Design

Specification

목차

1. Preface

1.1 Objectives

Preface 에서는 본 문서의 대상 독자층을 정의하고, 문서의 전반적인 구조와 각 목차의 역할에 대하여 제시한다. 그리고 문서의 버전 관리 정책, 버전 변경 기록, 변경 사항들을 서술한다.

1.2 Readership

설계 명세서의 대상 독자는 시스템의 개발과 유지 보수에 참여하는 모든 구성원이다. 시스템을 개발하는 소프트웨어 엔지니어, 시스템을 설계하는 아키텍쳐 외에도 매니저, 고객, 외주 업체 등 개발에 참여하는 모든 구성원들이 대상 독자이다.

1.3 Document Structure

A. Preface

Preface 에서는 본 문서의 대상 독자층을 정의하고, 문서의 전반적인 구조와 각 목차의 역할에 대하여 제시한다. 그리고 문서의 버전 관리 정책, 버전 변경 기록, 변경 사항들을 서술한다.

B. Introduction

Introduction 에서는 시스템 설계에 사용한 모든 종류의 다이어그램과 툴에 대해 소개하고 설명한다.

C. System Architecture

System Architecture 에서는 목표 시스템에 대한 고수준의 개요를 보여주고, 시스템 기능의 전체적 분포를 보여준다. System 의 구조를 Block Diagram 으로 나타낸 뒤, 그것들의 관계와 실제 사용을 Package Diagram, Development Diagram 으로 표현한다. 각 서브 시스템의 modular decomposition 방식은 대제목 번호 4, 5, 6, 7 번에서 설명한다.

D. 유저 관리 시스템

실제 사용자가 회원 가입과 로그인 기능을 통해 시스템을 이용하는 도중에 발생하는 데이터를 처리하고 도식화하는 사용자 관리 시스템의 설계를 설명한다. 또한

사용자의 완성된 식단 관리 기능에 대한 설계를 설명한다. Class Diagram, Sequence Diagram, State Diagra 을 통해 유저 관리 시스템의 구조를 표현하고 설명한다.

E. 맞춤 식단 시스템

사용자로부터 필요한 정보를 입력받으면 그 정보에 맞춰서 사용자 맞춤 식단을 추천해 주는 시스템의 설계를 설명한다. Class Diagram, Sequence Diagram 과 State Diagram 을 통해 맞춤 식단 시스템의 구조와 사용자와의 상호 작용을 표현하고 설명한다.

F. 커뮤니티 시스템

사용자가 사용하는 커뮤니티 게시판과 이를 이용하는 도중에 발생하는 데이터 처리하고, 수정사항에 해당하는 데이터를 수정, 변경하며 도식화하는 커뮤니티 시스템의 설계를 설명한다. Class Diagram, Sequence Diagram, State Diagram 을 통해 커뮤니티 시스템의 구조를 표현하고 설명한다.

G. 재료 구매 시스템

재료 구매 서버는 사용자가 출력된 식단의 재료를 구매할 수 있게 하는 시스템이다. Class Diagram, Sequence Diagram, State Diagram 을 통해 재료 구매 시스템의 구조를 표현하고 설명한다.

H. Protocol Design

Protocol Design 에서는 서브시스템들이 상호작용하는 프로토콜에 대해 서술한다. 메세지는 JSON 을 통해서 전달되며, 각 메세지의 형식, 용도, 의미를 설명한다.

I. Database Design

Database Design 에서는 요구사항 명세서에 서술된 데이터베이스에 대한 요구사항을 기반으로 데이터베이스를 설계한다. 데이터베이스의 요구사항을 기반으로 ER Diagram 을 그리고, 그 것을 기초로 Relational Schema 를 작성한다. 그리고 Normalization 을 수행하여 Redundancy 와 Anomaly 를 제거한 후 마지막으로 SQL DDL 을 작성한다.

J. Testing Plan

Testing Plan 에서는 Testing Policy 와 Test Case 에 대해 설명한다. 시스템이 의도한 방향으로 실행이 되는지 확인하고, 또 시스템 내부의 결함을 찾을 수 있도록 testing 을 설계단계에 미리 계획하는 것이다.

K. Development Environment

Development Environment 에서는 개발자의 환경에 대해 설명한다. 사용한 프로그래밍 언어와 IDE 에 대해 서술한다.

L. Index

Index 에서는 문서의 인덱스를 정리한다. Alphabet order, picture, image, diagram 의 인덱스들이 포함된다.

1.4 Version of the Document

A. Version Format

버전 번호는 major number 와 minor number 로 이루어져 있으며, (major number).(minor number)의 형태로 표현한다. 문서의 버전은 0.1 부터 시작한다.

B. Version Management Policy

설계 명세서를 수정할 때 마다 버전을 업데이트한다. 다만 변경 간의 간격이 1 시간 이내일 때에는 버전 번호를 업데이트하지 않고, 하나의 업데이트로 간주한다. 이미 완성된 파트를 변경할 때에는 minor number 를 변경하며, 새로운 부분을 추가하거나 문서의 구성이 예전에 비해 괄목할 변화가 있을 경우 minor number 를 변경한다.

C. Version Update History

Version	Modified Date	Explanation
0.1	2018. 05. 11.	대제목과 소제목 완성 목차와 문서의 초안 작성 Preface 작성 Introduction 작성
1.0	2018. 05. 13.	System Architecture 작성
2.0	2018. 05. 15.	4, 5, 6, 7 에 해당하는 각 Subsystem 작성
3.0	2018. 05. 17.	Protocol Design 작성 Database Design 작성
4.0	2018. 05. 18.	Testing Plan 작성 Development Environment 작성
4.1	2018. 05. 19.	Subsystem 수정 Database Design 수정
5.0	2018. 05. 20.	Appendices 작성
5.1	2018. 05. 21.	Package Diagram, Development Diagram 수정

2. Introduction

2.1 Objectives

Introduction 에서는 시스템 설계에 사용한 모든 종류의 다이어그램과 툴에 대해 소개하고 설명한다.

2.2 Applied Diagrams

A. UML

통합모델링 언어, 즉 UML 은 1994 년 소프트웨어 방법론의 선구자인 Grady Booch, James Rumbaugh, Ivar Jacobson 에 의해서 연구되었고, 1997 년 객체관리그룹(OMG, Object Management Group)에서 여러 표기법을 통합하여 UML 을 발표하였다. 현재 UML 은 객체지향 시스템 개발 분야에서 가장 우수한 모델링 언어로 인식되고 있다.

그 이유는 UML 이 시스템 개발자가 구축하고자 하는 소프트웨어를 코딩하기에 앞서 표준화되고 이해되기 쉬운 방법으로 설계하여 다른 사람들과 효율적으로 의사소통할 수 있는 메커니즘을 제공하기 때문이다. UML 이 있기 전의 시스템 개발은 “운이 좋으면 성공하고, 그렇지 않으면 망한다”는 말로 흔히 표현했다. 시스템 개발 초창기에는 프로그래머들이 프로그래머들만의 이해와 논리로 시스템을 구축하였기 때문에, 그 결과물은 의뢰인의 입장에서 보았을 때 부족함이 많을 수밖에 없었다. 이는 프로그래머들이 대개 의뢰인의 요구사항에 대해 면밀하게 분석하지 않았기 때문이다.

그러나 경쟁이 심화되는 오늘날에는 치밀한 사고와 기획만이 성공하는 지름길이다. 따라서 의뢰인은 개발팀이 무엇을 해야 하는지 이해해야 하고, 개발팀이 의뢰인의 요구를 제대로 이해하지 못했을 경우에는 변경사항을 지적해 주어야 한다. 또한

시스템 개발은 개인이 아닌 팀으로 운영하는 활동이기 때문에, 개발팀의 각 멤버는 자신이 개발 작업의 어느 분야에 적합한지를 파악해야 한다. 시스템 개발에 참여하는 분석가, 의뢰인, 프로그래머, 그 외의 모든 사람들이 이해하고 동의할 수 있는 방법으로 설계 과정을 조직화해야 한다.

UML 이 바로 이러한 조직화 수단을 제공해준다. 다시 말해 의뢰인의 요구사항에 맞는 소프트웨어 설계의 필요성은 바로 설계 표기에 대한 필요성을 낳았다. 집을 짓기 위한 설계도가 표준 기법으로 작성되어야 건축자가 충분히 이해하면서 집을 지을 수 있듯이, 시스템 개발에서 분석가, 의뢰인, 개발자가 서로 이해하며, 표준으로 받아들일 수 있는 시스템 설계를 위해 마련한 표기법이 바로 UML 이다.

UML 은 구축할 시스템의 유형에 관계없이 적용될 수 있다. 또한 개발 방법론에 관계없이 적용될 수 있다. 프로그래밍 언어에 관계없이 적용될 수 있다. CASE 도구에 관계없이 적용될 수 있다. 따라서 UML 을 사용해 시스템 설계를 하였다.

위에서 소개한 UML은 객체 지향 설계를 위한 표준 언어로서, 소프트웨어 개발의 산출물을 시각화, 상세화, 구축, 문서화하는 특징을 가지고 있다. 이 4가지 특징에 대해 자세하게 설명한다.

1) UML은 시각화, 가시화 언어이다.
UML은 소프트웨어 개념 모델을 시각적인 그래픽 형태로 작성하며, 그 표기법에 있어서는 각 심볼에 대한 명확한 정의가 존재한다. 따라서 개발자들 사이에 오해 없는 원활한 의사소통이 이루어질 수 있으며, 여러 가지 시각적인 다이어그램을 통해 특정 관점으로 시스템을 표현할 수 있다.

2) UML은 _상세화, 명세화 _언어이다.
UML은 시스템 개발 과정인 분석, 설계, 구현 단계의 각 과정에서 필요한 모델을 정확하고 명백하고 완전하게 표현할 수 있는 수단이다. UML을 통해 시스템 설계의 복잡성을 명확하게 기술한다.

