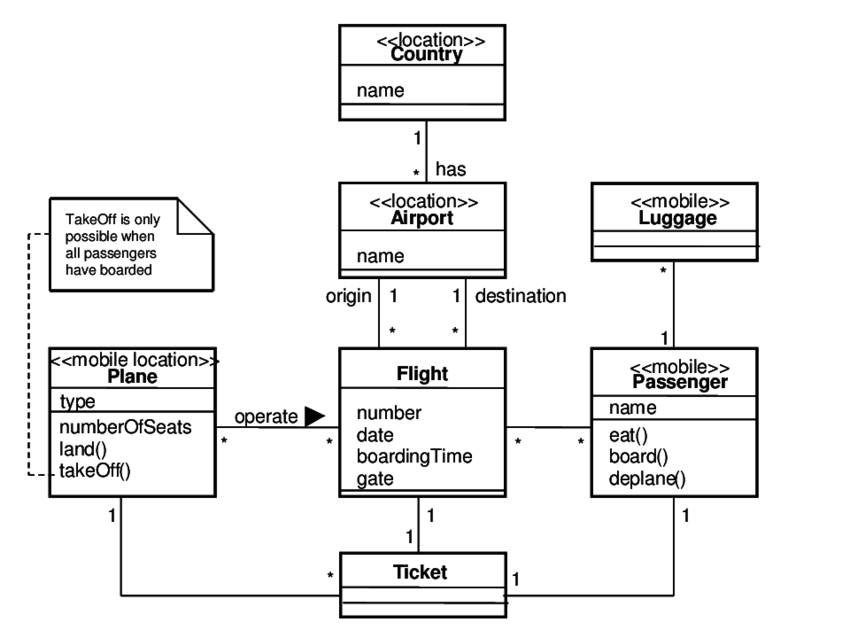
1. **Class Diagram**

Figure 2.2 Class Diagram

Class Diagram은 시스템의 클래스, 속성, 작업이나 메소드 및 객체간 관계를 보여줌으로써 시스템의 구조를 설명하는 정적 구조의 다이어그램이다. 클래스 다이어그램의 클래스는 주요 요소, 응용 프로그램의 상호 작용 및 프로그래밍할 클래스를 모두 나타낸다. 클래스 다이어그램의 경우 시스템 내부에 존재하는 클래스들을 선별하여 나타낸다. 각 클래스에는 Attribute와 Behavior가 포함되어야 한다. 또한 클래스 사이의 관계도 고려되어야 한다. 상속관계, 의존관계 등 클래스간 상호 관계가 나타날 수 있다. 그렇기 때문에 항상 추상화를 고려해서 작성하여야 한다.

1. **Sequence Diagram**

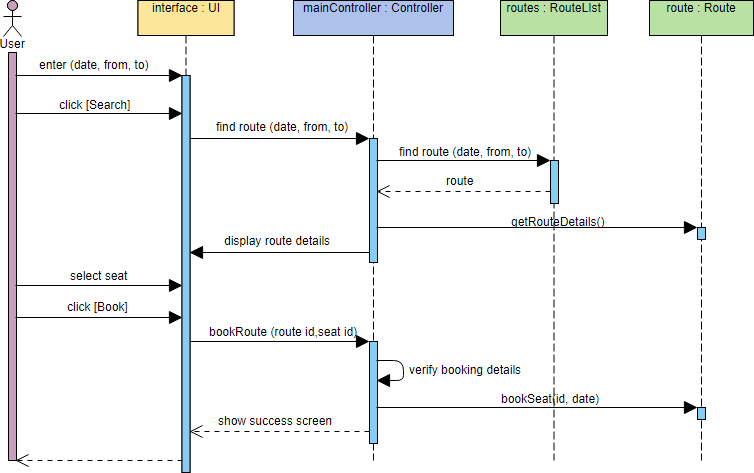


Figure 2.3 Sequence Diagram

Sequence Diagram은 시간 순서대로 정렬된 객체 간의 상호 작용을 보여준다. Sequence Diagram은 시스템의 동적인 부분을 나타내는 다이어그램 중 하나이다. 시스템이 실행되면 생성되거나 소멸되는 객체를 표기한다. 또한 객체들 사이에서 주고받는 메시지도 표시한다. 메시지의 발생보다는 메시지의 발생 과정이나 순서가 중요하기 때문에 이를 중점으로 표기한다. 메시지의 전달 순서는 수직 축과 수평 축으로 표기한다. 수직 축은 메시지가 발생한 시간 순서를 축의 상단에서 하단으로 진행하는 방향으로 나타내고 수평 축은 화살표의 방향으로 송신객체와 수신객체를 구분한다.

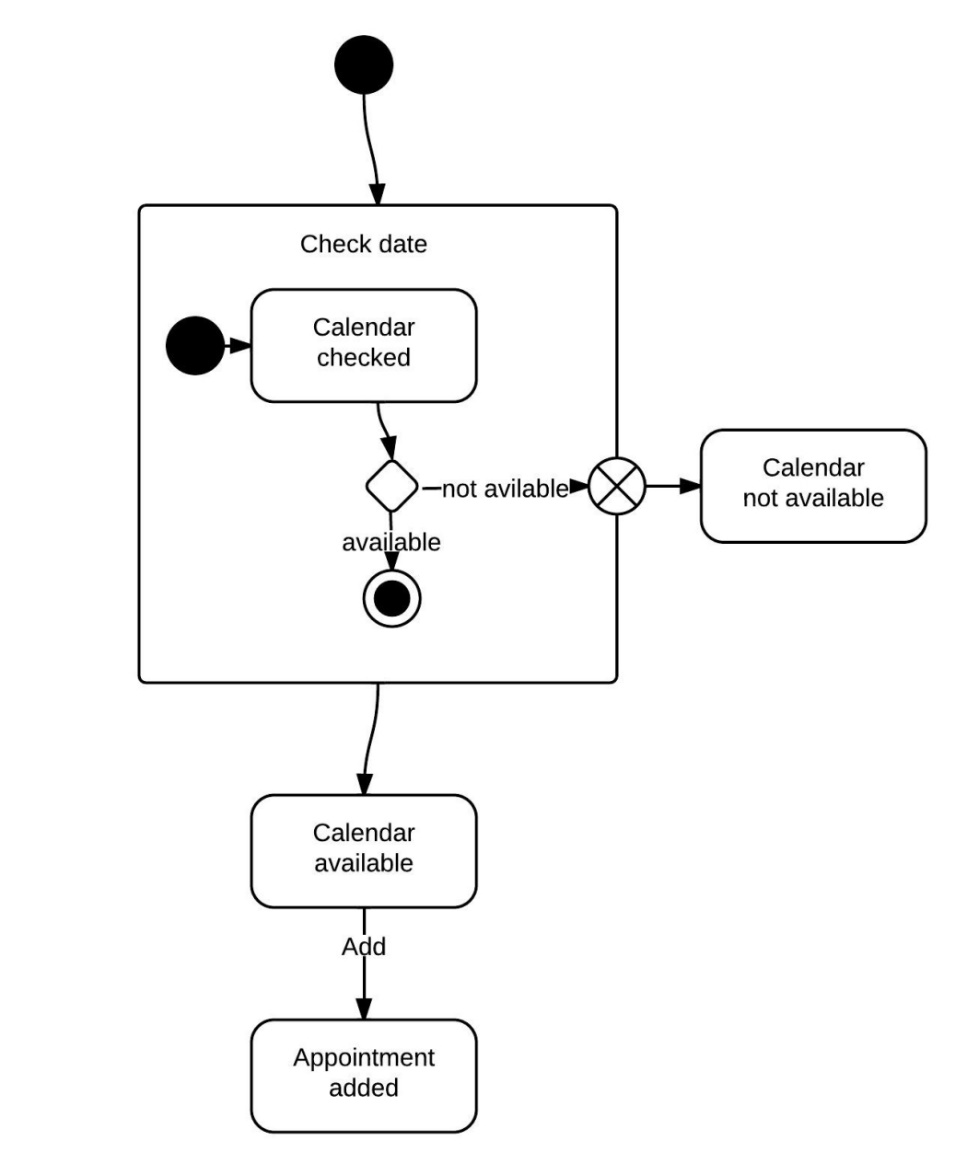
1. **State Machine Diagram**

Figure 2.4 State Diagram

State Machine Diagram은 객체 지향 모델에서 클래스의 인스턴스 이벤트(Event)에 의거한 시스템의 전체적인 작동을 상세하게 기술하는 다이어그램이다. State Machine Diagram은 상태의 변화에 의한 동작 또는 하나의 상태에서 다른 상태로 변화하는 사건의 주어진 시간 동안의 상태를 나타낸다. 시스템의 실행 시 많은 객체가 생성되고 소멸되기 때문에 특정 조건을 만족하는 기간 동안의 중요한 객체에 관하여 표시하여야 한다.

1. **E-R Diagram**

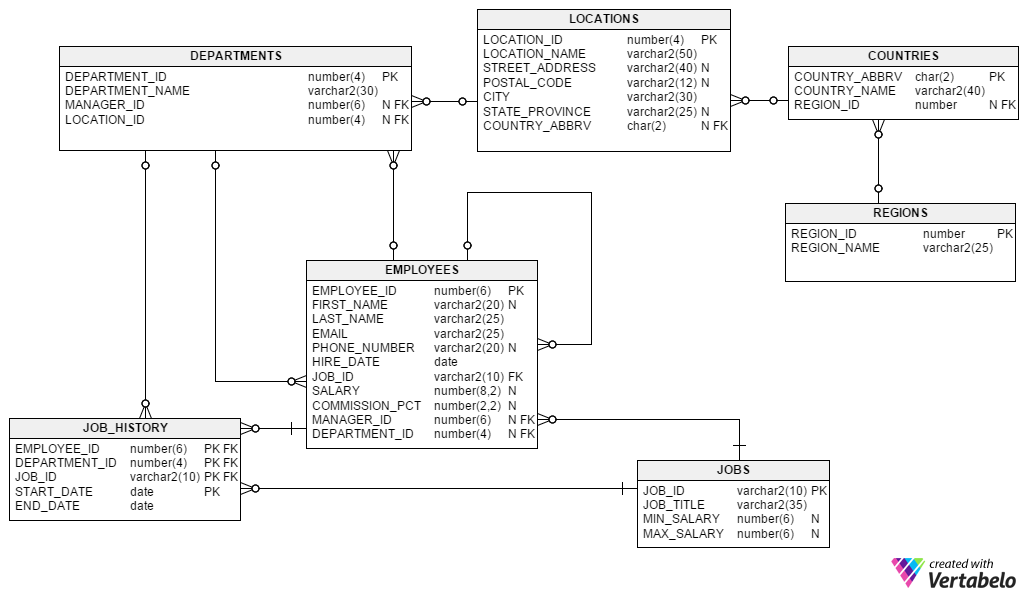


Figure 2.5 E-R Diagram

Entity - Relationship Diagram(이하 E-R Diagram)은 데이터베이스에서 각 개체들 간의 관계를 표현하기 위한 UML이 아닌 다이어그램이다. Entity는 현실에서의 개체를 뜻한다. Relationship은 두 개 이상의 개체간 상호관계를 의미한다. E-R Diagram을 통해 제약조건(Constraint)를 정의할 수 있다. E-R Diagram은 각 조직의 정보자원을 관리하는데 있어서 매우 유용한 도구이다.

* 1. **Applied Tool**

1. **Atom**

Figure 2.6 Atom Editor

개발 과정에서 사용한 에디터는 Atom이다. Atom Editor은 Github에서 만든 Electron 프레임워크 기반의 텍스트 에디터이다. 다양한 프로그래밍 언어의 편집기로 이용할 수 있으며, Linux, Windows, OS X등의 다양한 운영체제에서 독립적으로 사용이 가능하다. 또한 패키지 및 테마를 자유롭게 설정할 수 있는 장점이 있다.

1. **Draw.io**

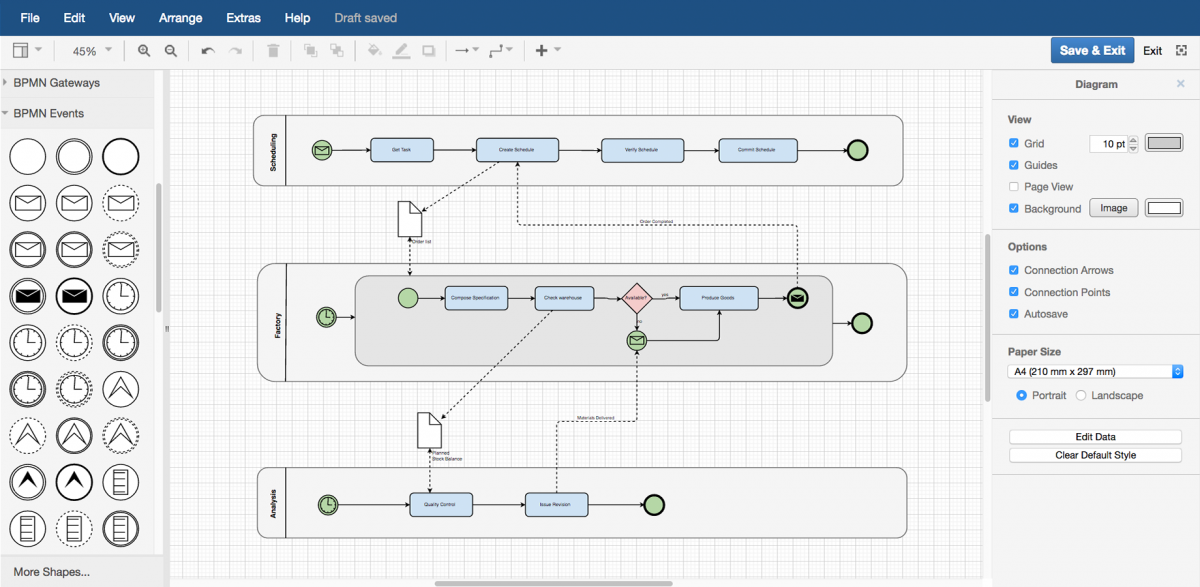


Figure 2.7 Draw.io

본 문서에 그림으로 첨부된 Diagram들은 draw.io를 사용하여 작성했다. 웹 기반 서비스로 운영되기 때문에 우수한 접근성과 범용성을 가지고 있으며 다양한 UML을 작성할 수 있다. Github, Dropbox와도 연동이 되기 때문에 공유 기능을 사용하여 쉽게 협업이 가능하다는 장점을 가지고 있다.

