BALANCED ONE

Design Specification



|  |  |
| --- | --- |
| Student Number | Name |
| 2016314843 | 김윤재 |
| 2015312802 | 김두원 |
| 2016313088 | 조원이 |
| 2016313760 | 너드르 |
| 2015313016 | 이재현 |
| 2016313352 | 모하메드 |

Contents

[1. Preface 10](#_Toc24579313)

[1.1. Objectives 10](#_Toc24579314)

[1.2. Readership 10](#_Toc24579315)

[1.3. Document Structure 10](#_Toc24579316)

[A. Introduction 10](#_Toc24579317)

[B. System Architecture 10](#_Toc24579318)

[D. Database Design 11](#_Toc24579319)

[E. Testing Plan 11](#_Toc24579320)

[F. Development Plan 11](#_Toc24579321)

[G. Index 11](#_Toc24579322)

[2. Introduction 12](#_Toc24579323)

[2.1. Objectives 12](#_Toc24579324)

[2.2 Applied Diagram 12](#_Toc24579325)

[A. UML 12](#_Toc24579326)

[B. Package Diagram 13](#_Toc24579327)

[C. Deployment Diagram 13](#_Toc24579328)

[D. Class Diagram 13](#_Toc24579329)

[E. State Diagram 14](#_Toc24579330)

[F. Sequence Diagram 14](#_Toc24579331)

[G. ER Diagram 14](#_Toc24579332)

[2.3 Applied Tool 15](#_Toc24579333)

[A. Draw.io 15](#_Toc24579334)

[B. PowerPoint 15](#_Toc24579335)

[2.4 Project Scope 16](#_Toc24579336)

[3. System Architecture – Overall 17](#_Toc24579337)

[3.1 Objectives 17](#_Toc24579338)

[3.2 System Organization 17](#_Toc24579339)

[A. Overall System 17](#_Toc24579340)

[B. Frontend Application 19](#_Toc24579341)

[C. Backend Application 20](#_Toc24579342)

[4. System Architecture – Frontend 21](#_Toc24579343)

[4.1 Objectives 21](#_Toc24579344)

[4.2 Subcomponents 21](#_Toc24579345)

[A. Sign up 21](#_Toc24579346)

[A-1. Class Diagram 21](#_Toc24579347)

[1. Sign Up – 회원가입 객체 22](#_Toc24579348)

[A. attributes 22](#_Toc24579349)

[B. methods 22](#_Toc24579350)

[2. User – 유저 객체 22](#_Toc24579351)

[A. attribute 22](#_Toc24579352)

[A-2. Sequence Diagram 23](#_Toc24579353)

[B. Login 24](#_Toc24579354)

[B-1. Class Diagram 24](#_Toc24579355)

[1. LoginPage – 로그인 페이지 객체 24](#_Toc24579356)

[A. attribute 24](#_Toc24579357)

[B. methods 25](#_Toc24579358)

[2. User – 유저 객체 25](#_Toc24579359)

[A. attribute 25](#_Toc24579360)

[B-2 Sequence Diagram 26](#_Toc24579361)

[C. AddFood 27](#_Toc24579362)

[C-1 Class Diagram 27](#_Toc24579363)

[1. AddFoodPage – 음식 추가 페이지 객체 27](#_Toc24579364)

[A. attribute 27](#_Toc24579365)

[B. methods 27](#_Toc24579366)

[2. Food – 음식 객체 27](#_Toc24579367)

[A. attribute 27](#_Toc24579368)

[3. UserFood – 유저의 음식 정보 객체 28](#_Toc24579369)

[A. attribute 28](#_Toc24579370)

[C-2 Sequence Diagram 29](#_Toc24579371)

[D. Recommendation 30](#_Toc24579372)

[D-1 Class Diagram 30](#_Toc24579373)

[1. RecommendationPage – 추천 페이지 객체 30](#_Toc24579374)

[A. attribute 30](#_Toc24579375)

[B. methods 30](#_Toc24579376)

[2. Recommendation – 추천 객체 30](#_Toc24579377)