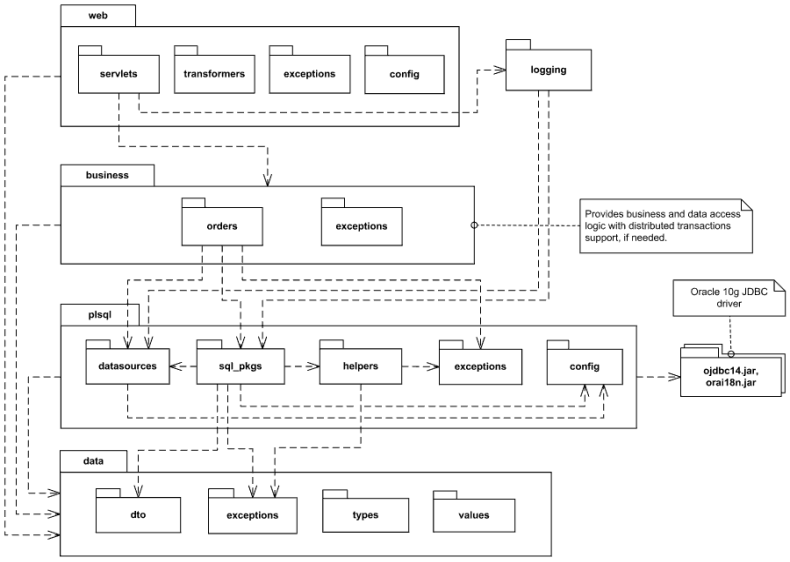
3) UML은 _구축 _언어이다.
UML 은 C++, Visual Basic, Java 등과 같은 다양한 프로그래밍 언어로 표현할 수 있다. 따라서 UML 로 상세화 된 설계 모델은 프로그래밍 소스 코드로 변환하여 구현이 가능하다. 또한 이미 구현되어 있는 소스 코드를 UML 로 역변환하는 데 이용할 수 있어서 분석하는 역공학(Reverse Engineering)이 가능하다.

B. Package Diagram

복잡한 시스템을 이해하는 방법은 추상적인 개념들을 하나의 그룹으로 묶어서 이해하는 것이다. 이렇게 만든 그룹은 클래스와 같은 추상 개념으로 많은 인스턴스들을 가지고, 오로지 시스템을 이해하기 위한 목적으로만 존재한다.

UML에서 시스템을 이해하기 위한 목적으로 추상적인 개념들을 모은 하나의 그룹을 패키지(Package)라고 한다. 패키지는 요소들의 그룹으로 조직하기 위한 범용 메커니즘으로 모델의 요소들을 조직하고 이해할 수 있도록 해준다. 패키지에 안에 담기는 것은 클래스에만 국한되는 것은 아니며, Use Case, Activity Diagram 등도 담을 수 있고, 다른 패키지도 담을 수 있다. 패키지를 구성할 때에는 여러 사람이 동의할 수 있는 형태로 구성되어야 하며, 패키지의 구성과 이름 체계는 개발자들이 쉽게 이해하고 사용할 수 있어야 한다.

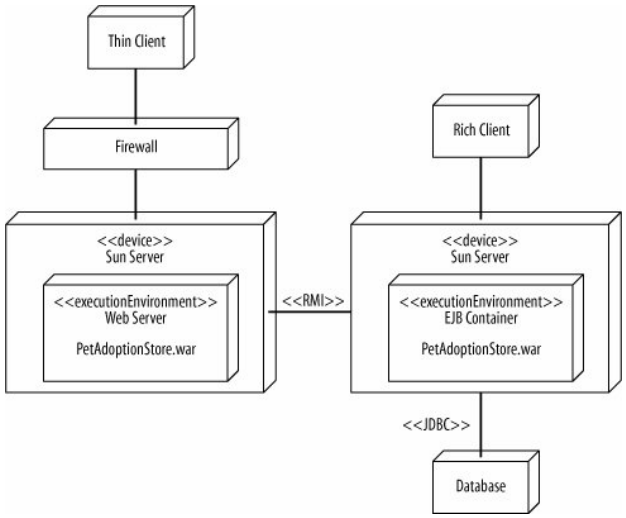
패키지 내부의 모든 클래스들은 개념적, 기능적, 변화적, 관리적 측면에서 유사한 면을 가진다. 하나의 패키지 내부의 클래스들은 밀접한 관련성을 가지며, 다른 패키지의 클래스들과는 약한 의존관계가 있다. 이와 같이 패키지 다이어그램은 패키지 삽입 및 패키지 확장을 포함하여 모델 요소들을 그룹화(패키지화)함으로써 조직 및 요소의 독립성을 묘사한다. 또한 패키지 다이어그램은 요소들간의 관계를 시각화하여 제공한다.



C. Deployment Diagram

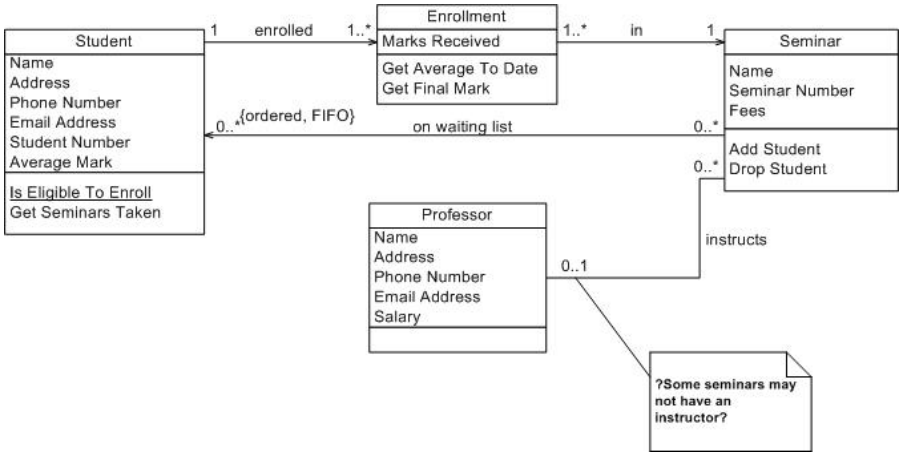
Deployment Diagram 은 네트워크, 하드웨어 또는 소프트웨어들을 실행 파일 수준의 컴포넌트들과 함께 표현한 다이어그램이다. 이들은 시스템을 구성하는 하드웨어간의 연결 관계를 표현하고, 하드웨어 자원에 대한 소프트웨어 컴포넌트의 배치 상태를 표현한 다이어그램이다. 즉, 시스템이 실행되는 환경인 노드와 그 노드에 배치된 컴포넌트의 구성을 나타내는 다이어그램이다. 노드는 처리 능력을 가진 장치를 의미하며, Deployment Diagram 에서 직육면체로 표기한다. Deployment Diagram 은 시스템의 설계 단계의 마지막에 작성되며, 이는 모든 설계가 마무리

되어야 소프트웨어의 컴포넌트가 정의되고, 시스템의 하드웨어 사양도 확정되기 때문이다. 배치 다이어그램은 소프트웨어 시스템이 배치, 실행될 하드웨어의 자원들을 정의하고, 소프트웨어의 컴포넌트가 어떤 하드웨어 자원에 탑재되어 실행될지를 정의하는 목적을 가지고 있다.



D. Class Diagram

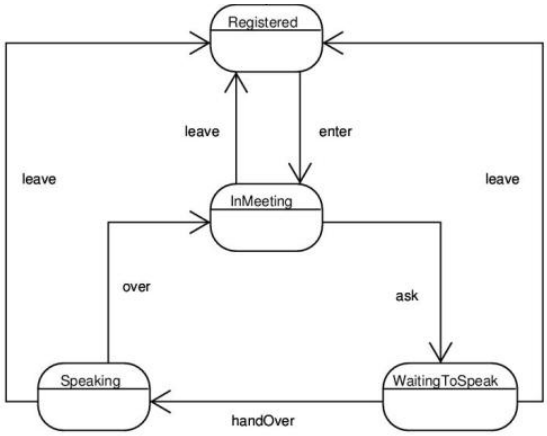
Class Diagram 은 객체지향 설계에서 가장 널리 사용되는 다이어그램으로, 시스템의 정적인 상태인 논리적인 구조(클래스)를 표현하는 다이어그램이다. 클래스(Class)란 객체지향 프로그래밍에서 속성(Attribute)과 행위(Method, Member Function)를 갖는 하나의 객체 단위이다. 시스템의 클래스와 클래스 Attribute, 클래스들의 관계를 나타낼 수 있기 때문에 객체지향 시스템의 상속 등의 관계를 명확하게 표현할 수 있다. 또한 코드 생성의 직접적인 원인이 되기 때문에 프로그래밍 개념과 같은 의미의 표현을 통해 도식화 한다.



E. State Diagram

State Diagram은 객체지향 모델에서 클래스의 인스턴스 사건(Event)에 의거한 시스템의 전체적인 작동을 상세히 기술하는 UML의 다이어그램이다. State Diagram은 상태의 변화에 의한 동작 또는 하나의 상태에서 다른 상태로 변화되게 하는 사건의 주어진 시간 동안의 상태를 나타낸다. 이러한 표기는 실제 구현에서 UI를 정의하는 데 도움을 준다.

상태 다이어그램은 어떤 이벤트에 대한 반응적인(Reactive) 특성을 가지는 객체에 대해 기술하기 위해 이용되며, 객체가 갖는 여러 가지 상태를 표현한다. 객체는 기존 상태에 따라 동일한 메시지에 대해서 다른 행동을 보이기 때문에, 상태 다이어그램을 통해 시스템 내의 객체가 가질 수 있는 상태가 어떤 것이 있는지에 대해, 또 각 상태를 나타낼 때 특정 이벤트에 대해 어떤 반응을 보일지에 대해 기술한다.