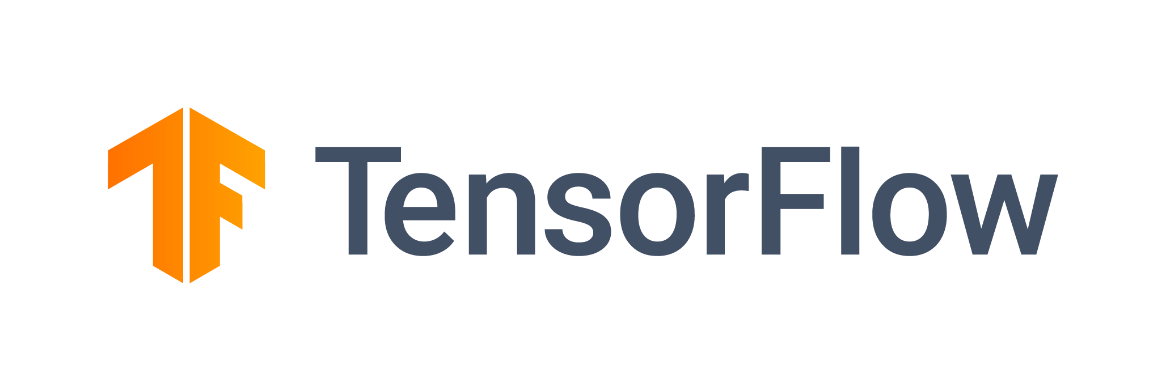
1. **Tensorflow**

Figure 2.8 Tensorflow

검색 기능을 강화하기 위해서 기계학습 라이브러리인 텐서플로우를 사용한다. 사용자가 검색한 기록들이 일정 수 이상 모이면 기록들을 바탕으로 자주 검색한 단어들과 연관되는 내용을 찾아서 로그인 후 첫 화면에 추천 음식을 보여준다.

**3. System Architecture**

**3.1. Objective**

System architecture design part, as a tool to assist thinking, helps us keep our attention on the most important business concepts and their relationships, and enables us to analyze and understand requirements in depth and systematically.Domain modeling is often a process from fuzzy to clear, from scattered to system

**3.2 SYSTEM ORGANIZATION**

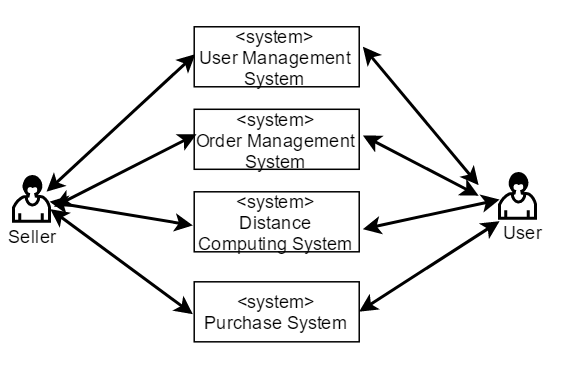


Figure 3.1 System Oranization Block Diagram

The System structure of “For Single Life” as shown in the figure, is composed of User Management System, Order Management Sustem, Distancs Computing System and Purchase System, and realizes all functions of the software through the four main systems

1. **User Management System**

User Management System is a system to manage the user’s information. It provide two kinds of functions: login and sign in.

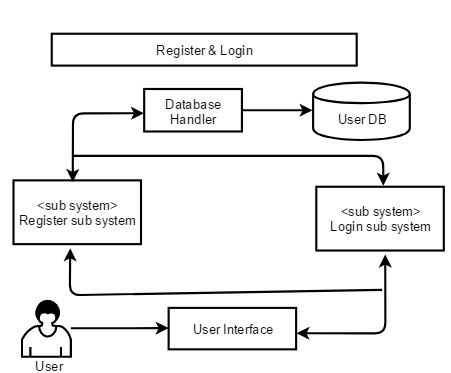
.

Figure 3.2 User Management System Architecture

1. **Order Management System**

Order management system is a system to manage the order, There are two sub systems, Browse System is to support browse the order or menu, Suggestion system can suggest some menu at front page.

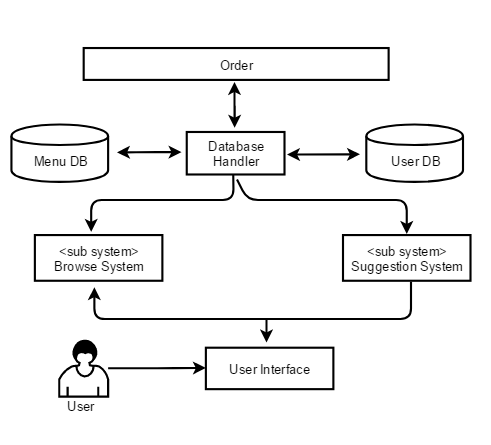


Figure 3.3 Order Management System Architecture

1. **Distance Computing System**

The system USES machine learning algorithms to optimize the distance between the user and the seller.The Map part USES the Map API of Google, and the subsystem consists of Address upload sub system, Distance Calculation sub system and Position upload sub system

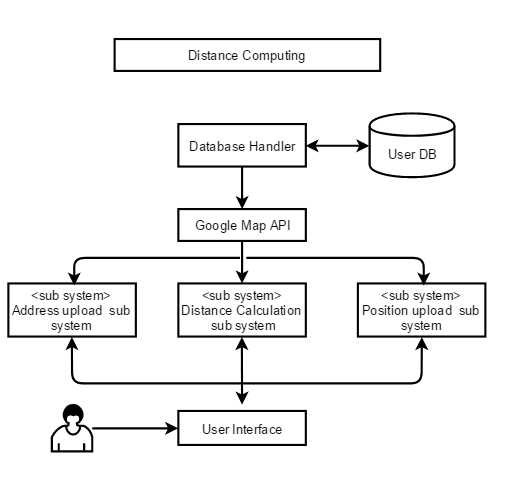


Figure 3.4 Distance Calculating System Architecture

1. **Purchase System**

Purchase System is  a system that connects sellers and buyers

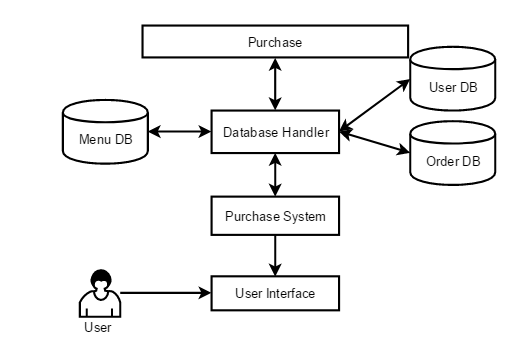


Figure 3.5 Purchase System Architecture

**3.3 Package Diagram**

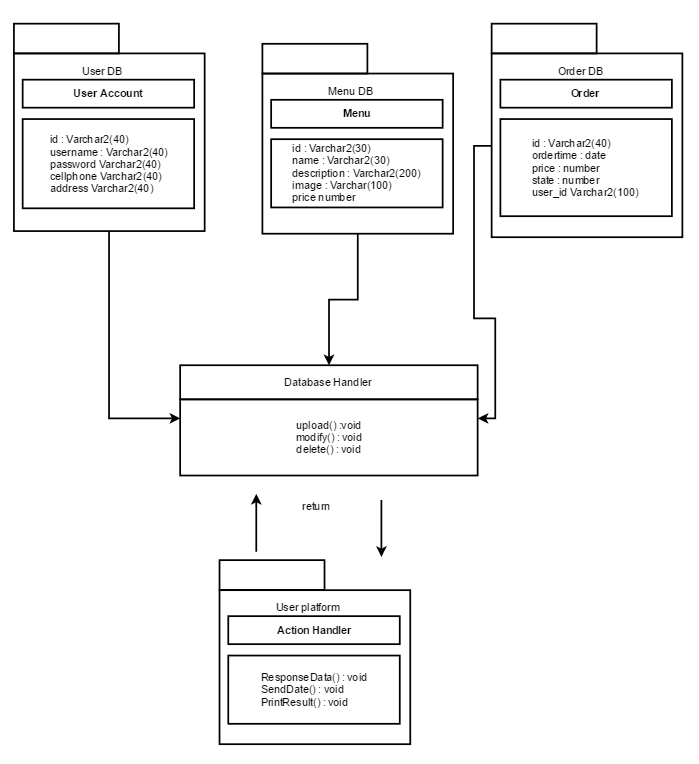


Figure 3.6 Package Diagram

**3.4 Deployment Diagram**

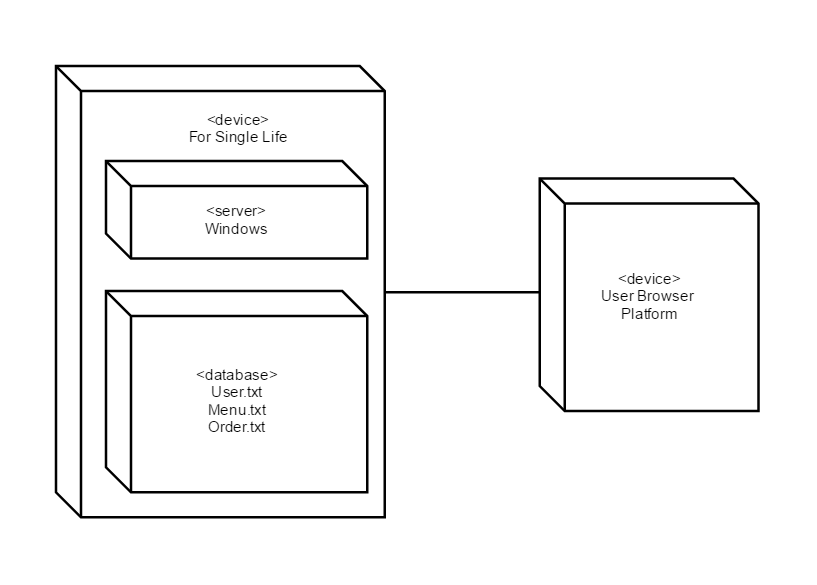


Figure 3.7 Deployment Diagram

**4. User Management System**

**4.1 Objectives**

Users need to register and log in through the system before using this system. This part will illustrate the operation of data generated in this process.The Class diagram, Sequence diagram and State diagram are included.The above three parts constitute User management system.