[A. attribute 30](#_Toc24579378)

[D-2 Sequence Diagram 31](#_Toc24579379)

[E. My page 32](#_Toc24579380)

[E-1 Class Diagram 32](#_Toc24579381)

[1. MyPagePage – 마이페이지 페이지 객체 32](#_Toc24579382)

[A. attribute 32](#_Toc24579383)

[B. methods 32](#_Toc24579384)

[2. User – 유저 객체 33](#_Toc24579385)

[A. attribute 33](#_Toc24579386)

[E-2 Sequence Diagram 33](#_Toc24579387)

[F. Order 34](#_Toc24579388)

[F-1 Class Diagram 34](#_Toc24579389)

[1. OrderPage – 주문 페이지 객체 34](#_Toc24579390)

[A. attribute 34](#_Toc24579391)

[B. methods 34](#_Toc24579392)

[2. Product – 물품 객체 34](#_Toc24579393)

[A. attribute 34](#_Toc24579394)

[F-2 Sequence Diagram 35](#_Toc24579395)

[G. Seller Sign Up 36](#_Toc24579396)

[G-1 Class Diagram 36](#_Toc24579397)

[1. SellerSignUpPage – 판매자 회원가입 페이지 객체 36](#_Toc24579398)

[A. attribute 36](#_Toc24579399)

[B. methods 36](#_Toc24579400)

[2. Seller – 판매자 객체 36](#_Toc24579401)

[A. attribute 36](#_Toc24579402)

[G-2 Sequence Diagram 37](#_Toc24579403)

[H. Seller Login 38](#_Toc24579404)

[H-1 Class Diagram 38](#_Toc24579405)

[1. SellerLoginPage – 판매자 로그인 페이지 객체 38](#_Toc24579406)

[A. attribute 38](#_Toc24579407)

[B. methods 38](#_Toc24579408)

[2. Seller – 판매자 객체 38](#_Toc24579409)

[A. attribute 38](#_Toc24579410)

[H-2 Sequence Diagram 39](#_Toc24579411)

[I. AddProduct 40](#_Toc24579412)

[I-1 Class Diagram 40](#_Toc24579413)

[1. AddProductPage – 물품 추가 페이지 객체 40](#_Toc24579414)

[A. attribute 40](#_Toc24579415)

[B. method 41](#_Toc24579416)

[2. Product – 물품 객체 41](#_Toc24579417)

[A. attribute 41](#_Toc24579418)

[I-2 Sequence Diagram 41](#_Toc24579419)

[J. Product Management 42](#_Toc24579420)

[J-1 Class Diagram 42](#_Toc24579421)

[1. ProductManagementPage – 물품 관지 페이지 객체 42](#_Toc24579422)

[A. attribute 42](#_Toc24579423)

[2. Product – 물품 객체 43](#_Toc24579424)

[A. attribute 43](#_Toc24579425)

[J-2 Sequence Diagram 43](#_Toc24579426)

[5. System Architecture – Backend 44](#_Toc24579427)

[5.1 Objectives 44](#_Toc24579428)

[5.2 Overall Architecture 44](#_Toc24579429)

[5.3 Subcomponents 45](#_Toc24579430)

[A. ImageClassificationSystem 45](#_Toc24579431)

[A-1 Class Diagram 45](#_Toc24579432)

[A-2 Sequence Diagram 46](#_Toc24579433)

[B. Nutrient State Analysis System 47](#_Toc24579434)

[B-1 Class Diagram 47](#_Toc24579435)

[B-2 Sequence Diagram 47](#_Toc24579436)

[C. Recommendation System 48](#_Toc24579437)

[C-1 Class Diagram 48](#_Toc24579438)

[C-2 Sequence Diagram 49](#_Toc24579439)

[D. Token Distribution System 50](#_Toc24579440)

[D-1 Class Diagram 50](#_Toc24579441)

[D-2 Sequence Diagram 51](#_Toc24579442)

[6. Protocol Design 52](#_Toc24579443)