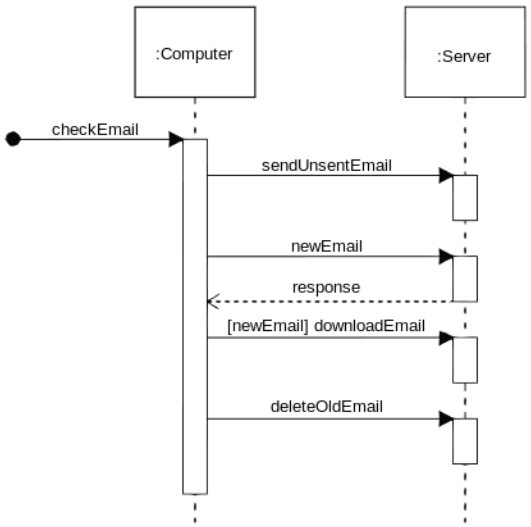


F. Sequence Diagram

Sequence Diagram은 시스템 내에서의 각 컴포넌트들이 주고 받는 메시지의 흐름을 시간 순차적으로 표현하는 상호작용 다이어그램이다. Sequence Diagram은 Use-Case Diagram과 자주 같이 사용되는데, 이는 각 Use-Case를 상세히 표현하는데 Sequence Diagram을 사용하는 것이 유리하기 때문이다. 각 컴포넌트들의 상호작용 명확히 보이기 때문에 각 컴포넌트간의 관계와 각 컴포넌트들이 갖고 있는 속성, 행동들을 더욱 명확히 할 수 있다. 따라서 Sequence Diagram은 시스템 내부의 동적인 움직임을 표현해주는 다이어그램이기 때문에 속성 및 함수로 이루어진 Class Diagram을 동적 프로그래밍으로 설계하는 중요한 과정이다.

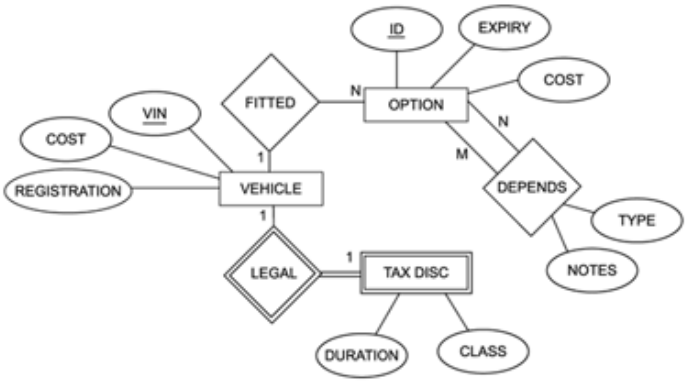
Sequence Diagram에서는 객체의 오퍼레이션과 속성을 상세히 정의한다. 이는 객체간 상호작용을 정의하는 과정에서 객체들이 가져야 하는 오퍼레이션과 속성을 구체적으로 정의할 필요가 있기 때문이다. 객체는 다른 객체가 의뢰하는 일을 처리하는 객체의 책임으로서, 객체의 책임은 오퍼레이션으로 정의되어야 하며, 이 행위를 위해 필요한 객체의 속성도 정의되어야 한다.

Sequence Diagram은 상호작용 다이어그램의 일종으로, 시스템의 동적 측면을 모델링 하기 위한 용도로 사용되며, 객체들과 그 사이의 교류 및 메시지를 보여주는 역할을 한다. Sequence Diagram은 Actor, 객체, 메시지, 회귀 메시지, 제어 블록 등으로 구성된다.



G. ER Diagram

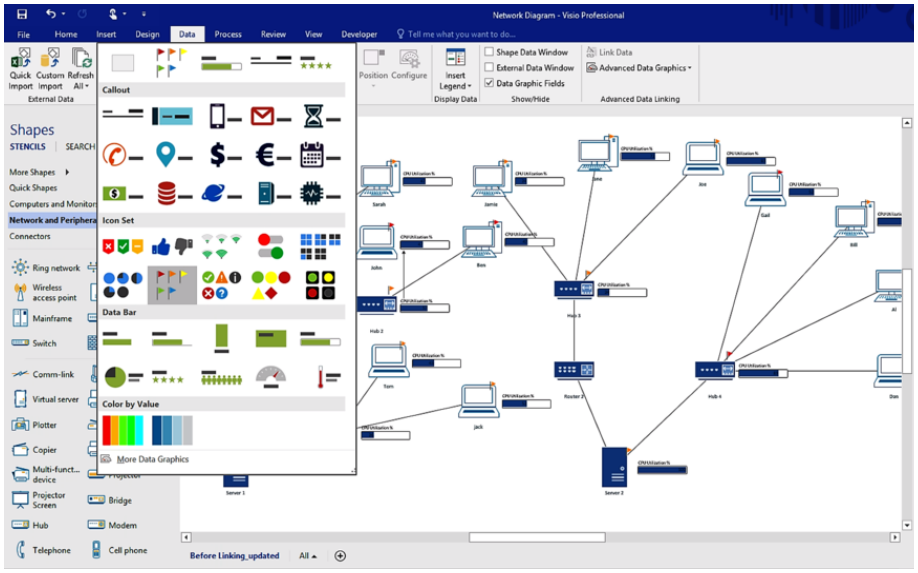
ER Diagram 은 데이터베이스에서 각 개체들의 관계를 표현하기 위해 사용하는 다이어그램으로 UML 에 포함되는 다이어그램은 아니다. 데이터의 저장 공간인 데이터베이스에 저장되는 데이터의 구조 및 그에 수반한 제약 조건들은 다양한 기법에 의해 정의될 수 있다. 그 기법 중 하나가 개체-관계 모델링(Entity-Relationship Modelling)이며, ERM 프로세스의 산출물이 ER Diagram(Entity-Relationship Diagram)이다. ER Diagram 에서 개체(Entity)란 현실 세계의 객체로서 유형 또는 무형의 정보를 대상으로 서로 구별할 수 있는 것이며, 관계(Relationship)란 두 개 이상의 개체 사이에 연관성을 기술한 것이다. 따라서 ER Diagram 은 조직의 정보 자원(Resource)을 전반적으로 계획하는데 있어서 필수적이며 유용한 도구이다.



2.3 Applied Tools

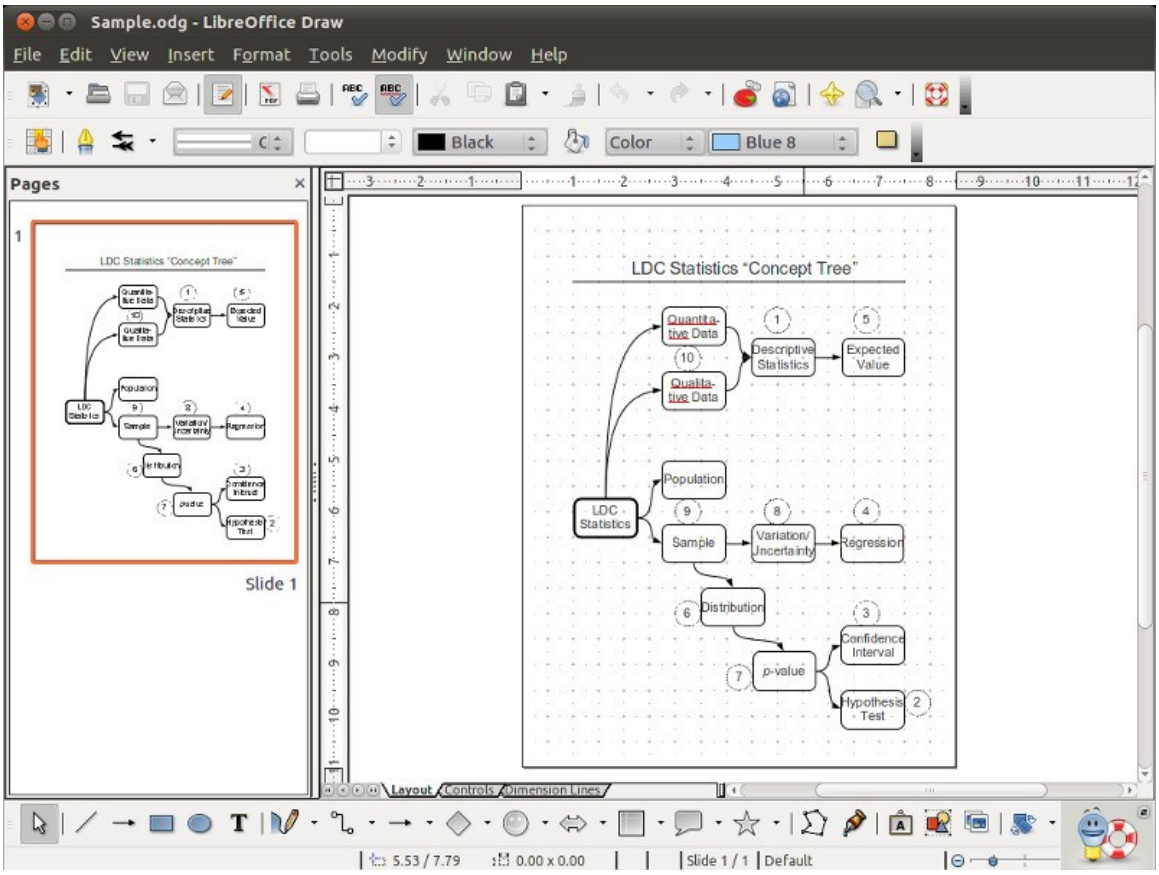
A. Microsoft Visio

Microsoft Visio 는 그림이나 도표를 그리는 데 사용되는 소프트웨어이다. 벡터 그래픽을 사용하여 그림을 그려내기 때문에 그림을 확대해도 깨지거나 손상될 우려가 없다. 기본적인 여러 스텐실을 제공하므로, 매우 편리하게 다이어그램을 그릴 수 있다.



B. LibreOffice Draw

리브레오피스 드로우는 간단한 스케치에서부터 복잡한 계획도까지 다양한 그림과 도형, 다이어그램을 그릴 수 있는 프로그램이다. 드로우를 이용하여 쉽게 직관적인 플로우차트를 그릴 수 있다. 스마트 연결자 기능은 플로우차트를 손쉽게 만들고, 차크를 구성하고, 다이어그램을 연결할 수 있게 해준다.

