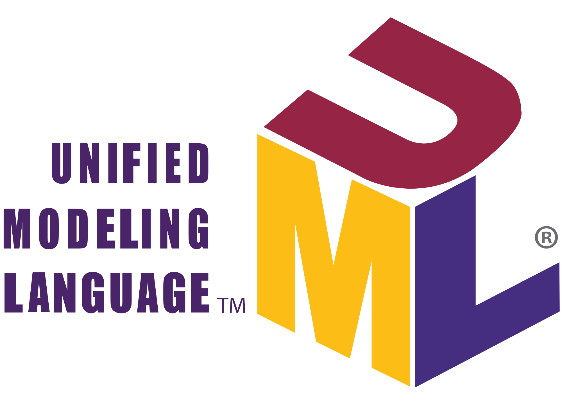
1. **Introduction**
   1. **Objective**

Introduction에서는 본 명세서의 System Design을 위해 사용된 Diagram과 Tool을 소개하고 본 프로젝트의 범위에 대해 논한다.

* 1. **Applied Diagrams**
     1. **Block Diagram**

본 설계 명세서에는 시스템과 각 Subsystem의 전반적인 구조를 표현하기 위해 간단한 비공식 Block Diagram을 사용한다. 이는 특정 System, 혹은 Subsystem의 Entity와 Database, 그들 간의 Interaction과 교환하는 정보를 간략하게 기술한다.

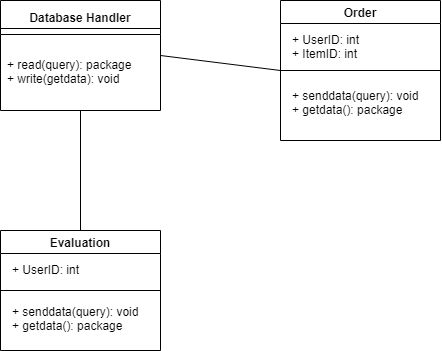
* + 1. **UML**

****

[Figure ] Logo of UML

UML(Unified Modeling Language)은 소프트웨어 공학에서 사용되는 표준화된 범용 modeling 언어로, 현업에서 많이 사용되는 여러 Diagram들을 통합하여 만들어졌다. UML은 Object-Oriented Design을 수행할 때 시스템의 구조를 표준화된 시각적 Model로 표현하여 시스템의 Stakeholder들의 의사소통을 원활히 하기 위해 사용된다. 본 설계 명세서에는 UML의 Class Diagram, Sequence Diagram, State Diagram을 사용하여 개발할 시스템의 구조를 시각화 한다.

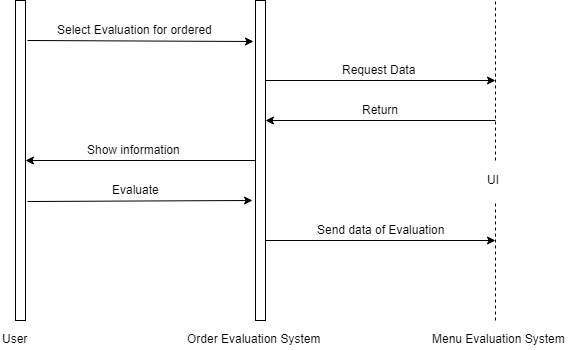
* + 1. **Class Diagram**

****

[Figure ] Example of Class Diagram

본 설계 명세서에서는 Class Diagram을 이용하여 시스템의 object들을 class로 표현한다. Class Diagram은 OOD(Object-oriented Design)의 관점에서 시스템의 정적 구조를 표현하는 UML Diagram으로, 시스템의 class들과 그들 사이의 관계인 Association을 나타낸다. 여기서 Class란 System Object의 한 종류에 대한 보편적인 정의이며, Association이란 Class들 사이의 관계를 나타내는 link를 의미한다.

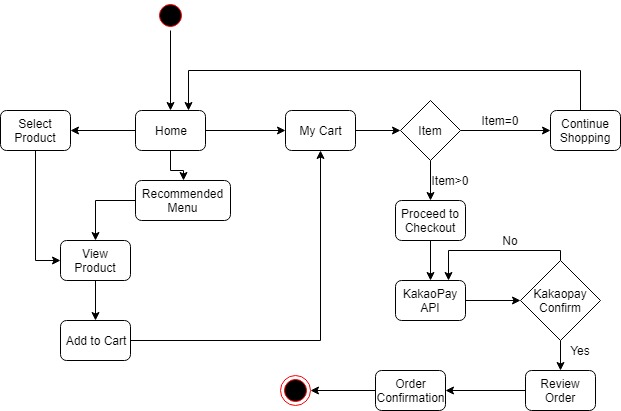
* + 1. **Sequence Diagram**

****

[Figure ] Example of Sequence Diagram

본 설계 명세서에서는 Sequence Diagram을 이용하여 시스템의 구성요소들 간의 상호작용을 표현한다. Sequence Diagram이란 시스템과 그의 Actor, 그리고 System Components간의 상호작용을 나타내는 UML Diagram이다. 이는 시스템의 특정 use case에서 일어나는 상호작용들의 sequence로 표현된다.