**4.2 Communication Diagram**

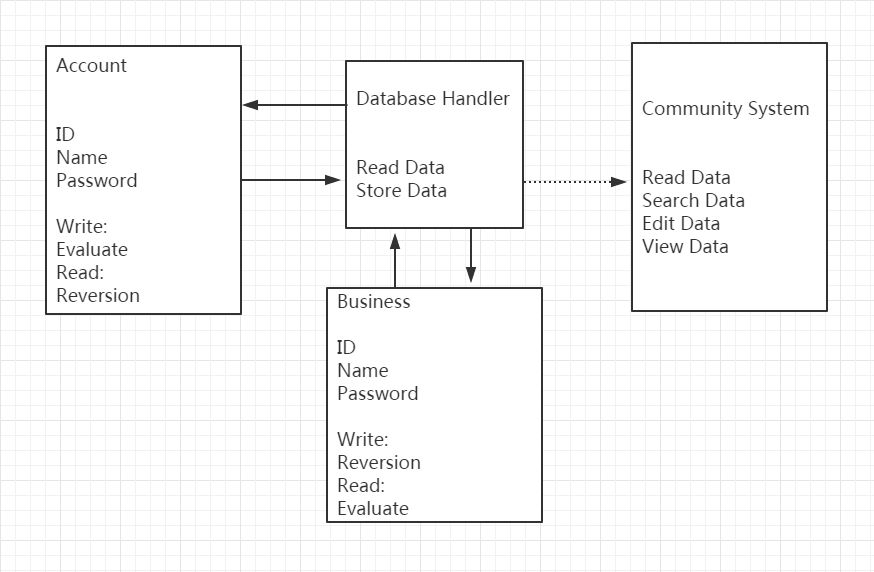
****

Figure 4.1 User Management System Class Diagram

**A.Account**

Account ID: Customer account number

Account Name: Customer Network Name

Account Password: Customer account password

Write Evaluate: Customer Evaluation of Purchased Goods

Read Reversion: Commerce response to evaluation

**B.Business**

Business ID: Business account number

Business Name: Shop Name of Merchant

Business Password: Business account password

Read Evaluate: Businessmen can read customer reviews

Write Reversion: Businessmen can write reversion

**C.DB Handler**

Read Date: Database reading data

Store Date: Database storing data

**D.Community System**

Read Date: System can read data

Search Date: System can search data

Edit Date: System can edit data

View Date: System can view data

**4.3 Sequence Diagram**

**A. Sign in**

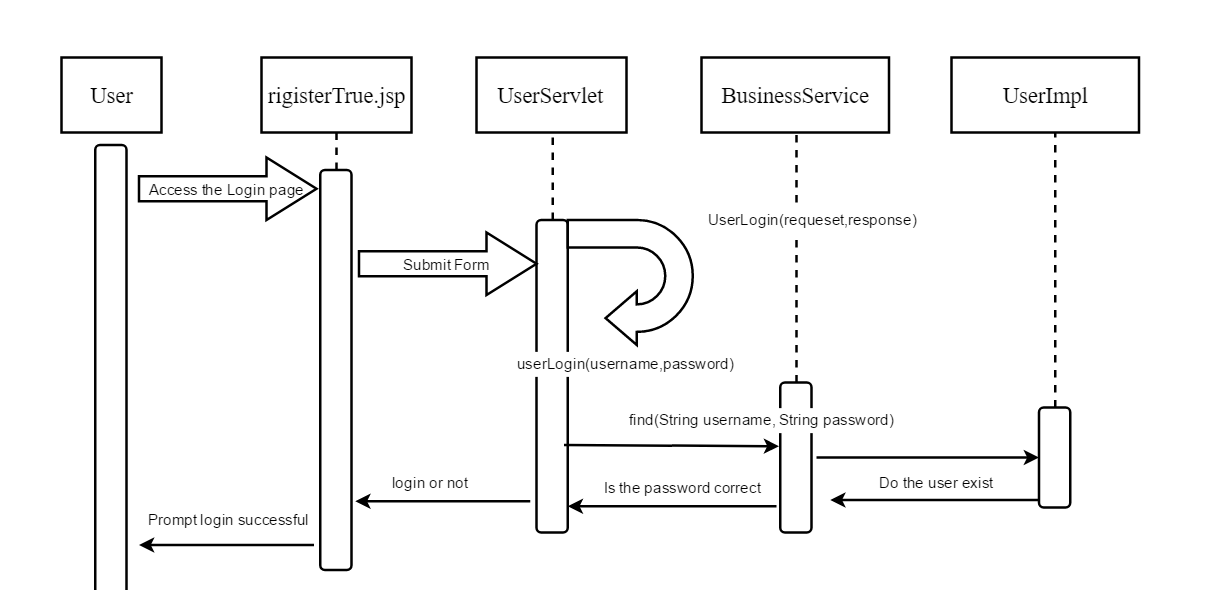


Figure 4.2 User Management System Sequential Diagram-sign in

**B. Login**

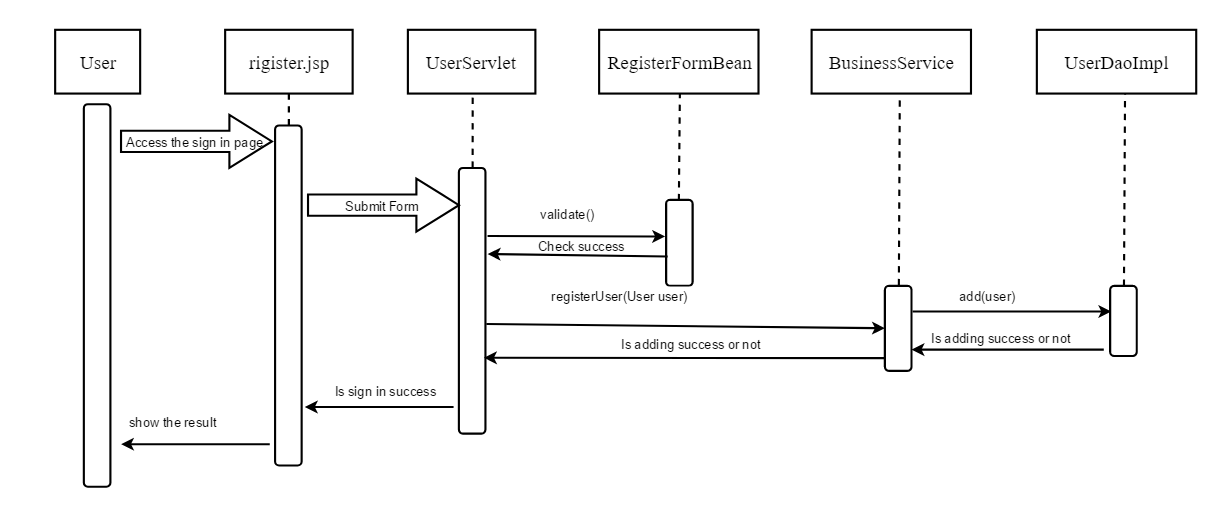


Figure 4.3 User Management System Sequential Diagram-login

**4.4 State Diagram**

**A. Sign in**

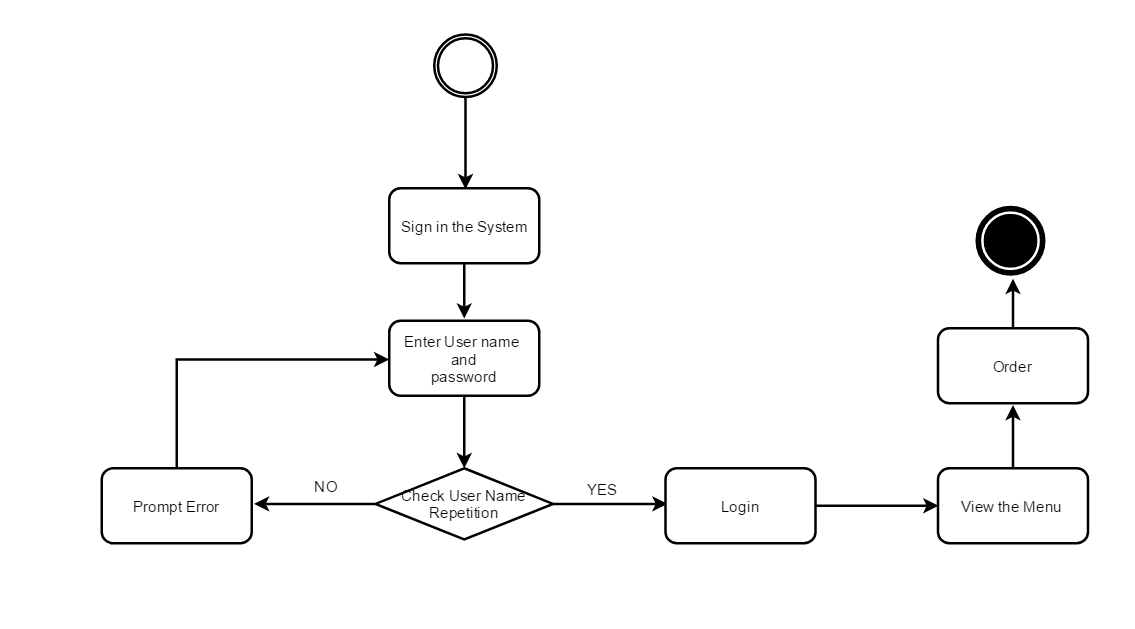


Figure 4.4 User Management System State Diagram-sign in

**B. Login**

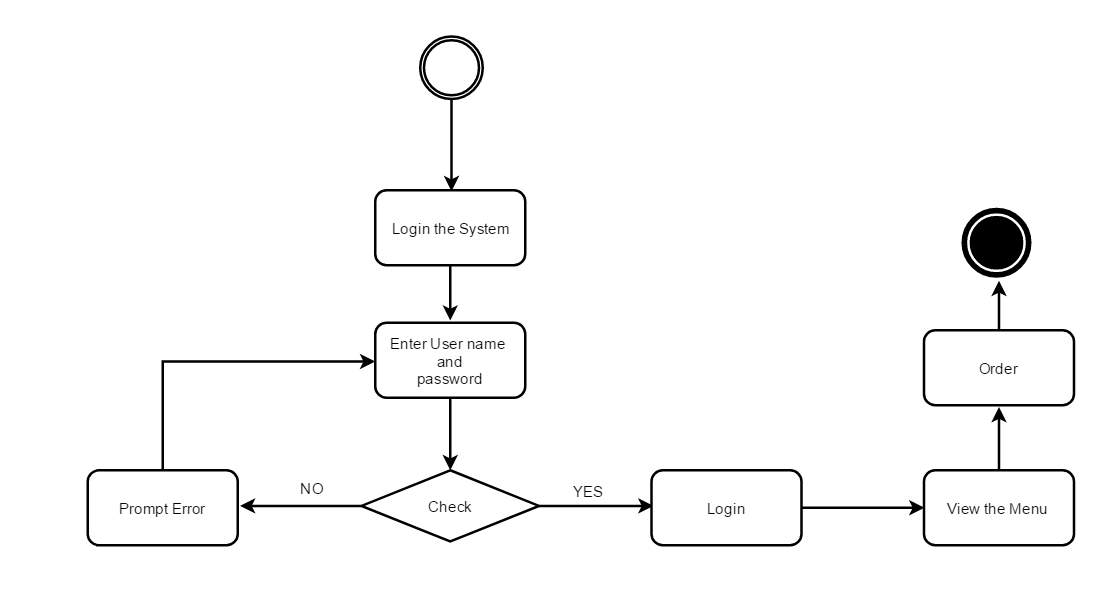


Figure 4.5 User Management System State Diagram-login

**5.Community System**

**5.1 Objection**

This chapter describes the community system between customers and businesses. This system allows customers to evaluate the goods they have purchased, and businesses can see the evaluation and respond to the evaluation.

**5.2 Communication Diagram**

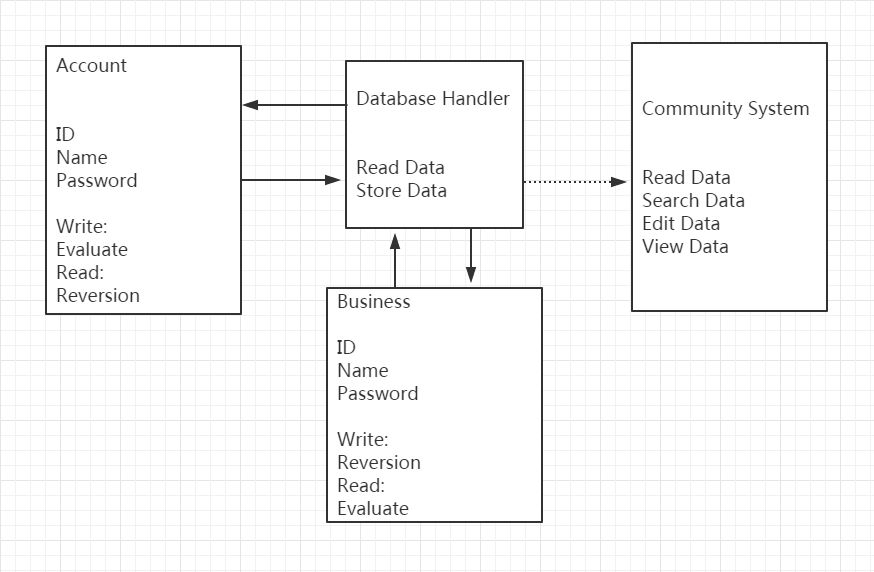
****

Figure 5.1 Diagram for community system

**A.Account**

Account ID: Customer account number

Account Name: Customer Network Name

Account Password: Customer account password

Write Evaluate: Customer Evaluation of Purchased Goods

Read Reversion: Commerce response to evaluation

**B.Business**

Business ID: Business account number

Business Name: Shop Name of Merchant

Business Password: Business account password

Read Evaluate: Businessmen can read customer reviews

Write Reversion: Businessmen can write reversion

**C.DB Handler**

Read Date: Database reading data

Store Date: Database storing data

**D.Community System**

Read Date: System can read data

Search Date: System can search data

Edit Date: System can edit data

View Date: System can view data

**5.3 Sequence Diagram**

**A.View**

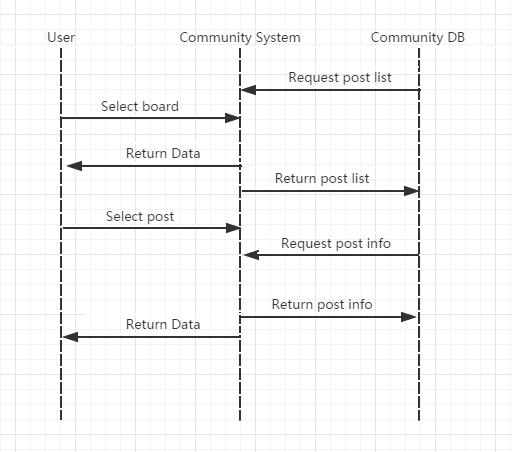
****

Figure 5.2 View Sequence Diagram for community system

**B.Write**

**By Account**

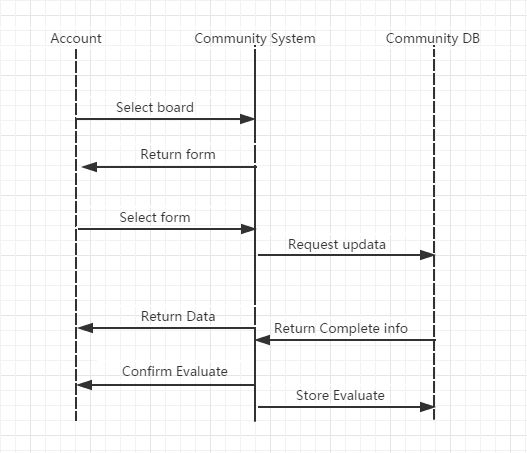
****

Figure 5.3 Write Sequence Diagram for community system by account

**By Business**

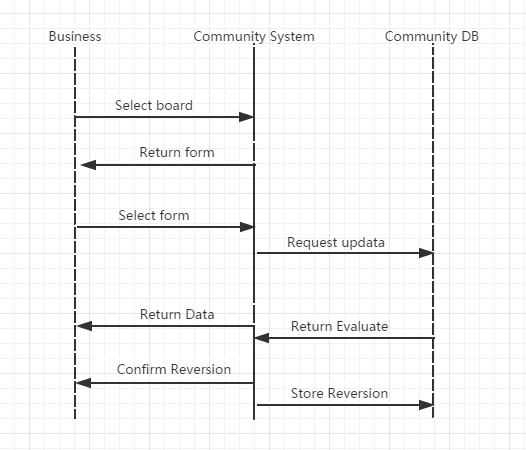
****

Figure 5.4 View Sequence Diagram for community system by business

**C.Edit**

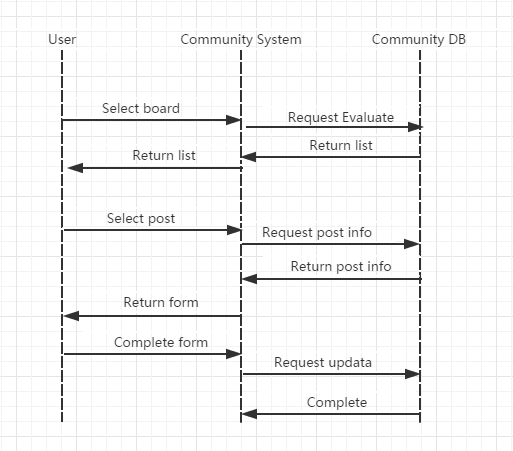
****

Figure 5.5 Edit Sequence Diagram for community system

**5.4 State Diagram**

**A.View**

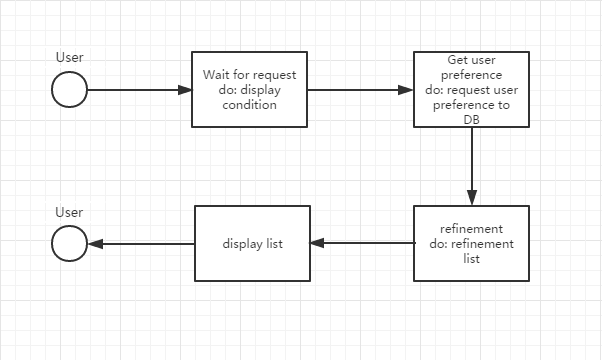
****

Figure 5.6 View state Diagram for community system

**B.Write**

**By Account**

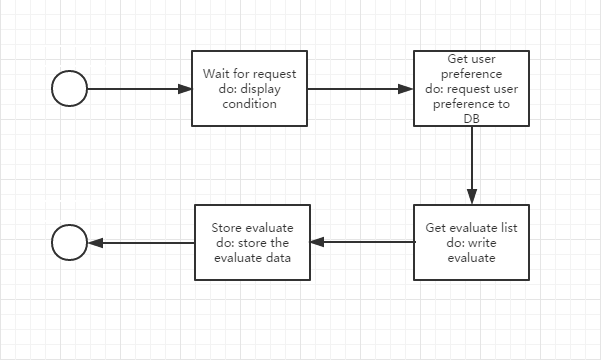
****

Figure 5.7 Write state Diagram for community system by account

**By Business**

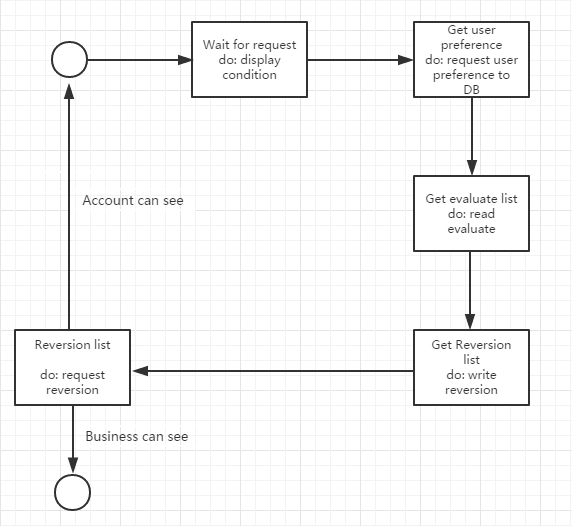
****

Figure 5.8 Write state Diagram for community system by business

**6.Search System**

**6.1 Objection**

This chapter says that after the user enters the address, the system can match the nearby merchants for the customers and show them the merchandise owned by the merchants.