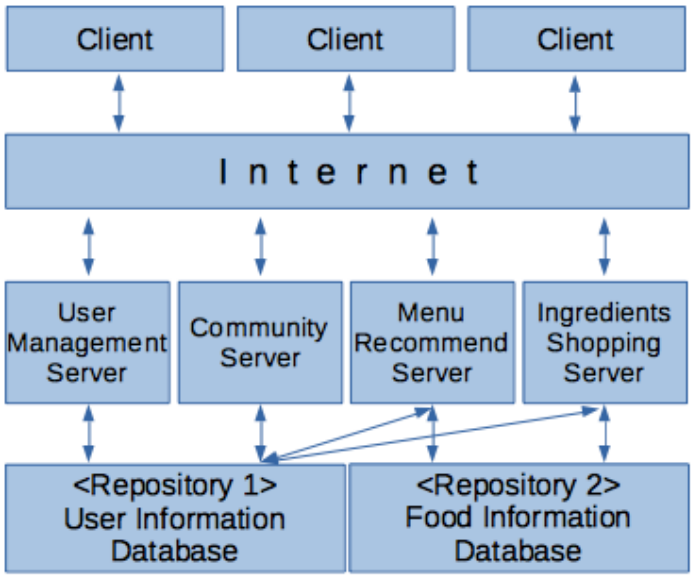


3. System Architecture

3.1 Objectives

System Architecture 에서는 목표 시스템에 대한 고수준의 개요를 보여주고, 시스템 기능의 전체적 분포를 보여준다. System 의 구조를 Block Diagram 으로 나타낸 뒤, 그것들의 관계와 실제 사용을 Package Diagram, Development Diagram 으로 표현한다. 각 서브 시스템의 modular decomposition 방식은 대제목 번호 4, 5, 6, 7 번에서 설명한다.

3.2 System Organization



메뉴얼(Menu-All) 시스템은 Client-Server Model (클라이언트-서버 모델)과 Repository Model(리포지토리 모델)을 사용하여 구현된다. 유저와 시스템 사이에는 클라이언트-서버 모델을 사용하여 연결되고, 각 시스템은 리포지토리 모델을 사용하여 작동한다.

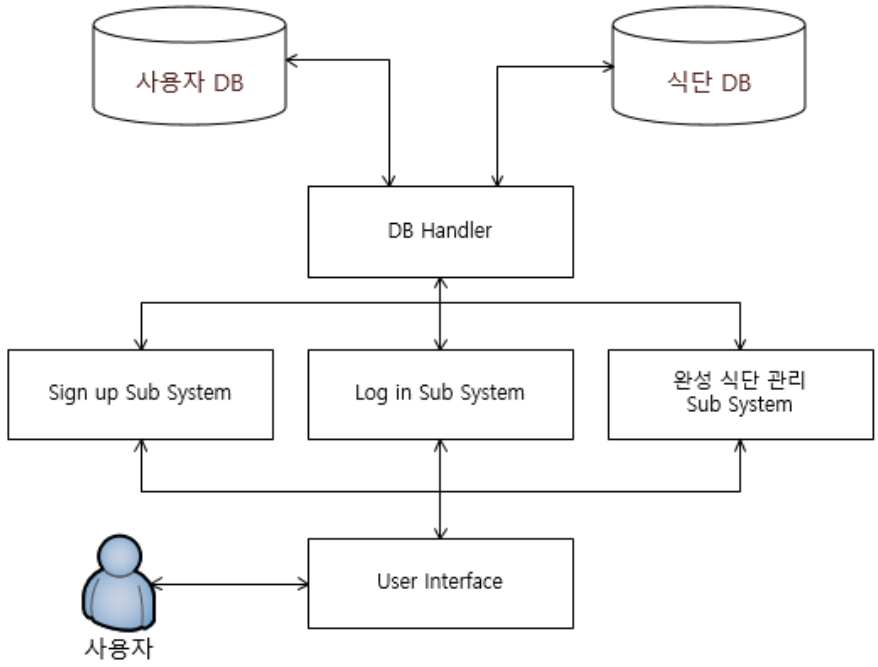
클라이언트-서버 모델은 네트워크를 이용하는 분산 시스템 형태의 모델로, 데이터와 처리 기능을 클라이언트와 서버에 분할하여 사용한다. 서버, 서비스, 클라이언트로 구성되며, 서브시스템들이 서비스를 서로 요청하면서 상호작용한다. 이 모델은 분산 아키텍쳐에 유용하다. 서버(Server)는 클라이언트에 서비스를 제공한다. 메뉴얼

시스템에는 유저 관리 서버(User Management Server), 맞춤 식단 서버(Menu Recommendation Server), 커뮤니티 서버(Community Server), 재료 구매 서버(Ingredients Shopping Server) 네 가지의 서버가 있다. 클라이언트(Client)는 사용자와 대화하기 위해 서버가 제공하는 서비스를 요청하는 서브시스템이다. 본 시스템에서는 쉽게 말해 서비스에 접근하여 서비스를 이용하는 각각의 유저라고 생각하면 된다. 메시지나 원격 프로시저 호출을 통해 서비스를 요청하고, 서버가 이 요청을 받아 수행한 후 그 결과를 클라이언트에 보낸다. 클라이언트가 서버에 서비스를 요청할 때는 서버의 이름과 서버가 제공하는 서비스의 이름을 알아야 한다.

다음으로 리포지토리 모델을 사용하는 이유는 데이터가 리포지토리에서 중앙 관리될 수록 있도록 하기 위함이다. 즉, 리포지토리 모델은 데이터 중심형 모델이다. 리포지토리에 공동으로 활용하는 데이터를 보관하고, 모든 서브시스템이 여기에 저장된 공유데이터에 접근하여 정보를 저장, 검색, 변경한다. 이 모델에서는 데이터가 한 군데에 모여 있기 때문에 데이터를 모순되지 않고 일관성 있게 관리할 수 있게 한다. 대량의 데이터를 공유하는 시스템에 매우 유용한 모델이다. 본 시스템에는 두 리포지토리가 있다. 각각 사용자와 관련된 정보(User Information)를 저장하는 리포지토리와, 음식과 관련된 정보(Food Information)를 저장하는 리포지토리이다.

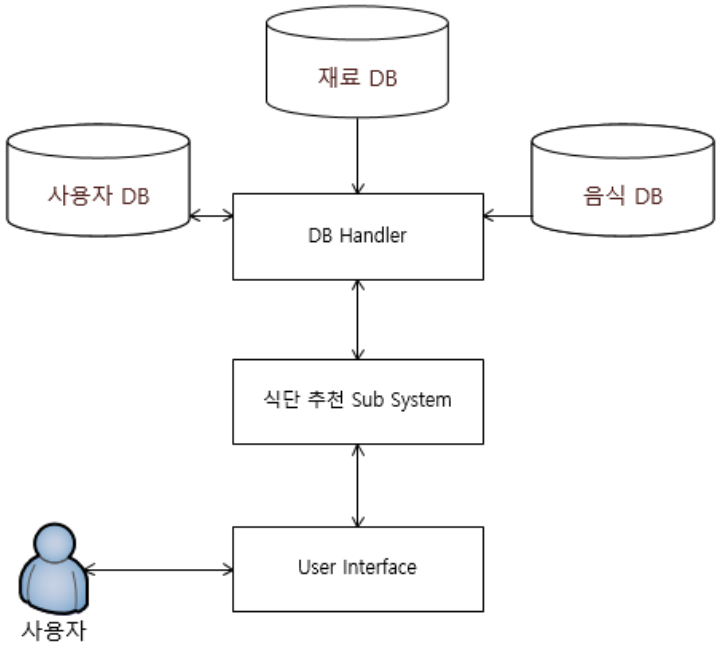
A. 유저 관리 시스템 (User Management System)

유저 관리 서버는 사용자의 이용과 정보에 관련된 시스템이다. 회원가입, 로그인, 식단 관리의 3 가지 서브시스템으로 이루어진다.



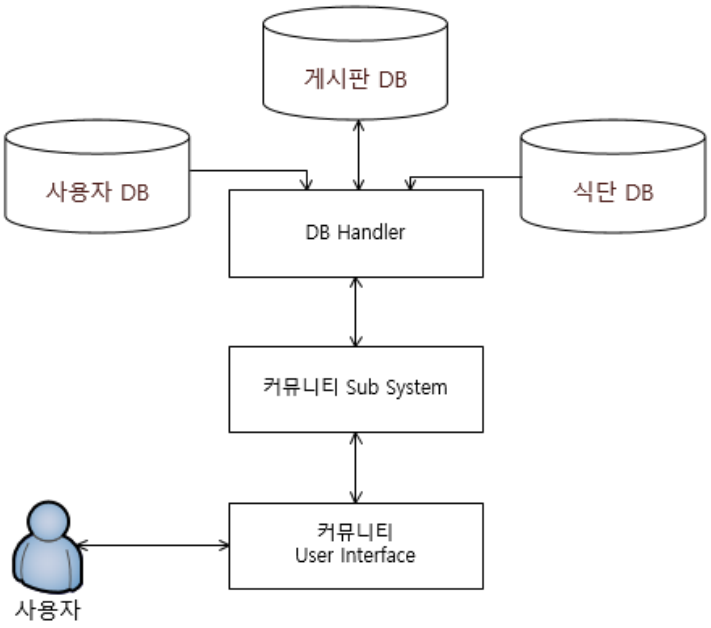
B. 맞춤 식단 시스템 (Menu Recommendation System)

맞춤 식단 서버는 사용자의 입력에 따라 식단을 생성하여 출력해주는 시스템이다. 식단 조건 설정, 맞춤 식단 생성, 맞춤 식단 출력, 맞춤 식단 수정의 4 가지 서브시스템으로 이루어진다.