[6.1 Objectives 52](#_Toc24579444)

[6.2. REST API 52](#_Toc24579445)

[6.3. JSON 53](#_Toc24579446)

[6.4. Details 54](#_Toc24579447)

[**A.** Authentication 54](#_Toc24579448)

[**B.** User 56](#_Toc24579449)

[**D.** Food 59](#_Toc24579450)

[**E.** Product 60](#_Toc24579451)

[**F.** Recommendation 63](#_Toc24579452)

[**G.** Seller 64](#_Toc24579453)

[7. Database Design 65](#_Toc24579454)

[7.1. Objectives 65](#_Toc24579455)

[7.2. Entity-Relation Diagram 66](#_Toc24579456)

[A. Entity 66](#_Toc24579457)

[b. User Food 67](#_Toc24579458)

[B. Relational Schema 69](#_Toc24579459)

[7.3. SQL DDL 71](#_Toc24579460)

[A. User 71](#_Toc24579461)

[B. User Food 71](#_Toc24579462)

[C. Seller 71](#_Toc24579463)

[D. Product 73](#_Toc24579464)

[E. Food 73](#_Toc24579465)

[F. Recommendation 73](#_Toc24579466)

[8. Testing Plan 74](#_Toc24579467)

[8.1 Objectives 74](#_Toc24579468)

[8.2 Testing Policy 74](#_Toc24579469)

[A. Development Testing 74](#_Toc24579470)

[B. Release Testing 75](#_Toc24579471)

[C. User Testing 75](#_Toc24579472)

[D. Testing Case 76](#_Toc24579473)

[9. Development Plan 77](#_Toc24579474)

[9.1 Objectives 77](#_Toc24579475)

[9.2 Frontend Environment 77](#_Toc24579476)

[A. Android 77](#_Toc24579477)

[B. React 77](#_Toc24579478)

[9.3 Backend Environment 78](#_Toc24579479)

[A. MySQL 78](#_Toc24579480)

[B. Node.js 78](#_Toc24579481)

[C. Klaytn 78](#_Toc24579482)

[D. Flask 79](#_Toc24579483)

[E. Google Cloud Vision API 79](#_Toc24579484)

[9.4 Schedule 79](#_Toc24579485)

[10. INDEX 81](#_Toc24579486)

[10.1 Table 81](#_Toc24579487)

[10.2 Figures 82](#_Toc24579488)

[10.3 Diagrams 83](#_Toc24579489)

# Preface

## 1.1. Objectives

preface에서는 해당 명세서의 대상 독자층을 정의하고, 각 챕터 별 내용을 간략하게 소개하고 있다. 또한, 이 문서는 버전의 변화와, 각 버전 사이의 변화를 요악한 내용을 포함하고 있다.

## 1.2. Readership

해당 명세서의 대상 독자층은 시스템을 직접 개발하는 소프트웨어 엔지니어, 시스템을 설계하는 아키텍처, 그 외 개발에 참여하는 모든 구성원을 독자로 정의한다. 다시 말해, 본 문서의 독자는 본 문서에서 소개하는 시스템의 개발 및 유지 보수에 관련된 모든 구성원이다.

## 1.3. Document Structure

1. Introduction

Introduction에서는 본 문서를 서술하는데 사용된 모든 종류의 다이어그램과 도구들에 대해 설명하고, 본 프로젝트가 다루는 시스템의 범위에 대해 서술한다.

1. System Architecture

System Architecture에서는 팀에서 개발하고자 하는 목표 시스템에 대한 고수준의 개요를 제시한다. 시스템과 각 서브시스템의 구조를 개괄적으로 기술하고, 시스템의 전체 기능이 각 서브시스템과 하드웨어에 어떻게 할당 되었는지를 설명한다.

1. Protocol Design

Protocol Design에서는 시스템의 각 구성요소, 특히 프론트엔드 시스템과 백엔드 시스템간의 상호작용을 규정하는 인터페이스와 프로토콜을 어떻게 구성하는지에 대해 기술하고, 해당 인터페이스가 어떤 기술에 기반해 있는지 설명한다.