**6.2 Search Diagram**

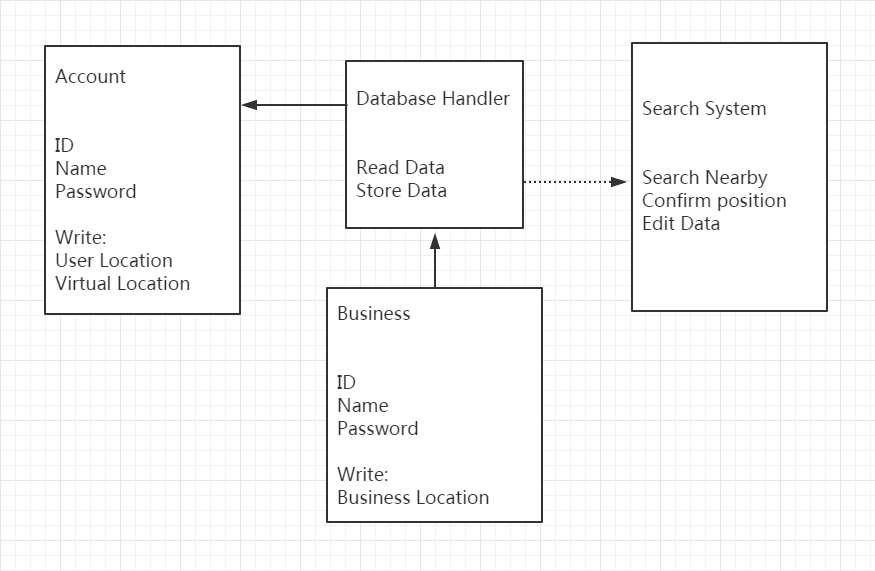
****

Figure 6.1 Diagram for search system

**A.Account**

Account ID: Customer account number

Account Name: Customer Network Name

Account Password: Customer account password

Write User Location: Address filled by customer

Write Virtual Location: Virtual address filled by customer

**B.Business**

Business ID: Business account number

Business Name: Shop Name of Merchant

Business Password: Business account password

Write Business Location: The address of the shop

**C.DB Handler**

Read Date: Database reading data

Store Date: Database storing data

**D.Search System**

Search Nearby: Search for nearby businesses

Confirm position: Identify customer and business location

Edit Date: System can edit data

**6.3 Sequence Diagram**

**A.View**

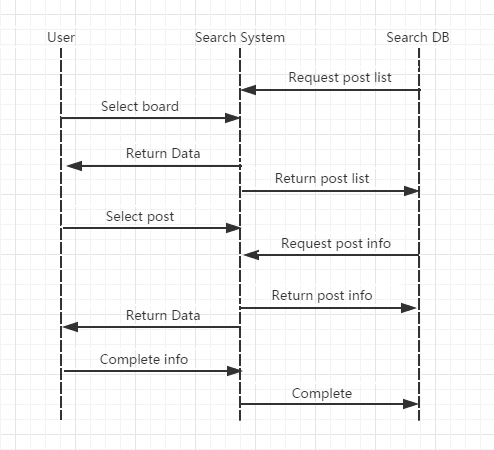
****

Figure 6.2 View Sequence Diagram for search system

**B.Write**

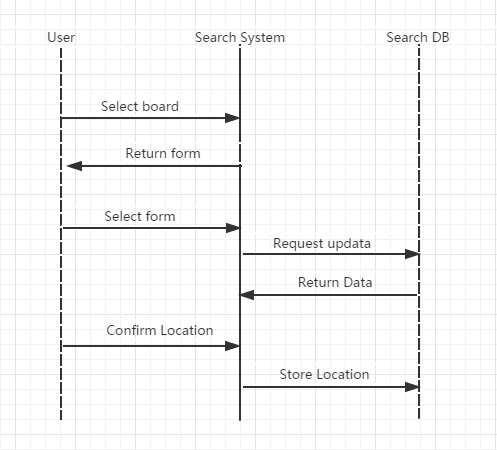
****

Figure 6.3 Write Sequence Diagram for search system

**C.Edit**

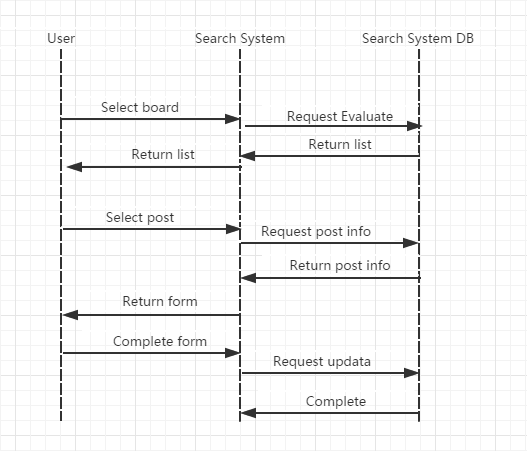
****

Figure 6.4 Edit Sequence Diagram for search system

**6.4 State Diagram**

**A.View**

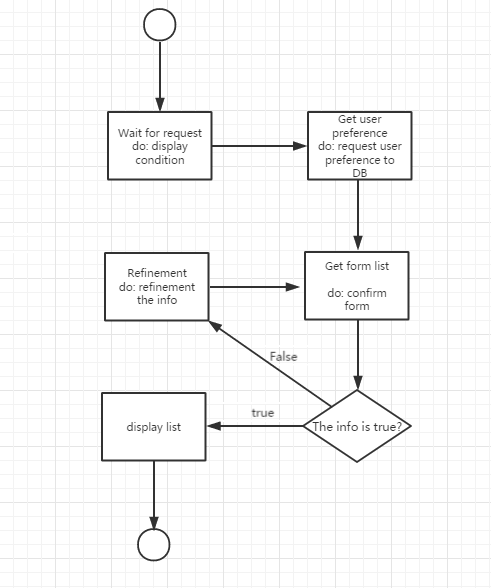
****

Figure 6.5 View state Diagram for search system

**B.Write**

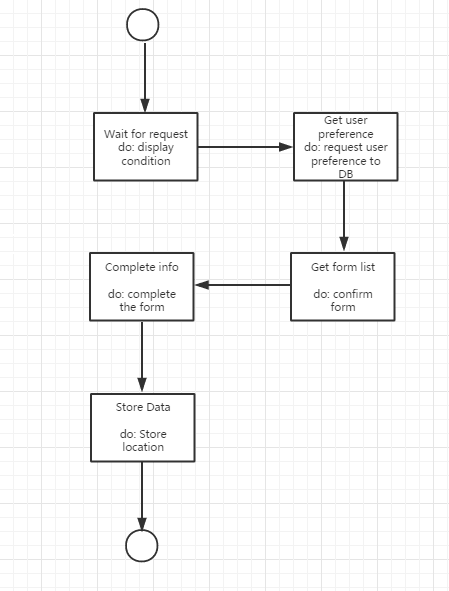
****

Figure 6.6 Write state Diagram for search system

**7.Selling System**

**7.1 Objection**

This chapter is about how quickly a merchant receives information from a customer when he or she decides to buy food, including the name, telephone number, address, etc. he or she fills in, and the information about the goods he or she wants to buy.

**7.2 Selling Diagram**

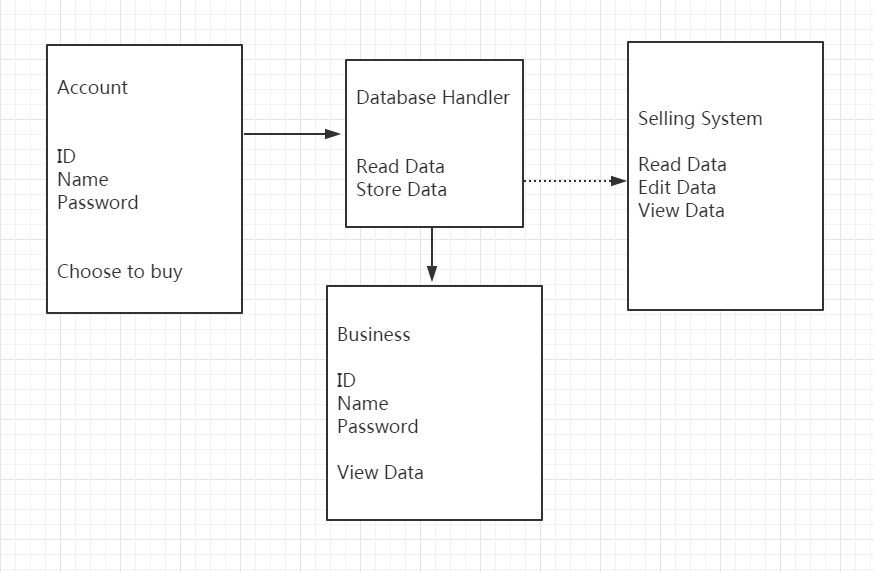
****

Figure 7.1 Diagram for selling system

**A.Account**

Account ID: Customer account number

Account Name: Customer Network Name

Account Password: Customer account password

Choose to buy: Customer can choose to buy the foods

**B.Business**

Business ID: Business account number

Business Name: Shop Name of Merchant

Business Password: Business account password

View Data: Business can view the information about account, including the name, the telephone, the address and what foods had to buy.