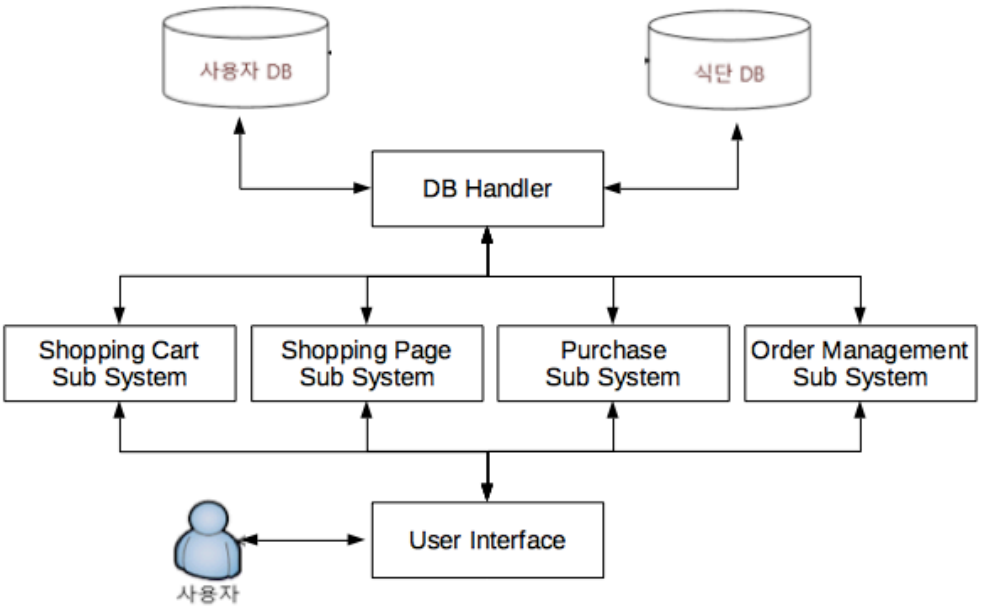
C. 커뮤니티 시스템 (Community System)

커뮤니티 시스템은



D. 재료 구매 시스템 (Ingredients Shopping System)

재료 구매 서버는 사용자가 출력된 식단의 재료를 구매할 수 있게 하는 시스템이다. 장바구니, 쇼핑페이지, 결제, 주문 관리의 4 가지 서브시스템으로 이루어진다.



3.3 Package Diagram

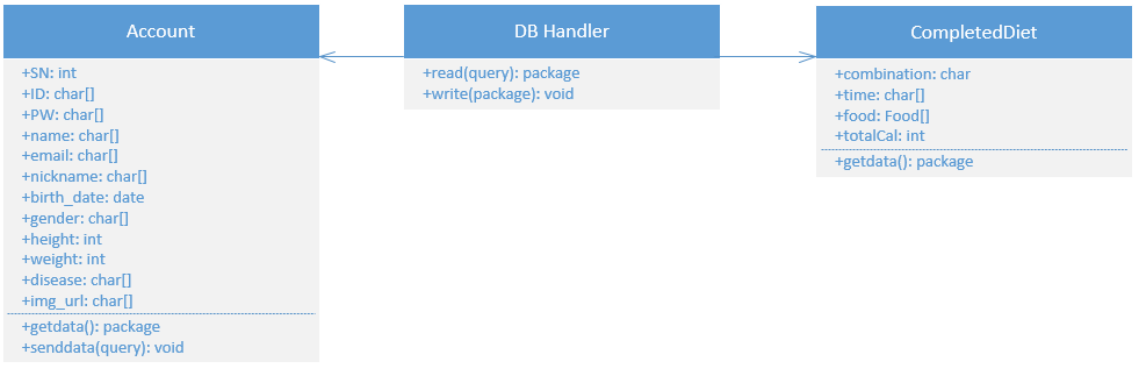
3.4 Development Diagram

4. 유저 관리 시스템 (User Management System)

1.1 Objectives

실제 사용자가 회원 가입과 로그인 기능을 통해 시스템을 이용하는 도중에 발생하는 데이터를 처리하고 도식화하는 사용자 관리 시스템의 설계를 설명한다. 또한 사용자의 완성된 식단 관리 기능에 대한 설계를 설명한다. Class Diagram, Sequence Diagram, State Diagram 을 통해 사용자 관리 시스템의 구조를 표현하고 설명한다.

1.2 Class Diagram



A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+package Read(query): 해당되는 DB 에서 원하는 data 를 읽어온다.

+void write(getdata): 해당되는 DB 에 data 를 저장한다.

B. Account

B.1. Attributes

+SN: account 고유번호 값

+ID: account ID 정보

+PW: account 비밀번호 정보

+name: account 이름 정보

+email: account e-mail 정보

+nickname: account 닉네임 정보

+birth_date: account 생년월일 정보

+gender: account 성별 정보

+height: account 키 정보

+weight: account 몸무게 정보

+disease: account 질병 정보
+img_url: account 완성 식단 이미지의 주소 정보

B.2. Methods
+package getdata(): DB 로부터 data 를 받는다.
+void sendData(query): DB 로 data 를 보낸다.

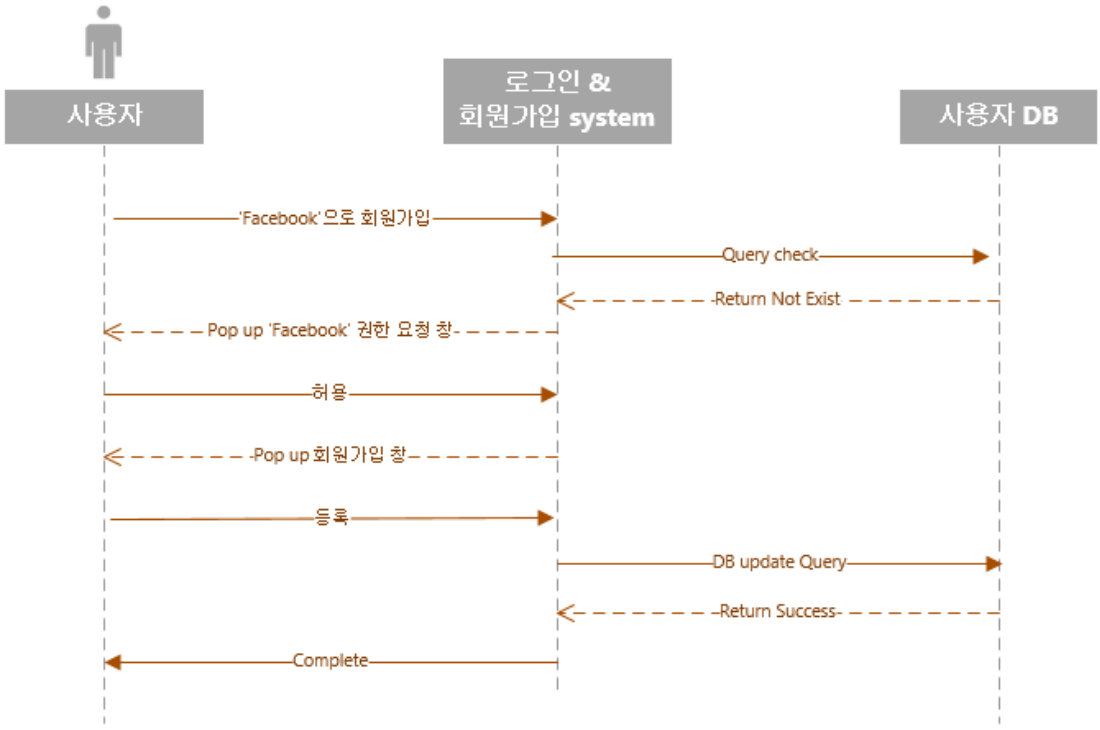
C. CompletedDiet

C.1. Attributes
+combination:
+time:
+food:
+totalCal:

C.2. Methods
+package getdata(): DB 로부터 data 를 받는다.