1. Database Design

Database Design은 Requirement Specification에서 기술한 데이터베이스 요구사항을 바탕으로 수정사항을 반영하여 요구사항을 다시 작성하였다. 수정한 데이터베이스 요구사항을 바탕으로 ER Diagram을 통해 각 데이터 엔티티의 속성과 관계를 표현하고, Relational Schema, SQL DDL을 작성한다.

1. Testing Plan

Testing Plan에서는 Test Policy와 Test Case에 대해 설명한다. 해당 챕터의 목적은 전체 시스템이 의도한대로 실행되는지를 확인, 검증(verification, validation)하기 위한 과정을 사전에 계획하는 데에 있다.

1. Development Plan

Development Plan에서는 개발 계획에 대해 서술한다. 시스템을 구현하는 데 필요한 개발 도구와 프로그래밍 언어, 라이브러리 등의 개발 환경에 대해 설명하고, 도표를 이용해 시스템 개발 일정을 기술한다.

1. Index

본 문서에서 사용된 그림, 표, 다이어그램 등의 색인을 기술한다.

# Introduction

## 2.1. Objectives

Introduction에서는 본 문서를 서술하는데 사용된 모든 종류의 다이어그램과 도구들에 대해 설명하고, 본 프로젝트가 다루는 시스템의 범위에 대해 서술한다.

## Applied Diagram

1. UML

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

UML은 요구분석, 시스템 설계, 시스템 구현 등의 과정에서 생길 수 있는 개발자 간의 의사 소통 불일치를 해소할 수 있다는 장점 덕분에, 현재 객체지향 시스템 개발 분야에서 가장 우수한 모델링 언어로 인식되고 있다. UML은 개발자가 구축하고자 하는 소프트웨어 시스템을 구현하기에 앞서서 표준화되고 이해가 쉬운 방식으로 설계되어, 소프트웨어에 관련된 모든 사람들과 상호 소통을 가능하게 하는 효율적인 매개체 역할을 한다. 이 때문에 생략되거나 불일치한 모델링 구조에 대한 지적도 용이하고, 그 스스로 모델링에 대한 표현력이 강하고 비교적 모순이 적은 논리적인 표기법을 가진 언어라는 장점이 있다.

UML은 여러 가지 다른 모델링 기술과 다이어그램을 포함하고 있다. 다양한 범위의 기호와 정의를 포함하고 있기 때문에 개발자와 유저 사이의 효율적인 의사소통 수단이 된다. 본 명세서에서는 UML에 포함된 다이어그램들 중 다음의 다이어그램들을 사용하고 있다: Package Diagram, Deployment Diagram, Class Diagram, State Diagram, Sequence Diagram, ER Diagram.

1. Package Diagram

Package Diagram은 모델 구성 요소들의 배열과 구성을 나타내는 structural diagram들 중 하나다. Package Diagram은 다른 UML 다아어그램들보다 더 추상적인 방법으로 서브 시스템들간의, 모듈간의 구조와 의존성을 모두 보여준다. 이러한 추상성이 복잡한 class diagram을 패키지로 그룹핑 함으로써 간단하게 만들어준다.

1. Deployment Diagram

Deployment Diagram은 소프트웨어 프로그램이 하드웨어 구성요소에 할당된 물리적 배치 상태를 나타낸다. 즉, 시스템이 실행되는 환경인 노드와 그 노드에 배치된 컴포넌트의 구성, 각 노드들 사이의 네트워크 특성이나 프로토콜과 같은 관계를 나타난다.

직육면체로 나타내지는 노드는 처리 능력을 가진 소프트웨어 장치, 하드웨어 장치들을 가리킨다. 노드와 노드를 연결하는 선은 둘의 관계를 나타내며, 박스 안에 포함된 작은 모형은 배치된 소프트웨어 아티팩트를 나타낸다.