**C.DB Handler**

Read Date: Database reading data

Store Date: Database storing data

**D.Selling System**

Read Data: System can read data

Edit Data: System can edit data

View Data: System can edit data

**7.3 Sequence Diagram**

**A.View**

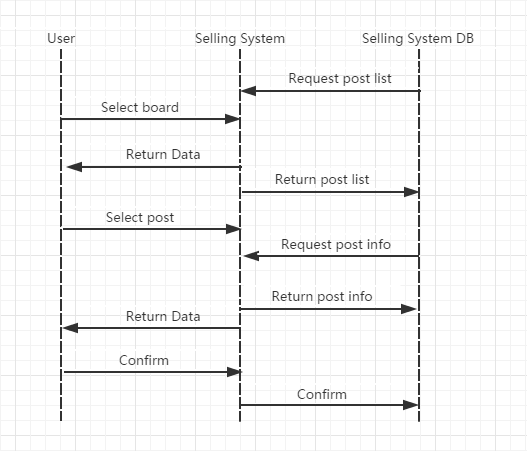
****

Figure 7.2 View Sequence Diagram for selling system

**B.Write**

**By account**

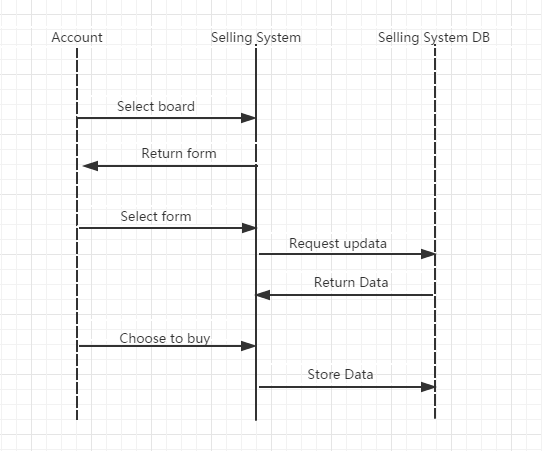
****

Figure 7.3 Write Sequence Diagram for selling system by account

**By Business**

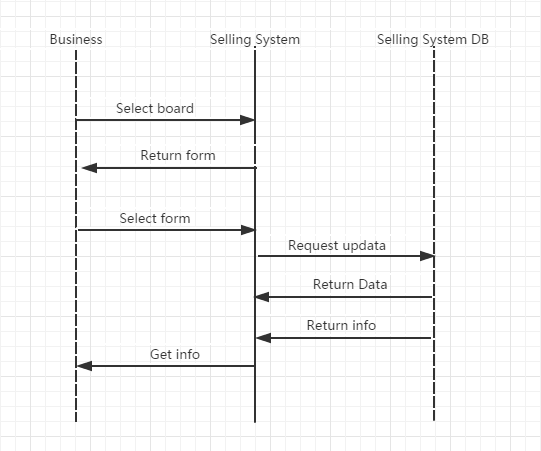
****

Figure 7.4 Write Sequence Diagram for selling system by business

**C.Edit**

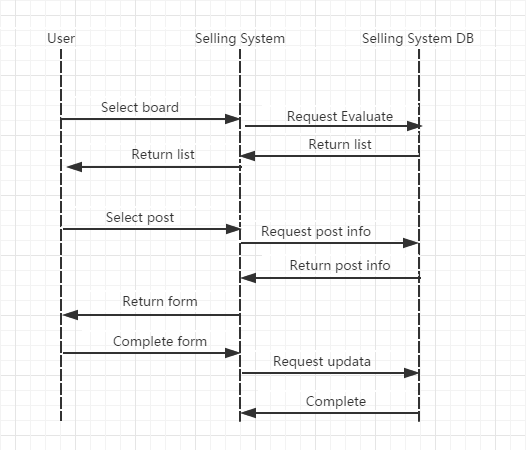
****

Figure 7.5 Edit Sequence Diagram for selling system

**7.4 State Diagram**

**A.View**

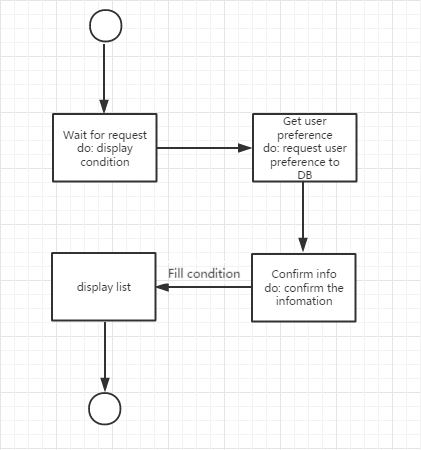
****

Figure 7.6 View state Diagram for selling system

**B.Write**

**By account**

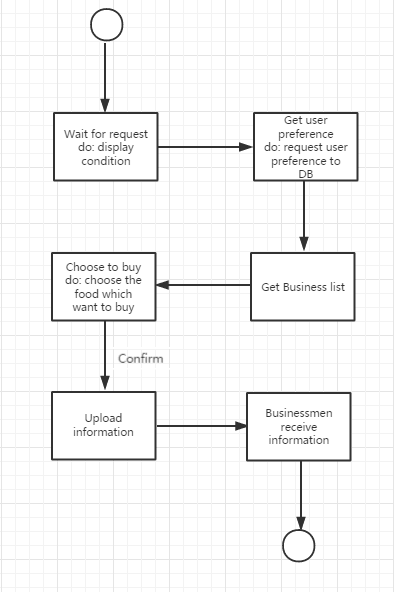
****

Figure 7.7 Write state Diagram for selling system by account

**By Business**

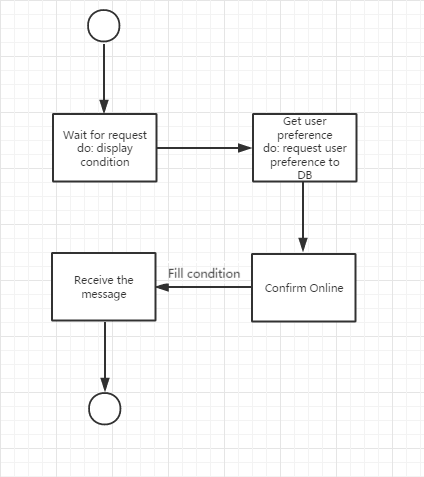
****

Figure 7.8 Write state Diagram for selling system by business

The information received includes the customer's name, location, telephone number, food demand, quantity of demand, etc.

**8. Purchase System**

**8.1 Objectives**

사용자가 선택한 음식을 판매자로부터 구매할 수 있도록 도와주는 시스템의 설계를 설명한다. Purchase System의 사용자는 Purchasing User에 해당되며, Purchasing User와 가까운 거리에 있는 Selling User들의 음식들 중 선택한 음식에 대해 결제를 진행할 수 있도록 하는 시스템의 설계를 Class Diagram, Sequence Diagram, State Diagram을 통하여 상호작용을 표현하고 설명한다.

**8.2 Class Diagram**

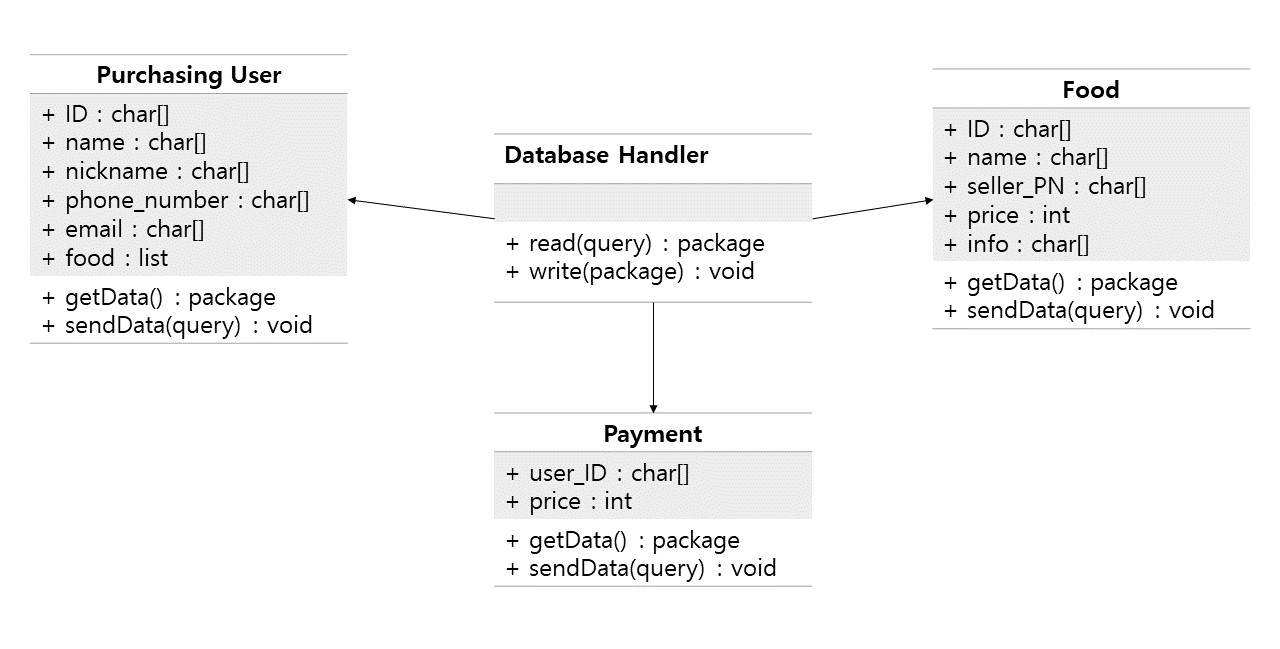


Figure 8.1 Purchasing system class diagram

1. **Database Handler**

**A.1 Attributes**

해당 사항 없음

**A.2 Methods**

+ package read(query) : 데이터베이스에서 원하는 데이터를 읽어온다.

+ void write(package) : 데이터베이스에 데이터를 저장한다.

1. **Purchasing User**

**B.1 Attributes**

+ ID : char[] : 구매자 아이디

+ name : char[] : 구매자 이름

+ nickname : char[] : 구매자 닉네임

+ phone\_number : char[] : 구매자 전화번호

+ email : char[] : 구매자 이메일

+ food : list : 구매자가 구매했던 음식 목록

**B.2 Methods**

+ package getData() : 데이터베이스로부터 데이터를 받는다.

+ void sendData(query) : 데이터베이스로 데이터를 보낸다.

1. **Food**

**C.1 Attributes**

+ ID : char[] : 판매자 아이디

+ name : char[] : 판매자 이름

+ seller\_PN : char[] : 판매자 전화번호

+ price : int : 해당 음식의 가격

+ info : char[] : 해당 음식 상세정보

**C.2 Methods**

+ package getData() : 데이터베이스로부터 데이터를 받는다.

+ void sendData(query) : 데이터베이스로 데이터를 보낸다.

1. **Payment**

**D.1 Attributes**

+ user\_ID : char[] : 사용자 아이디

+ price : int : 음식 가격

**D.2 Methods**

+ package getData() : 데이터베이스로부터 데이터를 받는다.

+ void sendData(query) : 데이터베이스로 데이터를 보낸다.

**8.3 Sequence Diagram**

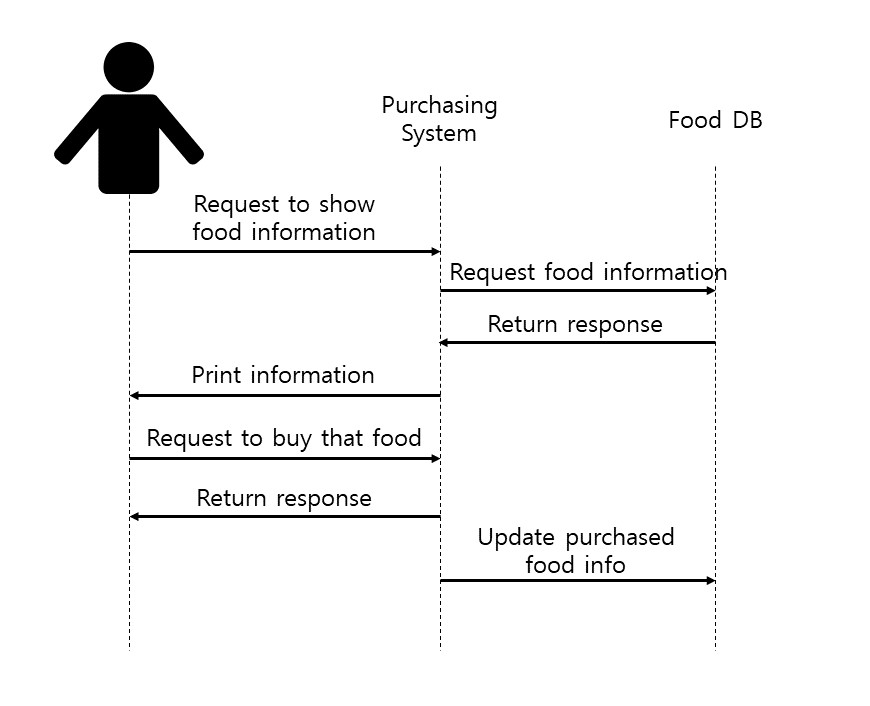


Figure 8.2 Purchasing system sequence diagram

**8.4 State Diagram**

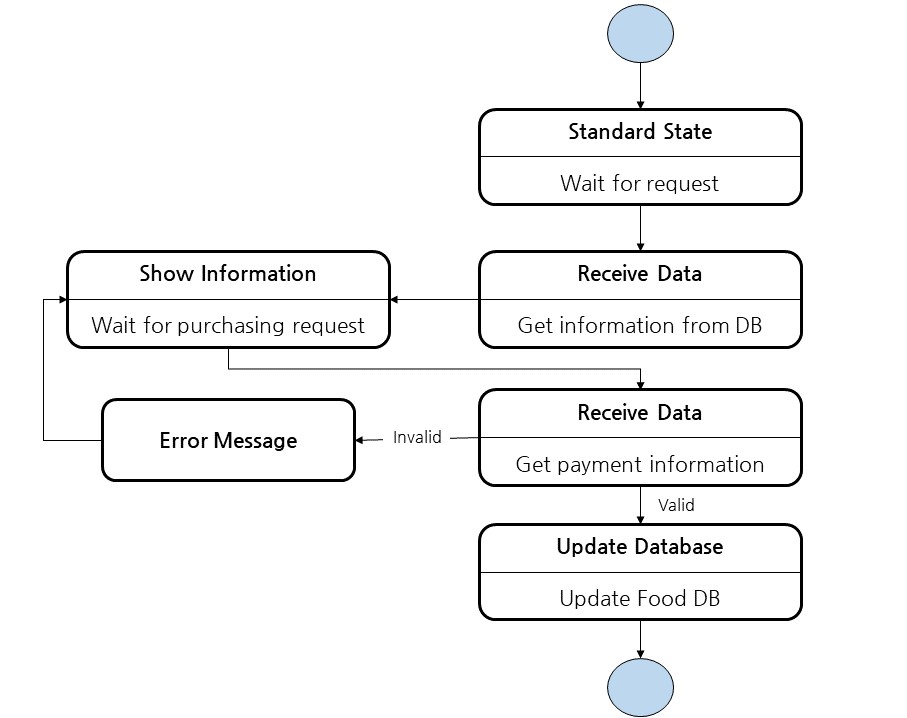


Figure 8.3 Purchasing system state diagram

**9. Protocol Design**

**9.1. Objectives**

Protocol Design에서는 subsystem들이 상호작용하는 프로토콜에 대해 서술한다. 프로토콜의 기본 형식은 JSON을 기본으로 하며 통신하는 메시지의 형식과 용도, 의미를 설명한다.

**9.2. JSON**



Figure 9.1 JSON Logo

JSON은 JavaScript Object Notation의 약어로써 XML과 더불어 대표적인 데이터 교환 방식이다. 자바스크립트에 기반하여 만들어진 데이터 표현형식이지만 프로그래밍언어나 플랫폼에 독립적인 특성을 갖고 있어, 다양한 프로그래밍 언어에서 JSON형식의 데이터를 생성하고 구문 분석하고 활용할 수 있다. 또한 기능이 적어 파싱(Parsing)이 빠르다는 장점을 가지고 있다. 자료의 종류에 큰 제한이 없으며, 컴퓨터 프로그램의 변수 값을 표현하는 데 적합하다. Attribute와 value의 쌍으로 표현되며, 거의 대부분의 자료형을 사용할 수 있다. JSON 의 공식 인터넷 미디어 유형은 응용 프로그램 이며 JSON 파일이름에는 .json 확장자를 사용한다.

**9.3. Protocol Description**

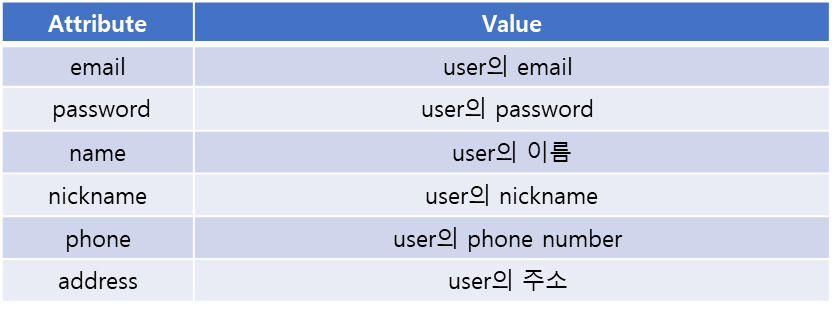
1. **Overview**

본 항목에서는 ‘for single life’에서 사용되는 메시지에 대해 JSON에 대응하는 Attribute와 Value로 작성하여 나타낸다. Server와 Client 사이에서 전송되는 메시지를 HTTP(HyperText Transfer Protocol)에서는 request와 response로 구분한다. request는 client에서 server에 전송하는 메시지, response는 서버가 client의 request에 반응하는 메시지를 나타낸다.