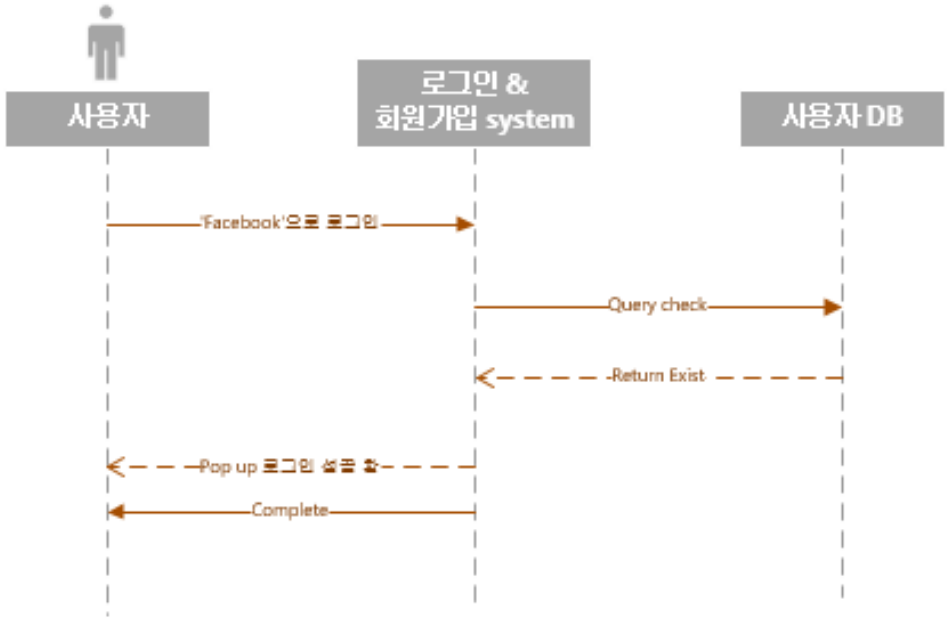
1.3 Sequence Diagram

A. Sign up with Facebook

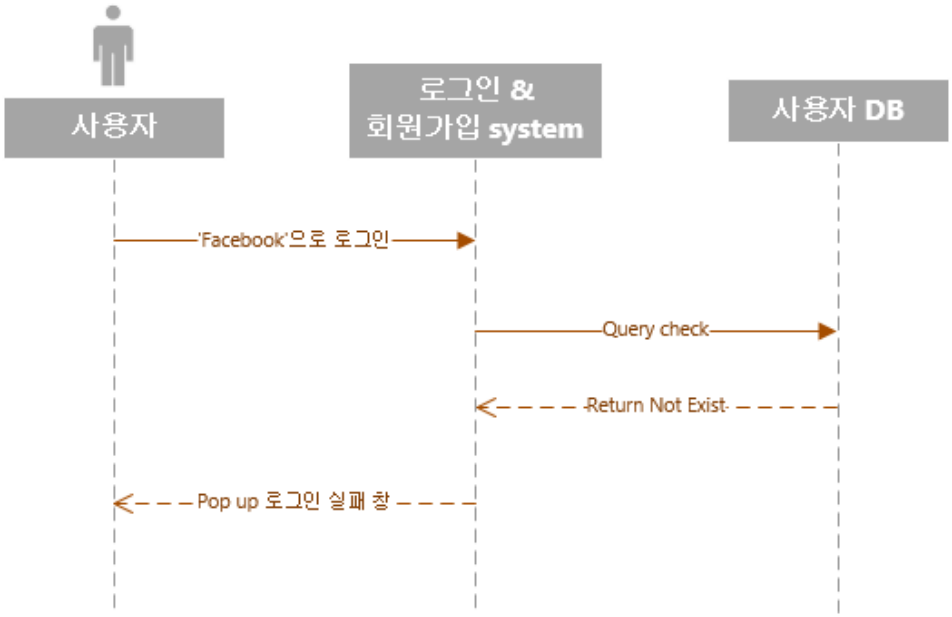


B. Log in with Facebook

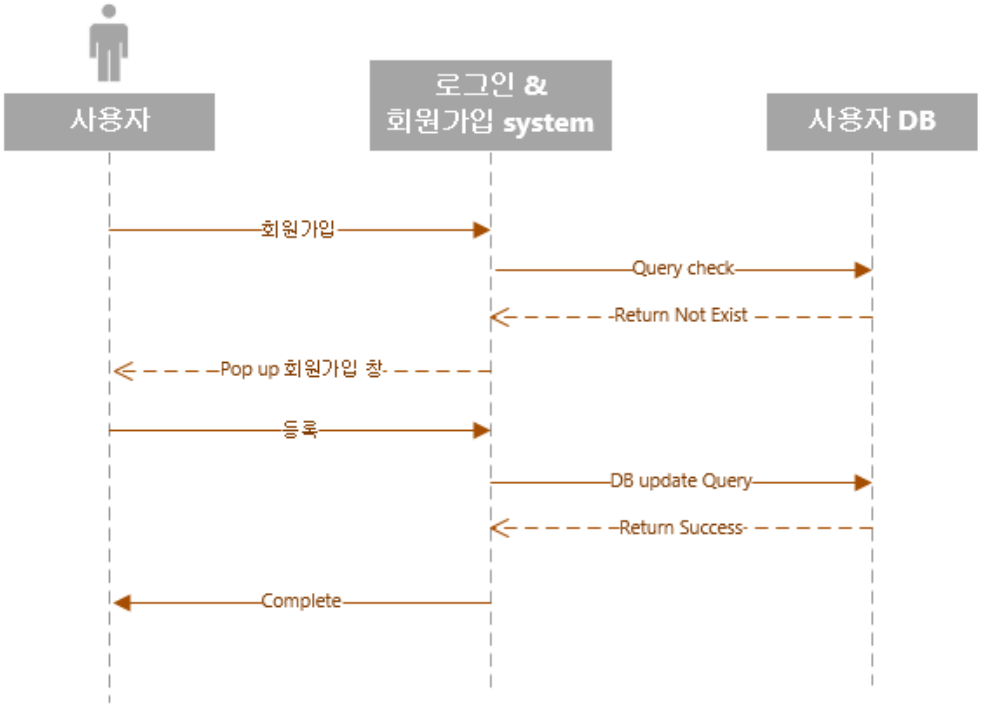
B.1. Log in Success



B.1. Log in Failure

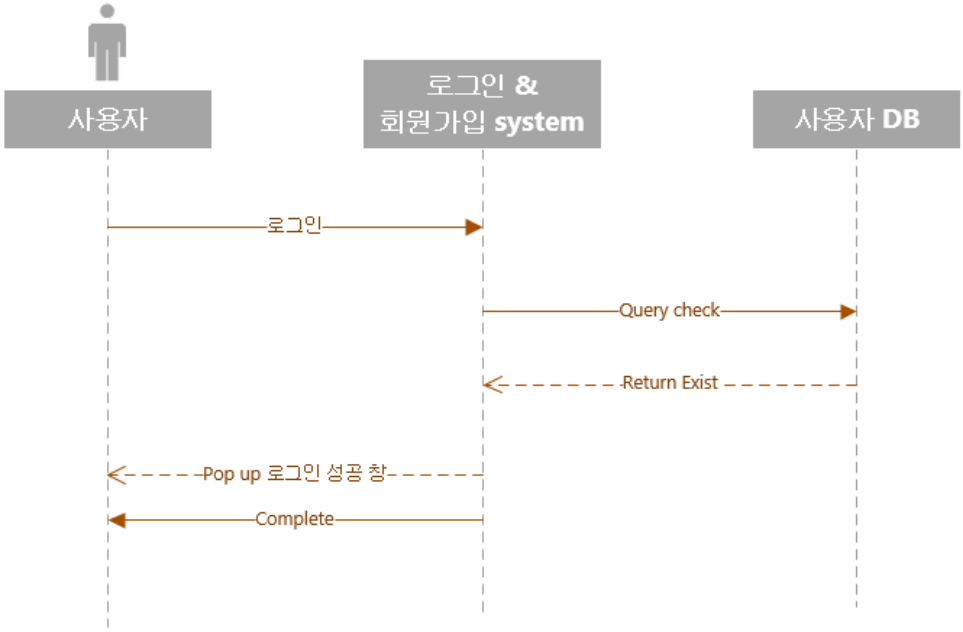


C. Sign up without Facebook

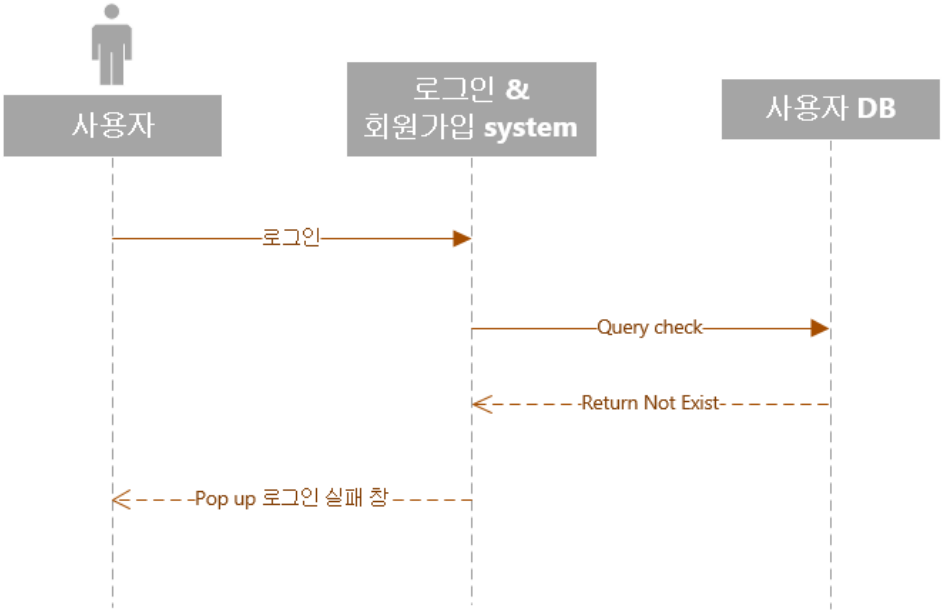


D. Log in without Facebook

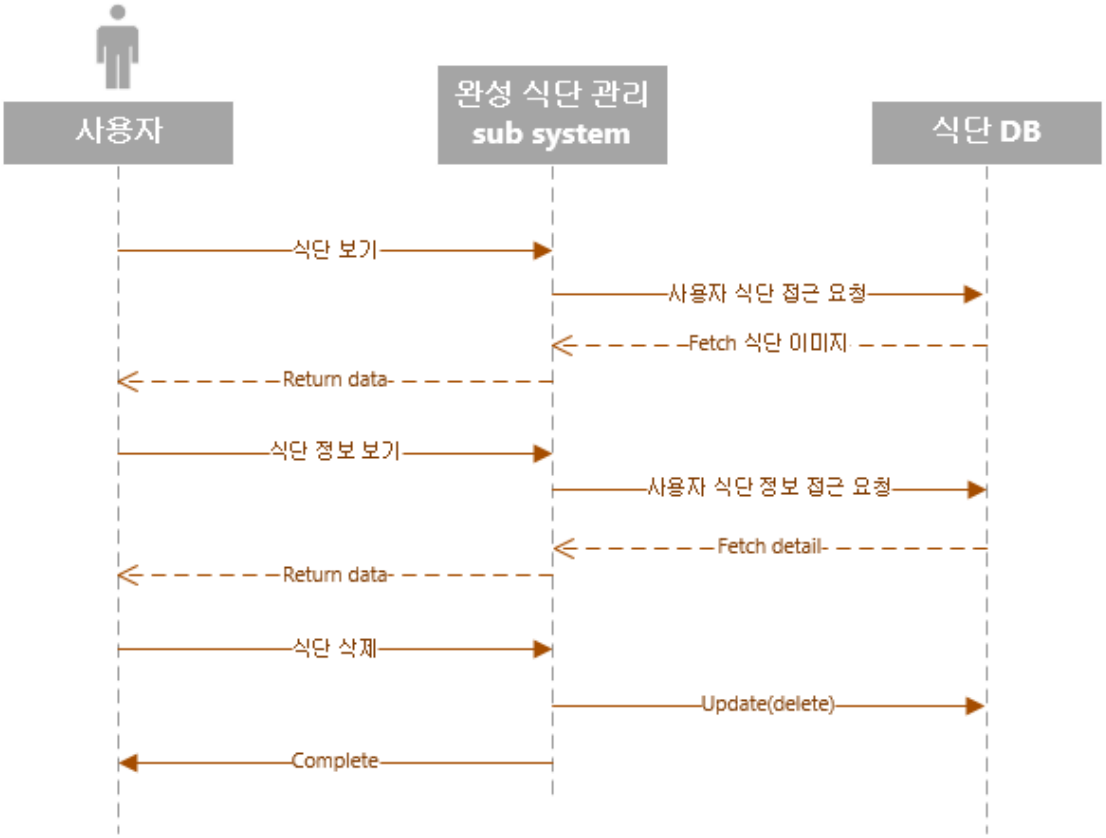
D.1. Log in Success



D.1. Log in Failure

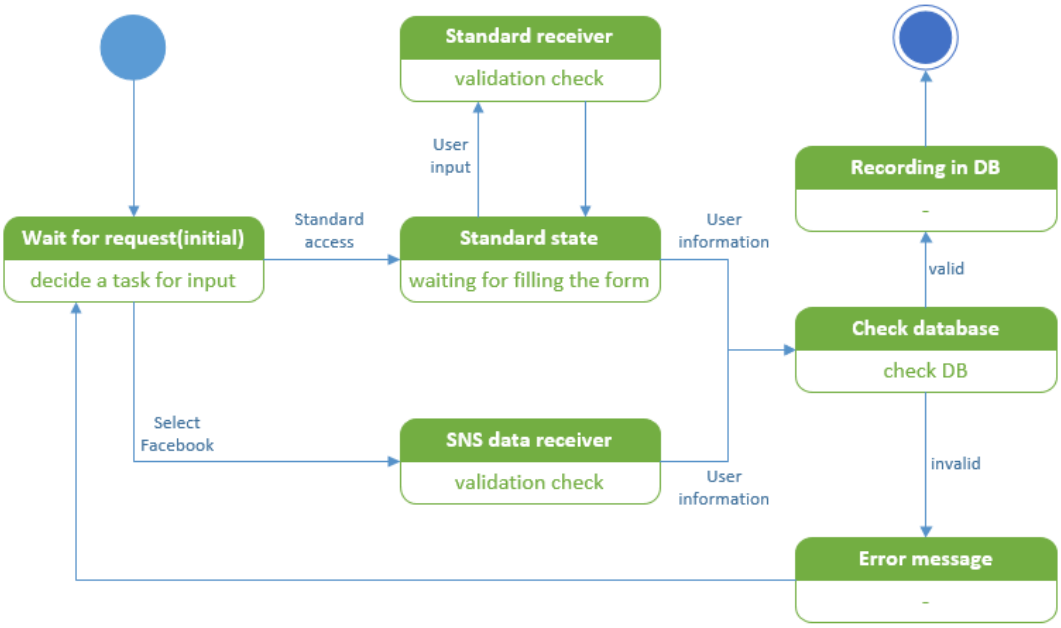


D. Menu Management (완성 식단 관리)

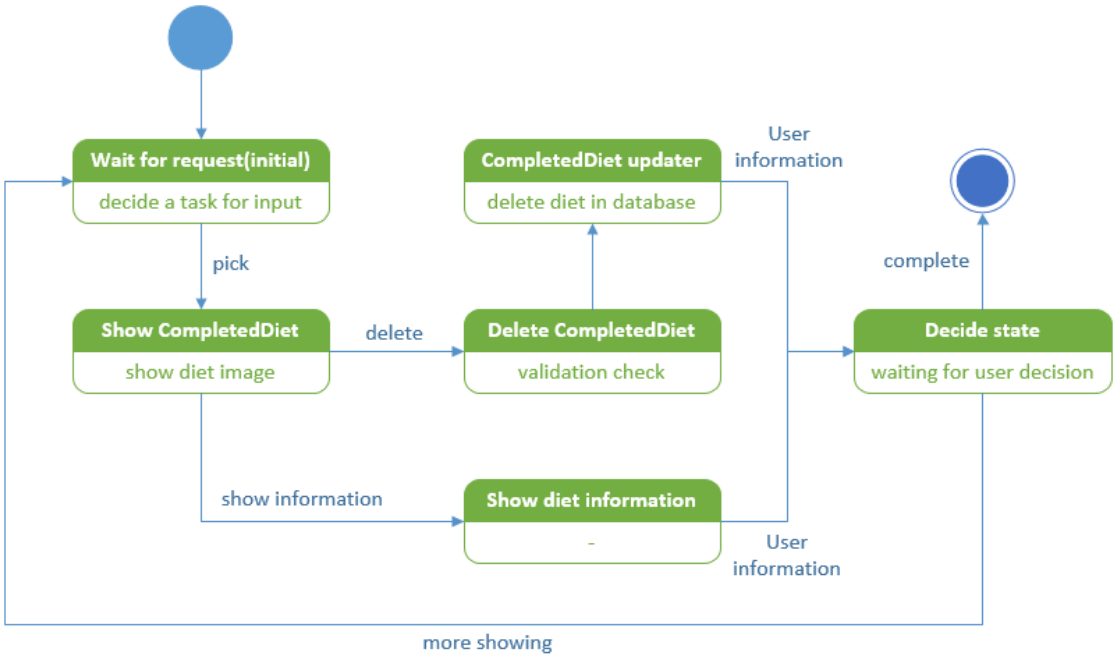


1.4 State Diagram

A. Sign up



B. Menu Management

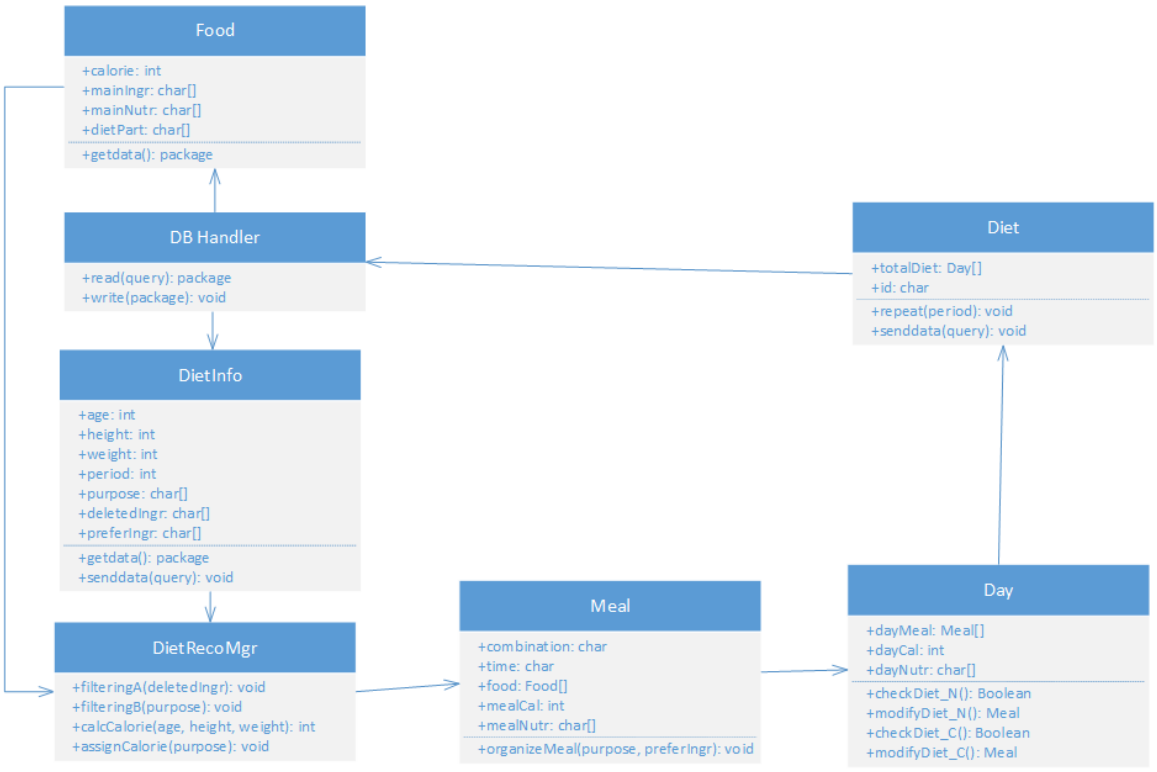


5. 맞춤 식단 시스템 (Menu Recommendation System)

1.1 Objectives

사용자로부터 필요한 정보를 입력받으면 그 정보에 맞춰서 사용자 맞춤 식단을 추천해 주는 시스템의 설계를 설명한다. Class Diagram, Sequence Diagram 과 State Diagram 을 통해 맞춤 식단 시스템의 구조와 사용자와의 상호 작용을 표현하고 설명한다.

1.2 Class Diagram



A. DB Handler

A.1. Attribute

해당 사항 없음.

A.2. Method