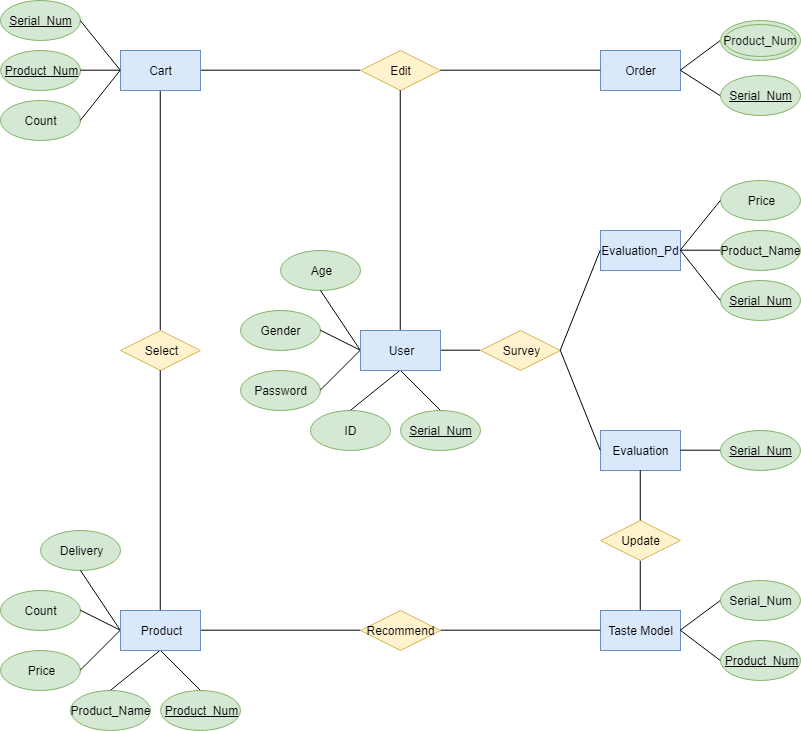
* + 1. **State Diagram**



[Figure ] Example of State Diagram

본 설계 명세서에서는 State Diagram을 이용하여 시스템의 내부, 외부에서 일어나는 Event들에 대한 시스템의 반응을 표현한다. State Diagram은 OOD의 관점에서 시스템의 동적 반응을 설명하기 위해 사용되는 Diagram으로, Class의 Event에 대한 시스템의 전체적인 반응, 즉 작동을 상세하게 기술하여 나타낸다. 본래의 상태에서 특정 Event가 발생하였을 때 시스템의 동작과 함께 클래스의 상태(State)가 변하게 되는데, 이 때의 동작과 상태의 변화를 기술한다.

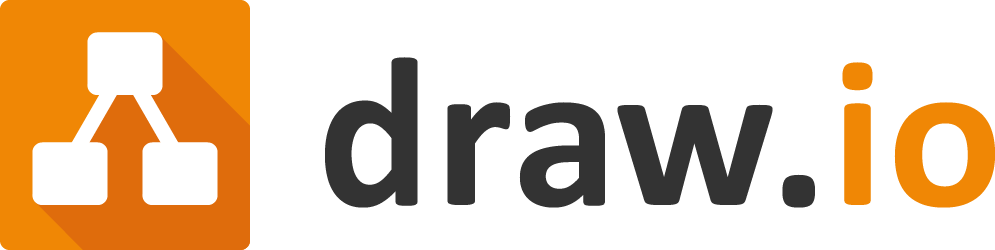
* + 1. **ER Diagram**



[Figure ] Example of ER Diagram

Entity-Relationship(ER) Model은 Database Design Process중 데이터베이스를 Table과 Constraints로 나타내는 Relational Model 이전에 수행되는 단계로, 데이터베이스를 Entity와 Entity들의 Relationship으로 나타내는 Conceptual Model이다. ER Model에서 Entity는 Real World에서의 특정한 Object를 Database에 표현한 것이고, Attribute는 Entity를 나타내기 위해 사용되는 특성이며, Relationship은 다수의 Entity들을 구체적인 의미를 통해 연결하는 것이다. 본 문서의 Database Design에서 사용되는 ER Diagram은 위의 ER Model을 시각화 하기위해 사용하는 Diagram이다.

* 1. **Applied Tools**

****

[Figure 6] Logo of draw.io

본 설계 명세서의 모든 System Architecture와 Diagram들은 ‘draw.io’를 이용하여 작성했다. ‘draw.io’는 온라인 환경이라면 누구나 다양한 목적의 Flowchart와 Diagram을 작성할 수 있는 Online Diagram Tool이다. 접근성이 뛰어나고 사용법이 간단하기 때문에 자체적인 Diagram Tool을 보유하지 않은 회사에서 많이 사용된다.

* 1. **Project Scope**

‘Eat it’ 시스템은 사용자들이 하나하나 상품을 선택해야 했던 기존의 식품 쇼핑 서비스들의 단점을 보완하여 온라인 식품 쇼핑 시스템에 사용자 취향 기반 상품 추천 시스템을 접목한 서비스이다. 따라서 식품 쇼핑 기능과 취향 기반 추천 기능이 가장 중요한 System Objective이며, 7개의 Subsystem들의 상호작용으로 이 기능들을 구현하도록 설계되었다. 따라서 본 프로젝트의 구현 범위는 다음과 같다.

‘User Management System’으로 사용자의 로그인 정보를 관리하며, 이는 회원가입과 로그인 기능을 담당한다. ‘Taste Analysis System’으로 사용자의 음식 취향 모델을 생성, 학습한다. 여기서 생성된 취향 모델을 기반으로 ‘Menu Recommendation System’에서 적합한 메뉴를 제안한다. 또한 사용자는 ‘Searching System’을 이용하여 구매하고 싶은 특정 상품을 검색할 수도 있다. ‘Menu Recommendation System’이나 ‘Searching System’의 Output으로 도출된 메뉴들의 목록을 ‘Item Display System’이 출력한다. 사용자가 마음에 드는 상품을 장바구니에 담으면 ‘Ordering System’을 통해 상품들을 주문할 수 있다. 마지막으로 ‘My Page System’으로 주문한 내역을 관리하거나, 주문한 상품을 평가하여 취향 모델을 업데이트한다.