1. **Sign Up Protocol**

**B.1 Request**

**Table 1. Sign Up Request Protocol**



**B.2 Response**

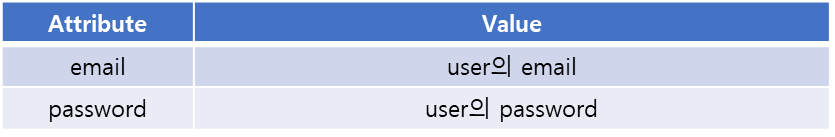
**Table 2. Sign Up Response Protocol**



1. **Login Protocol**

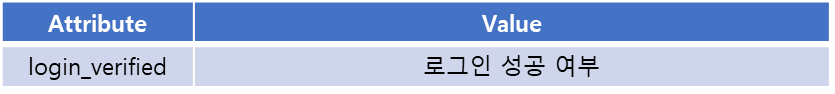
**C.1 Request**

**Table 3. Login Request Protocol**



**C.2 Response**

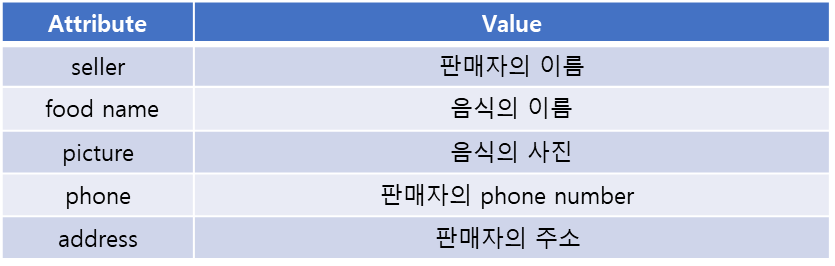
**Table 4. Login Response Protocol**



1. **food Sales registration / Edit Protocol**

**D.1 Request**

**Table 5. food Sales registration / Edit Request Protocol**



**D.2 Response**

**Table 6. food Sales registration / Edit Response Protocol**



1. **food recommendation Protocol**

**E.1 Request**

**Table 7. food recommendation Request Protocol**



**E.2 Response**

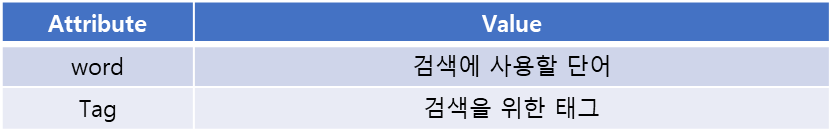
**Table 8. food recommendation Response Protocol**



1. **Search food Protocol**

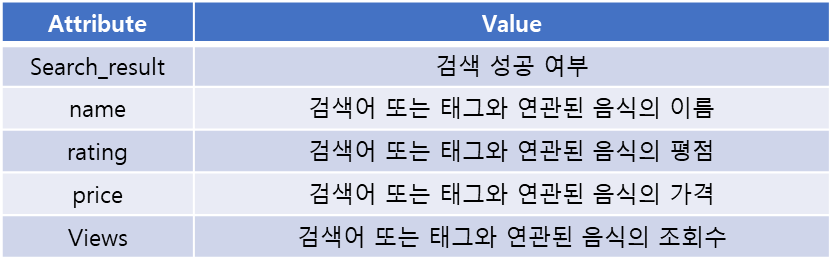
**F.1 Request**

**Table 9. Search food Request Protocol**



**F.2 Response**

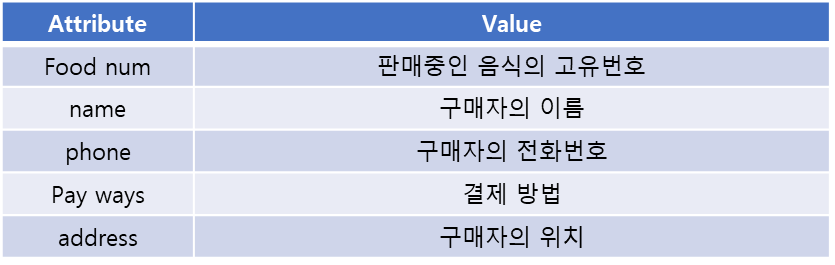
**Table 10. Search food Response Protocol**



1. **Purchase Protocol**

**G.1 Request**

**Table 11. Purchase Request Protocol**



**G.2 Response**

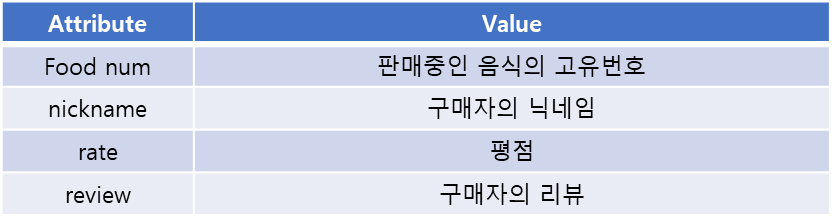
**Table 12. Purchase Response Protocol**



1. **Reviews and ratings Protocol**

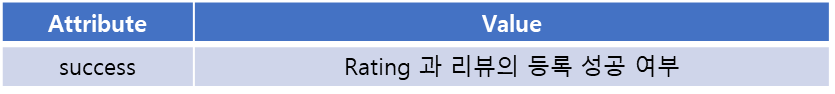
**H.1 Request**

**Table 13. Reviews and ratings Request Protocol**



**H.2 Response**

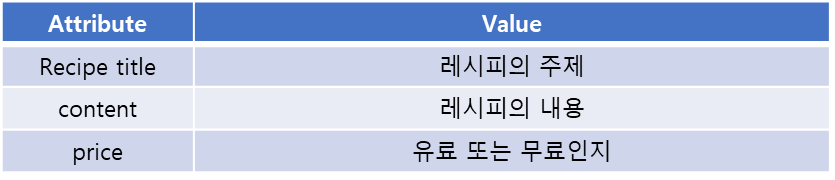
**Table 14. Reviews and ratings Response Protocol**



1. **Recipe post / edit Protocol**

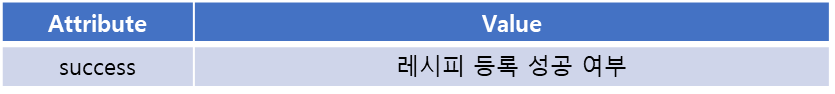
**I.1 Request**

**Table 15. Recipe post / edit Request Protocol**



**I.2 Response**

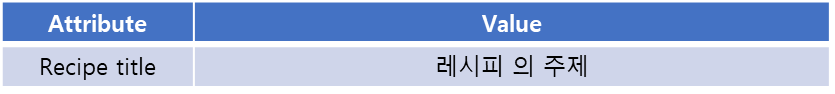
**Table 16. Recipe post / edit Response Protocol**



1. **Recipe confirmation Protocol**

**J.1 Request**

**Table 17. Recipe confirmation Request Protocol**



**J.2 Response**

**Table 18. Recipe confirmation Response Protocol**



1. **Community post / edit Protocol**

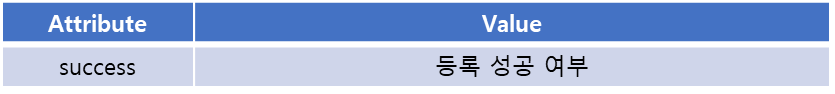
**K.1 Request**

**Table 19. Community post / edit Request Protocol**



**K.2 Response**

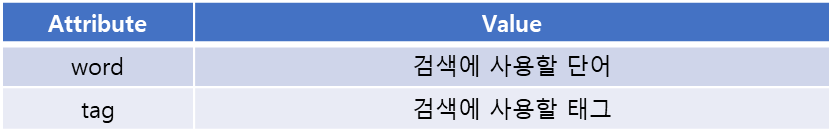
**Table 20 Community post / edit Response Protocol**



1. **Community Search Protocol**

**L.1 Request**

**Table 21. Community Search Request Protocol**



**L.2 Response**

**Table 22 Community Search Response Protocol**



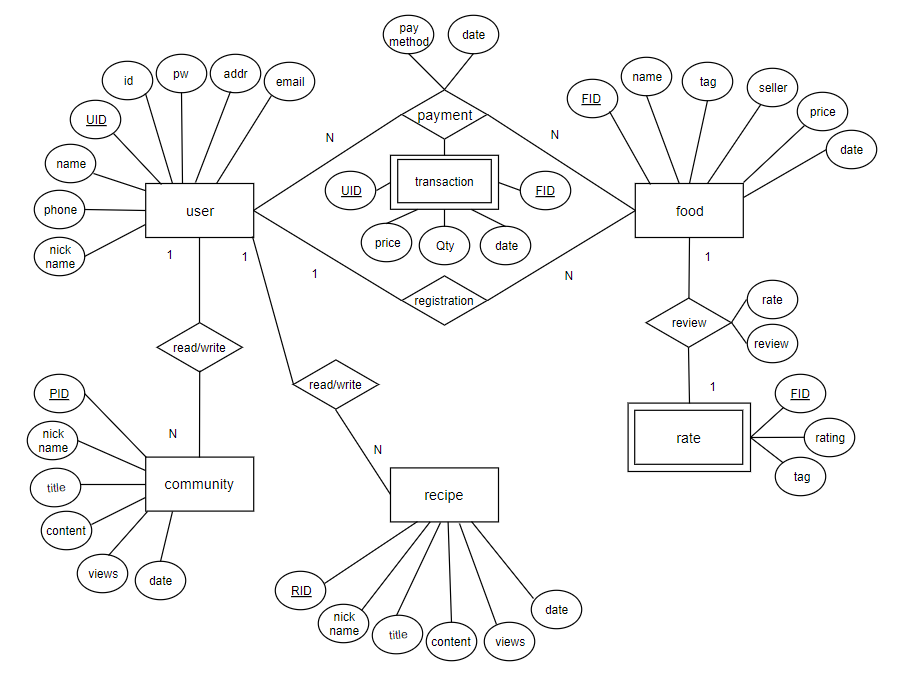
**10. Database Design**

**10.1. Objectives**

이번 장에서는 이전 명세서에서 기술했던 데이터베이스 요구사항에서 여러 수정사항을 적용한 뒤 보다 구체적으로 데이터베이스를 디자인하였 다. 데이터베이스 요구사항을 바탕으로 ER Diagram (Entity Relationship Diagram) 을 작성한 후, 그를 바탕으로 Relational Schema를 작성한 뒤 Normalization 과정을 거친 뒤 SQL DDL을 작성하는 것으로 마무리했다

**10.2. ER Diagram**

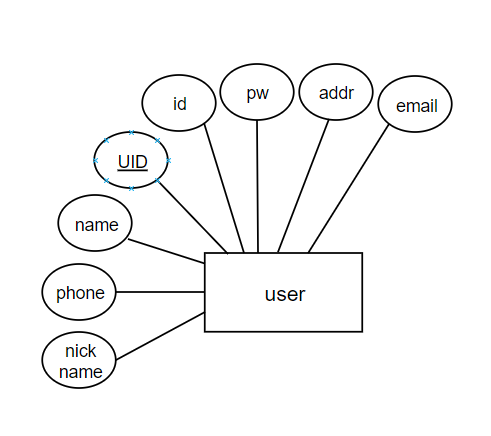
개체는 독립된 주체 하나를 나타낸다. 각 각체 사이의 관계는 다이아 몬드로, 주체는 사각형으로 표현된다. 관계, 또는 주체는 속성을 가질 수 있으며, 이 특성들은 실선으로 연결된 타원형으로 표현한다. 관계는 둘 이상의 객체 간 연관 관계를 표현하고, 모든 주체는 고유하게 식별되는 속성을 가지고 있는데 최소한의 고유 식별 속성을 그 주체의 기 본 키(Primary Key)라고 한다.



**Figure 10.1 ER diagram**

1. **Entity**

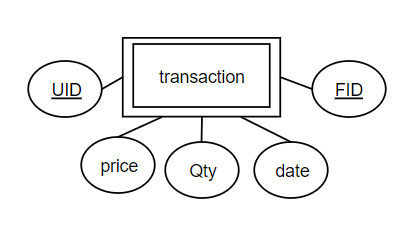
**A.1 USER**



**Figure 10.2 user entity**

User는 모든 사용자의 정보를 나타낸다. UID(user id), name, nickname, id, pw, email, addr(주소), phone(전화번호)의 속성을 가지고 있으며 UID는 Primary key이다.

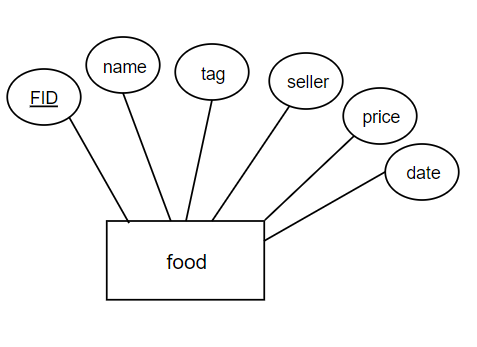
**A.2 TRANSACTION**



**Figure 10.3 transaction entity**

Transaction은 사용자의 거래내역을 나타낸다. Weak entity이며 UID(user id), price, Qty(수량), date(주문일), FID(food id)의 속성을 가지고 있으며, weak entity이기 때문에 primary key를 갖지 않는 대신 UID 과 FID 둘의 foreign key를 갖는다.