+package read(): DB 에서 원하는 데이터를 읽어온다.

+void write(getdata): DB 에 데이터를 저장한다.

B. DietInfo

B.1. Attribute
+age(int): 사용자의 나이
+height(int): 사용자의 키
+weight(double): 사용자의 몸무게
+period(int): 식단의 기간
+purpose(char[]): 식단의 목적
+deletedIngr(char[]): deleted ingredient. 식단 구성에서 제거할 재료
+preferIngr(char[]): preference ingredient. 사용자가 선호하는 재료
B.2. Method
+package getdata(): DB 로부터 데이터를 받음
+void senddata(query): DB 로 데이터를 보냄

C. Food
C.1. Attribute
+calorie(int): 음식의 총 칼로리
+mainIngr(char[]): 음식에 들어가는 주요 재료 3~5 가지
+mainNutr(char[]): 음식의 주 영양소 (key:영양소 이름, value: 영양소의 수치값)
+dietPart(char[]): 식단 중 담당파트 (밥, 면, 국/탕, 찌개/전골, 메인반찬, 서브반찬, 후식)
C.2. Method
+package getdata(): DB 로부터 데이터를 받음

D. DietRecoMgr (Diet Recommendation Manager)
D.1. Attribute
해당 사항 없음.
D.2. Method
+void filteringA(deletedIngr) : 제거할 재료를 필터링
+void filteringB(purpose) : 식단의 목적에 따라 사용자가 섭취 가능한 영양소 제한을 초과하는 음식을 필터링
+int calcCalorie(age, height, weight) : 사용자의 1 일 권장 칼로리를 계산
+void assignCalorie(purpose) : 식단의 목적에 따라 아침, 점심, 저녁에 칼로리를 할당