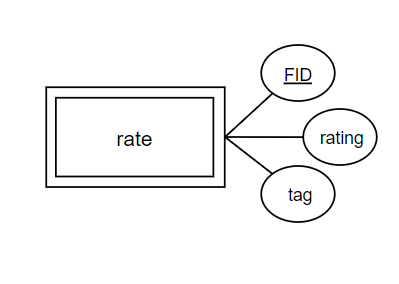
**A.3 food**



**Figure 10.4 food entity**

Food는 사용자가 등록한 그리고 거래 가능한 음식을 나타낸다. FID(food id), name(음식의 이름), tag, seller, price, date(등록된 날짜)의 속성을 갖고 있으며 primary key로 FID를 사용한다

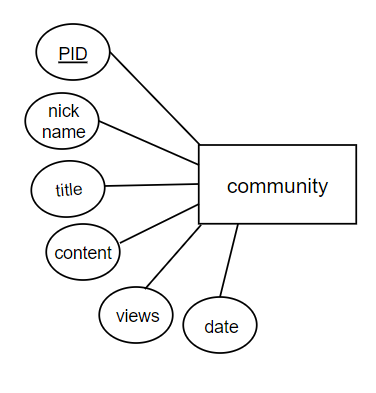
**A.4 rate**



**Figure 10.5 rate entity**

Rate는 등록된 음식에 대한 리뷰 또는 평가를 통한 rating을 나타낸다. FID, rating, tag의 속성을 가지고 있으며 weak entity 이기 때문에 primary key를 가지는 대신 foreign key 인 FID(food id )를 가진다.

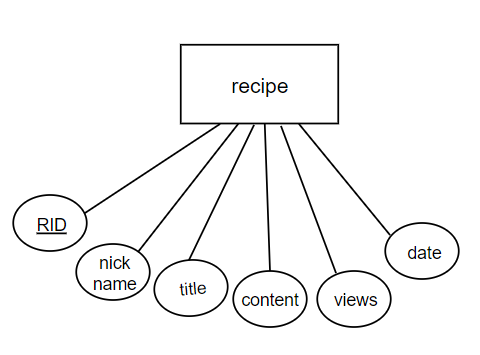
**A.5 Community**



**Figure 10.6 community entity**

Community는 사용자들이 작성한 글이나 정보를 나타낸다. PID(post id), nickname, title, content, views, date의 속성을 가지고 있으며, primary key로 PID를 사용한다.

**A.6 Recipe**

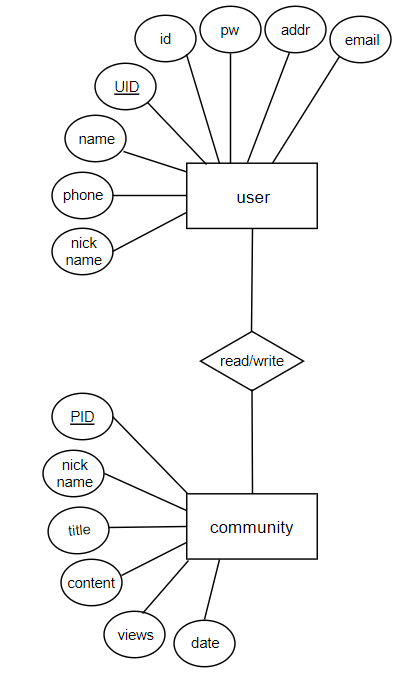


**Figure 10.7 recipe entity**

Recipe는 사용자가 등록하고 구입을 할 수 있는 레시피를 나타낸다. RID(recipe id), nickname, title, content, views, date(등록된 날짜)의 속성을 가지고 있으며, primary key로 RID를 가진다.

1. **Relationship**

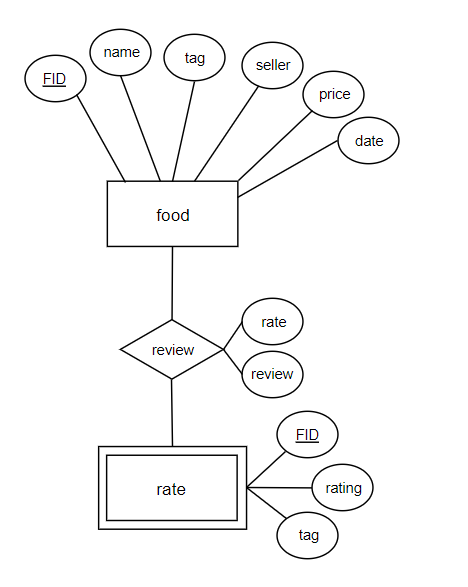
**B.1 community read/write**



**Figure 10.8 community read/write relationship diagram**

community read/write는 사용자(user)가 community과 송수신을 하는 관계로 한 명의 user는 여러 개의 글을 작성을 할 수 있고, 여러 개의 글을 확인 할 수 있으며, 모든 등록된 글은 특정한 사용자 한 명이 작성을 해야 한다.

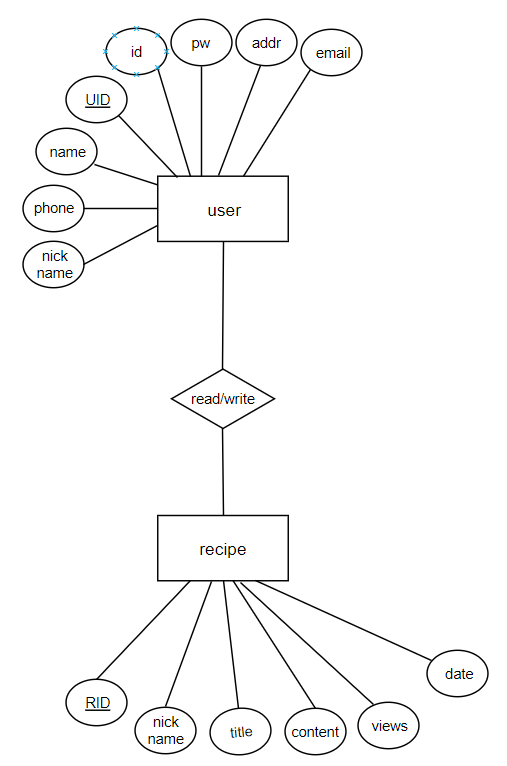
**B.2 food review**



**Figure 10.9 food review relationship diagram**

Food review은 사용자가 food에 대해 review을 작성하는 관계로. 하나의 음식에는 여러 개의 리뷰가 등록 될 수 있으며 하나의 리뷰는 한 개의 음식에 대한 리뷰이여야 한다.

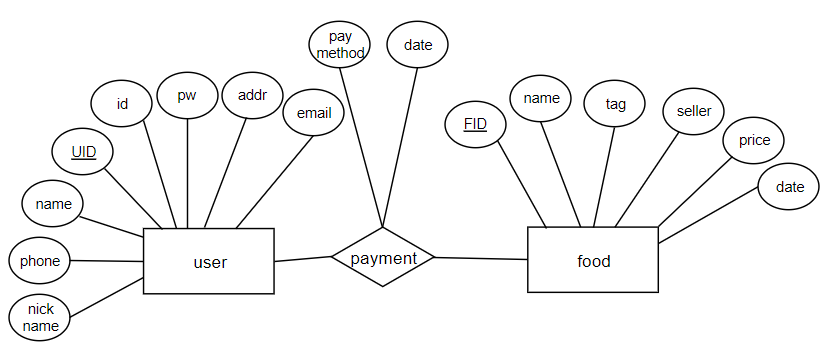
**B.3 recipe read/write**



**Figure 10.10 recipe read/write relationship diagram**

Recipe read/write는 사용자(user)가 레시피(recipe)를 이용하는 관계이다. 한 명의 사용자는 여러 개의 레시피를 작성할 수 있으며, 여러 개의 레시피를 확인 할 수 있다. 모든 레시피에는 작성한 유저가 한 명이 있어야한다.

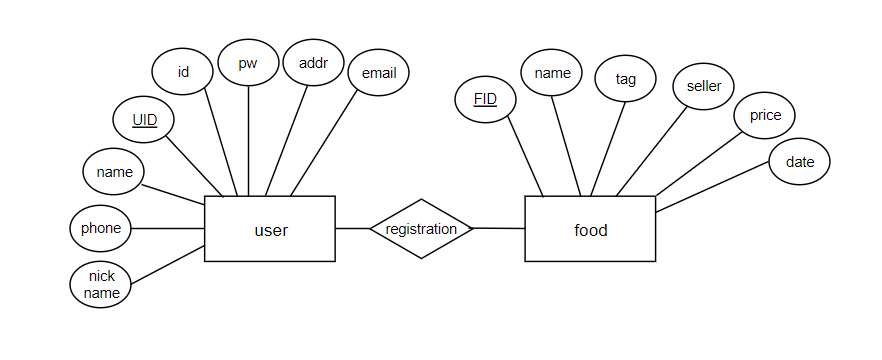
**B.4 payment**



**Figure 10.11 payment relationship diagram**

Payment는 사용자가 food를 구입하는 관계이다. 1명의 사용자는 여러 개의 음식을 구입할 수 있으며, 한 개의 음식은 여러 명의 사용자한테 판매 될 수 있다.

**B.5 Food Registration**



**Figure 10.12 food registration relationship diagram**

Food registration은 사용자가 자신이 판매하고자 하는 음식을 등록하는 관계이다. 한 명의 사용자의 여러 개의 음식을 등록 할 수 있으며, 모든 등록된 음식은 하명의 판매자(사용자)가 있다.

**B.1 community read/write**

**10.3.** **Relational Schema**

1. **User**

**Table 23 user Relational Schema**



Primary Key (PK) : UID

Foreign Key (FK) : none

Functional Dependency (FD) : {UID} -> {UID, id, pw, name, nickname, email, addr, phone}

Description: Instructor 객체에 관한 테이블이다. UID, id, pw, name, email, addr, phone 속성에 NULL을 허용하지 않는다.

1. **Food**

**Table 24 food relational schema**



Primary Key (PK) : FID

Foreign Key (FK) : none

Functional Dependency (FD) : {FID} -> {FID, name, tag, seller, price, date}

Description: Instructor 객체에 관한 테이블이다. FID, name, seller, price, date 속성에 NULL을 허용하지 않는다.

1. **Transaction**

**Table 25 transaction relational schema**



Primary Key (PK) : FID, UID

Foreign Key (FK) : FID, UID

Functional Dependency (FD) : {FID} -> {FID, price, Qty, date}

{UID} -> {UID,}

Description: Instructor 객체에 관한 테이블이다. FID, UID, price, date, Qty 속성에 NULL을 허용하지 않는다. 또한 FID는 food의 FID를 참조하고 UID는 user의 UID를 참조한다.

1. **Rate**

**Table 26 rate relational schema**



Primary Key (PK) : FID

Foreign Key (FK) : FID

Functional Dependency (FD) : {FID} -> {FID, rate, review, tag, rating}

Description: Instructor 객체에 관한 테이블이다. FID, name, rate, rating 속성에 NULL을 허용하지 않는다. FID는 food의 ID속성을 참조한다.

1. **Community**

**Table 27 community rotational schema**



Primary Key (PK) : PID

Foreign Key (FK) : NONE

Functional Dependency (FD) : {PID} -> {PID, nickname, title, content, views, date}

Description: Instructor 객체에 관한 테이블이다. PID, nickname, title, content, date 속성에 NULL을 허용하지 않는다.

1. **Recipe**

**Table 28 recipe rotational schema**



Primary Key (PK) : RID

Foreign Key (FK) : NONE

Functional Dependency (FD) : {RID} -> {RID, nickname, title, content, views, date}

Description: Instructor 객체에 관한 테이블이다. RID, nickname, title, content, date 속성에 NULL을 허용하지 않는다.

**10.4. Normalization**

데이터베이스 정규화란 데이터베이스의 설계를 재구성하는 테크닉이다. 데이터베이스 정규화의 목표는 이상이 있는 관계를 재구성, 작고 잘 조직된 관계를 생성하는 것에 있다. 일반적으로 정규화는 크고 제대로 조 직되지 않은 테이블과 관계들을 작고 잘 조직된 테이블과 관계들로 나누는 과정을 포함한다.

정규화:

1) 테이블 내용 변경 시 이상 현상 제거 – 테이블의 내용을 변경 (갱신, 삽입, 삭제)할 때, 의도치 않던 이상 현상이 발생할 수 있다. 이 이상 현상은 충분히 정규화되지 않은 테이블에서 발생하게 된 다.

2) 데이터베이스 구조 변경할 때 재 디자인 cost를 최소화하기 - 이는 데이터베이스와 연동된 프로그램에 주는 영향을 최소한으로 만들어 프로그램의 생명을 연장시킨다.

3) 유저에게 데이터를 더욱 의미 있게 받아들일 수 있게 하기 - 정규화된 테이블 간의 관계는 현실세계에서의 실제로 해당 객체 간의 관계를 반영한다.

4) 다양한 Query 지원 - 정규화된 테이블은 일반적인 Query에 적 합하다. 이는 어떤 정규화된 테이블에 대하여 세부사항이 예측되지 않은 query를 포함한 어떠한 query도 지원한다는 것이다.

관계형 데이터베이스 테이블이 제 3 정규화(3NF)가 되었으면 ‘정규 화되었다’ 라고 한다. 3NF 테이블은 대부분 삽입, 변경, 삭제에 이상이 없으며, 3NF 테이블의 대부분이 BCNF이다. 그렇기 때문에 본 프로젝트 개발에서도 BCNF까지 정규화를 수행했다.