E. Meal
E.1. Attribute
+combination(char) : 음식의 조합(밥/국/반찬...). a~e 로 표시
+time(char) : 아침(B. Breakfast)/점심(L. Lunch)/저녁(D. Dinner)
+food(Food[]) : 그 끼니에 포함하는 음식들
+mealCal(int) : 그 끼니의 총 칼로리
+mealNutr(char[]) : 그 끼니의 총 영양소
E.2. Method
+void organizeMeal(purpose, preferIngr) : 조건에 맞춰 한 끼 식단 구성

F. Day

F.1. Attribute

+dayMeal(Meal[]) : 아침, 점심, 저녁의 식단 구성
+dayCal(int) : 그날 식단의 총 칼로리
+dayNutr(char[]) : 그날 식단의 총 영양소

F.2. Method

+Boolean checkDiet_N() : 영양소 허용량을 초과하지 않는지 체크
+Meal modifyDiet_N(): 영양소를 초과하여 수정이 필요한 경우 식단을 수정
+Boolean checkDiet_C() : 칼로리가 하루 권장 칼로리를 초과하지 않는지 체크
+Meal modifyDiet_C(): 칼로리를 초과하여 수정이 필요한 경우 식단을 수정

G. Diet

G.1. Attribute

+totalDiet(Day[]) : 기간에 맞춘 식단 구성
+id(char) : 사용자 식별용 id

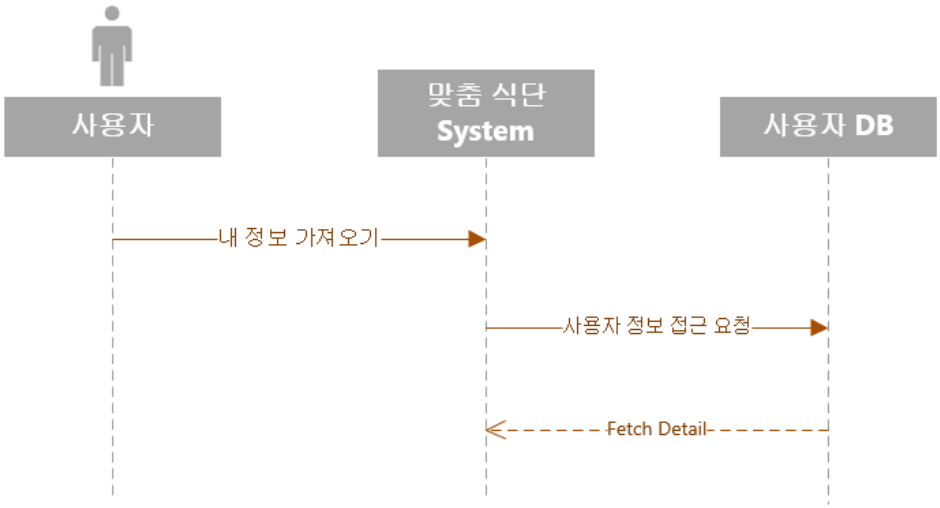
G.2. Method

+void repeat(period) : 식단의 기간을 받아 날짜만큼 반복하여 식단을 채움
+void senddata(query): DB 로 데이터를 보냄

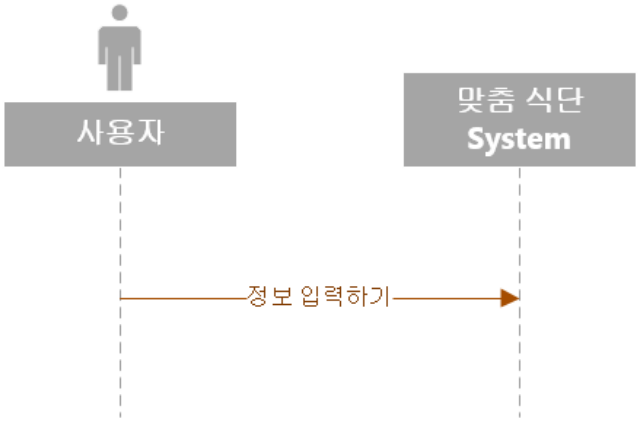
1.3 Sequence Diagram

A. Enter User Information (사용자 정보 입력)

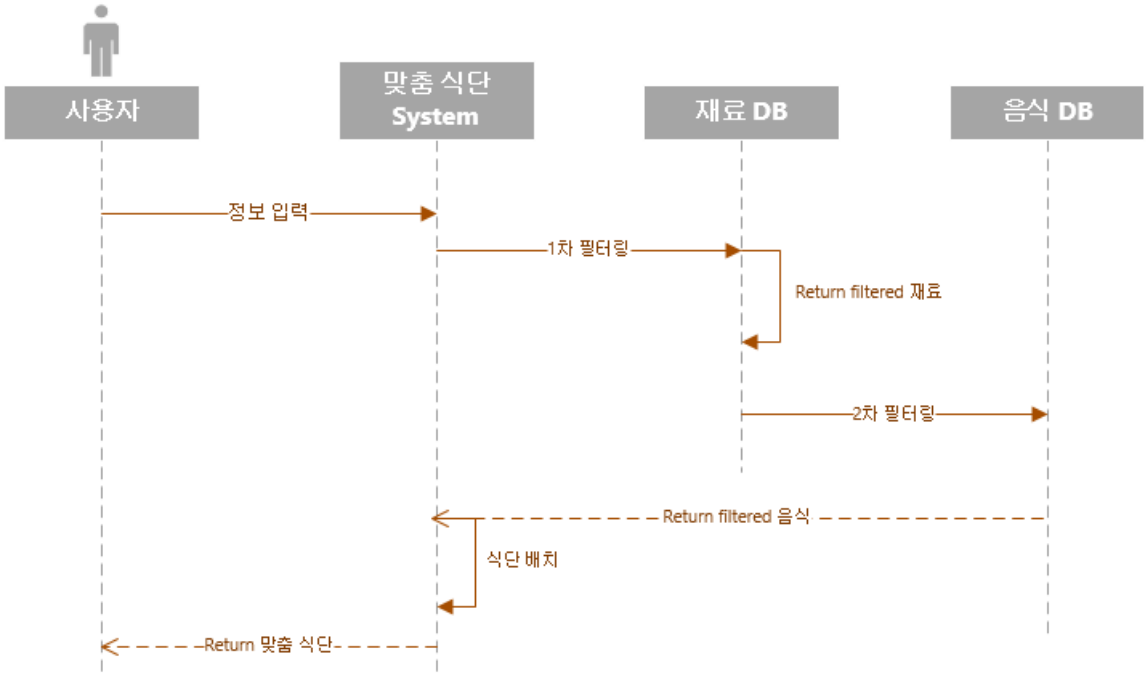
A.1. 정보 불러오기



A.2. 직접 입력하기

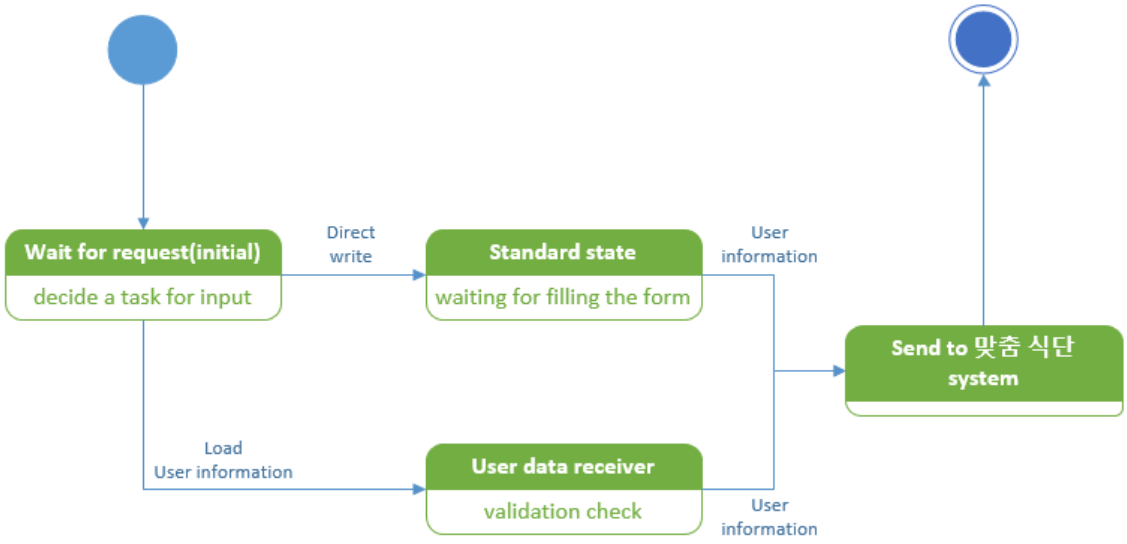


B. Menu Recommendation (식단 추천)

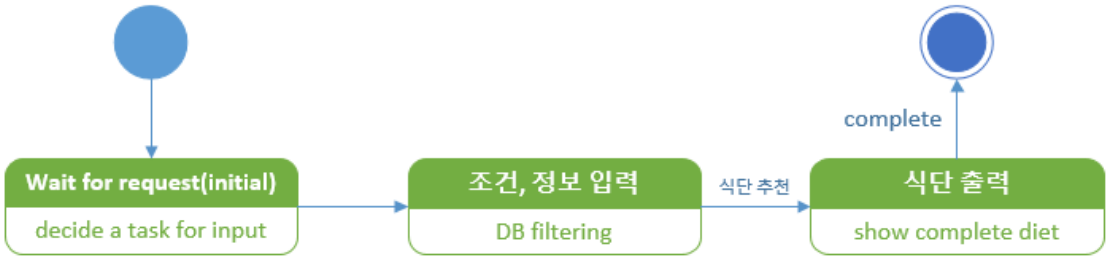


1.4 State Diagram

A. 식단 조건 입력



B. 식단 추천



6. 커뮤니티 시스템 (Community System)

1.1 Objectives

1.2 Class Diagram

1.3 Sequence Diagram

1.4 State Diagram

7. 재료 구매 시스템 (Ingredients Shopping System)

1.1 Objectives

1.2 Class Diagram

1.3 Sequence Diagram

1.4 State Diagram

8. Protocol Design

9. Database Design

10. Testing Plan

11. Development Environment

12. Index