**10.5. SQL DDL**

1. **User**

**Table 29 user SQL DDL**

|  |
| --- |
| CREATE TABLE user (  UID int NOT NULL,  id varchar(20) NOT NULL,  pw varchar(20) NOT NULL,  name varchar(10) NOT NULL,  email varchar(30) NOT NULL,  addr varchar(30), NOT NULL,  phone varchar(13), NOT, NULL  CONSTRAINT INSTRUCTOR\_P1 PRIMARY KEY(UID), |
| Primary Key가 되는 UID 속성은 NOT NULL로 설정했다. |

1. **Food**

**Table 30 food SQL DDL**

|  |
| --- |
| CREATE TABLE food (  FID int NOT NULL,  name varchar(20) NOT NULL,  seller varchar(20) NOT NULL,  price varchar(10) NOT NULL,  tag varchar(30),  date varchar(30),  CONSTRAINT INSTRUCTOR\_PK1 PRIMARY KEY(FID), |
| Primary Key가 되는 FID 속성은 NOT NULL로 설정했다. |

1. **Transaction**

**Table 31 transaction SQL DDL**

|  |
| --- |
| CREATE TABLE transaction (  FID int NOT NULL,  UID int NOT NULL,  Qty int NOT NULL,  price varchar(10) NOT NULL,  date varchar(30),  CONSTRAINT INSTRUCTOR\_PK1 PRIMARY KEY(FID, UID),  FOREIGN KEY(FID) REFERENCES food(FID) ON DELETE CASCADE ON UPDATE CASCADE);  FOREIGN KEY(UID) REFERENCES user(UID) ON DELETE CASCADE ON UPDATE CASCADE); |
| Primary Key가 되는 FID, UID 속성은 NOT NULL로 설정했다. |

1. **Rate**

**Table 32 rate SQL DDL**

|  |
| --- |
| CREATE TABLE rate (  FID int NOT NULL,  rate varchar(10) NOT NULL,  rating varchar(10) NOT NULL,  date varchar(30),  tag varchar(30),  CONSTRAINT INSTRUCTOR\_PK1 PRIMARY KEY(FID),  FOREIGN KEY(FID) REFERENCES food(FID) ON DELETE CASCADE ON UPDATE CASCADE); |
| Primary Key가 되는 FID 속성은 NOT NULL로 설정했다. |

1. **Community**

**Table 33 Community SQL DDL**

|  |
| --- |
| CREATE TABLE community (  PID int NOT NULL,  nickname varchar(10) NOT NULL,  title varchar(10) NOT NULL,  content varchar NOT NULL,  date varchar(30),  views varchar(30),  CONSTRAINT INSTRUCTOR\_PK1 PRIMARY KEY(PID), |
| Primary Key가 되는 PID 속성은 NOT NULL로 설정했다. |

1. **Recipe**

**Table 34 recipe SQL DDL**

|  |
| --- |
| CREATE TABLE recipe (  RID int NOT NULL,  nickname varchar(10) NOT NULL,  title varchar(10) NOT NULL,  content varchar NOT NULL,  date varchar(30),  views varchar(30),  CONSTRAINT INSTRUCTOR\_PK1 PRIMARY KEY(PID), |
| Primary Key가 되는 RID 속성은 NOT NULL로 설정했다. |

**11. Testing Plan**

**11.1 Objective**

시스템이 의도한 방향대로 실행되는지 확인하고 시스템 내부의 결함을 찾기 위해 Testing을 진행한다. 이러한 목적을 가진 Testing을 설계단계에서 미리 계획한다. 이 때 Testing Plan에서는 Testing Policy와 여러 Test Case에 대해서 기술한다.

**11.2 Testing Policy**

‘For Single Life’ 시스템의 개발 과정에서 Testing은 크게 3단계로 나누어 진행한다. Developing Testing, Release Testing, User Testing으로 나누어진다.

1. **Development Testing**

개발 과정에서 수행되는 Testing으로, 다음과 같은 세부 Testing으로 구성된다.

1. Component Testing

Component 단위로 개발한 후에 각각의 Component들이 정상적으로 작동하는지 확인하는 Testing이다.

1. Integrating Testing

Subsystem들을 점차적으로 통합하면서 제대로 작동하는지 확인하는 Testing이다.

1. System Testing

모든 Subsystem들을 합친 후에 시스템이 제대로 작동하는지 확인하는 Testing이다.

1. Acceptance Testing

사용자의 정보를 이용하여 시스템에서 사용자의 요구사항이 제대로 작동하는지 확인하는 Testing이다.

1. **Release Testing**

사용자에게 출시하기 전에 최종적으로 시스템을 테스트함으로써, Requirement Specification에 작성되었던 요구사항이 모두 반영되었는지 확인한다.

1. **User Testing**

사용자의 실제 환경에서 시스템이 정상적으로 동작하는지를 테스트한다.

**11.3 Test Case**

1. **User Management System**

**A.1 Sign up for User**

1. User : 회원가입 버튼을 눌러 회원가입을 시도한다
2. 시스템 동작 : 회원가입 양식 페이지로 이동한다.
3. User : 회원가입 양식을 채운 후에 양식을 제출한다.
4. 시스템 동작 : 모든 회원가입 양식이 채워졌는지 확인하고 User DB에 있는 ID와 중복 여부를 확인한다.

4-1) 회원가입 성공

시스템 알림 : “회원가입이 완료되었습니다.”

시스템 동작 : User DB에 회원 정보를 저장한다.

4-2) 회원가입 실패

시스템 알림 : “회원가입에 실패하였습니다.”

시스템 동작 : 회원가입 양식 페이지에 머무른다.

**A.2 Log in**

1. User : ID와 PW를 입력하여 로그인을 시도한다.
2. 시스템 동작 : User DB에 저장되어 있는 데이터와 입력한 User의 정보가 일치하는지 확인한다.

2-1) 로그인 성공

시스템 알림 : “로그인이 완료되었습니다.”

시스템 동작 : 시스템의 Main Page로 넘어간다.

2-2) 로그인 실패

시스템 알림 : “로그인에 실패하였습니다.”

시스템 동작 : 로그인 화면을 계속 유지한다.

1. **Community System**

**B.1 Community Post Write**

1. User : 글 작성을 하기 위해 Post 버튼을 클릭한다.
2. 시스템 동작 : 작성할 수 있는 페이지를 불러온다.
3. User : 작성 완료 후에 완료 버튼을 클릭한다.

3-1) Post Write 성공

시스템 알림 : “성공적으로 글이 등록되었습니다.”

시스템 동작 : 작성된 Post가 Post DB에 저장된다.

3-2) Post Write 취소

시스템 동작 : Community 글 목록 페이지로 돌아간다.

**B.2 Community Post Edit**

1. User : 수정하고 싶은 게시글에 들어간 후 Edit 버튼을 클릭한다.
2. 시스템 동작 : 해당 게시물을 수정할 수 있는 페이지를 불러온다.
3. User : 글을 수정한 후에 완료 버튼을 클릭한다.

3-1) Post Edit 성공

시스템 알림 : “성공적으로 글이 수정되었습니다.”

시스템 동작 : 수정된 데이터가 Post DB의 기존 데이터에서 Update된다.

3-2) Post Edit 취소

시스템 동작 : 수정하기 전의 게시글 페이지로 돌아간다.

1. **Search System**

**C.1 Near Food Search**

1. User : 가까운 판매자의 음식을 검색하기 위해 Purchase 버튼을 클릭한다.
2. 시스템 동작 : User와 가까운 판매자의 음식을 Food DB에서 확인하고, 적합한 음식들의 리스트를 가져온다.

2-1) 가까운 판매자의 음식이 존재할 경우

시스템 알림 : “검색이 완료되었습니다.”

시스템 동작 : 검색 결과를 화면에 출력한다.

2-2) 가까운 판매자의 음식이 존재하지 않을 경우

시스템 알림 : “해당 거리 안에 있는 판매되고 있는 음식이 없습니다.”

시스템 동작 : 더 넓은 범위로 재검색 또는 검색 취소의 옵션을 제공한다.

1. **Selling System**

**D.1 Food Upload**

1. User : Food의 상세정보를 작성한 후에 Upload 버튼을 클릭한다.

1-1)Food Upload 성공

시스템 알림 : “성공적으로 음식이 업로드 되었습니다.”

시스템 동작 : 음식 상세정보가 Food DB에 저장된다.

1-2) Food Upload 실패

시스템 알림 : “음식 등록에 실패하였습니다.”

시스템 동작 : 시스템의 Main Page로 돌아간다.

**D.2 Notification of Purchasing**

1. User : 구매 User가 원하는 음식의 결제를 완료한다.
2. 시스템 동작 : User DB에 저장되어있는 구매 User의 개인정보를 가져온 후에 판매 User에게 구매 User에 대한 개인정보와 결제완료 메시지를 전송한다.
3. **Purchase System**

**E.1 Purchase Food**

1. User : 검색한 리스트에서 원하는 음식을 클릭한다.
2. 시스템 동작 : 해당 음식의 상세정보 페이지를 출력한다.
3. User : 해당 음식의 상세정보 페이지에서 구매 버튼을 클릭한다.
4. 시스템 동작 : 음식의 결제 금액을 포함하고 있는 결제 페이지를 출력한다.

4-1) Purchase 성공

시스템 알림 : “결제가 완료되었습니다.”

시스템 동작 : User DB에 구매 정보를 저장하고 결제 완료 정보를 페이지에 출력한다.

4-2) Purchase 취소

시스템 동작 : 해당 음식의 상세정보 페이지를 다시 불러온다.

**12. Development Plan**

**12.1 Objectives**

Development Plan에서는 개발 계획에 대해 서술한다. 시스템의 실제 개발 과정을 Gantt Chart를 이용하여 서술한다.

**12.2 Gantt Chart**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 수행 내역 | 개발 일정 | | | | | | | | | | | | | | |
| 1주 | 2주 | 3주 | 4주 | 5주 | 6주 | 7주 | 8주 | 9주 | 10주 | 11주 | 12주 | 13주 | 14주 | 15주 | |
| 1 | Idea |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| 2 | Feasibility study |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| 3 | Requirement Analysis |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| 4 | Design Study |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| 5 | Design |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| 6 | Sub-system Develop |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| 7 | Sub-system Testing & Integration |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| 8 | Integration Testing |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| 9 | Validation & Verification |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |

**Figure 12.1 Gantt Chart**

위의 Gantt Chart는 실제 개발 상황과 앞으로의 예상 개발 계획을 나타내고 있다. 현재 11주차 까지 진행을 하면서 주제 선정이나 개발을 하기 위한 Requirement Specification과 Design Specification의 문서 작성에 많은 시간이 소요되어 생각했던 것 보다 Develop을 시작하는 시간이 늦어지게 되었다.

하지만 시스템에 대한 여러 단계의 문서 작성을 통해 Requirement와 Design의 Specification을 명확히 함으로써 남은 기간 내에 개발을 할 때 최대한 지연되지 않을 수 있도록 노력했다. 또한 가까운 판매자와 구매자를 연결하는 기술을 구현하는 것에 어려움이 있을 것이라 생각해서 그 부분에 초점을 맞춰서 시스템을 개발할 예정이다.

**13. Reference**

Software Evolution Process, https://www.ukessays.com/essays/information-systems/software-evolution-process.php

나홀로 대학생 급증… 영양상태 적신호 http://www.hynews.ac.kr/news/articleView.html?idxno=527

1인가구 보고서 <https://choonsik.blogspot.com/2017/02/2017-1.html>

**14. Index**

**14.1 Figure Index**

Figure 2.1 UML 12

Figure 2.2 Class Diagram 13

Figure 2.3 Sequence Diagram 14

Figure 2.4 State Diagram 15

Figure 2.5 E-R Diagram 16

Figure 2.6 Atom Editor 17

Figure 2.7 Draw.io 18

Figure 2.8 Tensorflow 19

Figure 3.1 System Oranization Block Diagram 20

Figure 3.2 User Management System Architecture 21

Figure 3.3 Order Management System Architecture 21

Figure 3.4 Distance Calculating System Architecture 22

Figure 3.5 Purchase System Architecture 23

Figure 3.6 Package Diagram 24

Figure 3.7 Deployment Diagram 25

Figure 4.1 User Management System Class Diagram 26

Figure 4.2 User Management System Sequential Diagram-sign in 27

Figure 4.3 User Management System Sequential Diagram-login 28

Figure 4.4 User Management System State Diagram-sign in 28

Figure 4.5 User Management System State Diagram-login 29

Figure 5.1 Diagram for community system 30

Figure 5.2 View Sequence Diagram for community system 32

Figure 5.3 Write Sequence Diagram for community system by account 33

Figure 5.4 View Sequence Diagram for community system by business 34

Figure 5.5 Edit Sequence Diagram for community system 35

Figure 5.6 View state Diagram for community system 36

Figure 5.7 Write state Diagram for community system by account 37

Figure 5.8 Write state Diagram for community system by business 38

Figure 6.1 Diagram for search system 39

Figure 6.2 View Sequence Diagram for search system 41

Figure 6.3 Write Sequence Diagram for search system 42

Figure 6.4 Edit Sequence Diagram for search system 43

Figure 6.5 View state Diagram for search system 44

Figure 6.6 Write state Diagram for search system 45

Figure 7.1 Diagram for selling system 46

Figure 7.2 View Sequence Diagram for selling system 48

Figure 7.3 Write Sequence Diagram for selling system by account 49

Figure 7.4 Write Sequence Diagram for selling system by business 50

Figure 7.5 Edit Sequence Diagram for selling system 51

Figure 7.6 View state Diagram for selling system 52

Figure 7.7 Write state Diagram for selling system by account 53

Figure 7.8 Write state Diagram for selling system by business 54

Figure 8.1 Purchasing system class diagram 55

Figure 8.2 Purchasing system sequence diagram 57

Figure 8.3 Purchasing system state diagram 58

Figure 9.1 JSON Logo 59

Figure 10.1 ER diagram 66

Figure 10.2 user entity 67

Figure 10.3 transaction entity 68

Figure 10.4 food entity 68

Figure 10.5 rate entity 69

Figure 10.6 community entity 69

Figure 10.7 recipe entity 70

Figure 10.8 community read/write relationship diagram 71

Figure 10.9 food review relationship diagram 72

Figure 10.10 recipe read/write relationship diagram 73

Figure 10.11 payment relationship diagram 74

Figure 10.12 food registration relationship diagram 74

Figure 12.1 Gantt Chart 86

**14.2 Table Index**

Table 1. Sign Up Request Protocol 59

Table 2. Sign Up Response Protocol 60

Table 3. Login Request Protocol 60

Table 4. Login Response Protocol 60

Table 5. food Sales registration / Edit Request Protocol 61

Table 6. food Sales registration / Edit Response Protocol 61

Table 7. food recommendation Request Protocol 61

Table 8. food recommendation Response Protocol 61

Table 9. Search food Request Protocol 62

Table 10. Search food Response Protocol 62

Table 11. Purchase Request Protocol 63

Table 12. Purchase Response Protocol 63

Table 13. Reviews and ratings Request Protocol 63

Table 14. Reviews and ratings Response Protocol 63

Table 15. Recipe post / edit Request Protocol 64

Table 16. Recipe post / edit Response Protocol 64

Table 17. Recipe confirmation Request Protocol 64

Table 18. Recipe confirmation Response Protocol 64

Table 19. Community post / edit Request Protocol 65

Table 20 Community post / edit Response Protocol 65

Table 21. Community Search Request Protocol 65

Table 22 Community Search Response Protocol 65

Table 23 user Relational Schema 75

Table 24 food relational schema 75

Table 25 transaction relational schema 75

Table 26 rate relational schema 76

Table 27 community rotational schema 76

Table 28 recipe rotational schema 77

Table 29 user SQL DDL 78

Table 30 food SQL DDL 78

Table 31 transaction SQL DDL 79

Table 32 rate SQL DDL 79

Table 33 Community SQL DDL 80

Table 34 recipe SQL DDL 80