1. Class Diagram

Class Diagram은 시스템의 오브젝트 클래스들과 클래스들 간의 관계를 나타내는데 사용된다. Class Diagram을 사용하는 가장 근본적인 이류는 클래스들간의 계층과 의존성을 명확하게 특정 지어 나타낼 수 있기 때문이다.

각 클래스 안에는 해당 클래스가 포함하고 있는 변수, 메소드와 클래스들 간의 링크 또는 연관성을 포함하고 있다.

1. State Diagram

State Diagram은 시스템의 여러 다른 상태들을 표현하는 기술이다. 상태의 변화를 촉발하는 자극(stimuli)을 바탕으로 각 상태가 가질 수 있는 모든 가능한 다음 상태를 나타낸다.

이는 클래스 다이어그램에서 정의된 오브젝트 클래스의 행동을 식별하는데 도움을 주기 때문에 시스템을 사용하면서 나타날 수 있는 시나리오를 분석하는데 아주 중요하다.

1. Sequence Diagram

Sequence Diagram은 시스템에 포함된 액터와 오브젝트들 간의 상호 작용을 나타낸다. 더 자세하게 보면, 이 다이어그램의 목적은 특정 use case에서 각 요소간의 상호작용 순서와 단계, 각 단계에서 만들어지는 결과물을 나타내는데 있다. 화살표 방향이 올바른 액션 흐름을 나타내는데 필수적이다.

1. ER Diagram

ER Diagram은 엔티티가 가지고 있는 속성과 엔티티 간의 관계를 나타낸 다이어그램이다. 이 다이어그램은 주로 데이터베이스를 설계하는 데 사용되며, 해당 다이어그램을 기반으로 Relational Schema, SQL DDL을 작성하게 된다.

엔티티는 직사각형으로, 속성은 타원형으로, 각 엔티티간의 관계는 마름모로 나타낸다.

## Applied Tool

1. Draw.io

Draw.io는 온라인 모델링 툴로서 많은 기본 템플릿과 도형을 제공하기 때문에 사용자가 편리하게 다이어그램을 만들 수 있다. 각 구성요소를 만들어진 도형들을 활용해 나타내고, 도형 간 연결선 중 필요한 것을 사용해 관계를 나타낼 수 있다. 또한, 격자 줄이 있어 도형을 정렬하기 편리하다. 해당 문서에서 사용된 대부분의 다이어그램은 본 도구로 작성되었다.

1. PowerPoint

PowerPoint는 본래 그래픽 프레젠테이션 툴로 주로 발표용으로 사용된다. 파워포인트 안에 내장된 도형 기능으로 draw i.o에서 만들기 어려운 복잡한 다이어그램들과 그래픽이 포함된 figure를 작성하기 위해 사용하였다.

## Project Scope

본 시스템은 먹은 음식을 일일이 검색해 등록해야 하며, 영양소 분석 기능만 해주는 기존 영양소 관리 시스템들의 단점을 극복하기 위해, 사진을 통한 간편한 음식 등록과 부족한 영양소를 포함하고 있는 음식의 추천, 해당 음식의 구매와의 간단한 연결을 도와주는 시스템이다. 본 시스템의 핵심 기능은 섭취한 음식의 영양소 분석 및 이를 바탕으로 한 음식 추천 기능이며, 해당 기능을 중심으로 각 서브시스템들이 상호작용하도록 설계하였다.

먼저 Front-end System은 시스템과 사용자와의 상호작용을 담당하며, Back-end System은 Front-end System에서 오는 데이터 요청에 응답하고, 음식 사진 분석을 통한 후보 추출, 섭취한 영양소 분석, 추천 음식 목록 생성을 실행시키는 역할을 담당한다. 사용자가 카메라로 음식 사진을 촬영하면 image classification을 실행해 음식을 분석하고 음식 후보를 추출한다. 사용자가 먹은 음식을 등록하면, 저장된 DB에서 해당 음식의 영양 성분을 가져오고 사용자가 등록한 음식 정보를 바탕으로 영양상태를 분석한다. 또한, 분석된 영양 상태를 그래픽을 통해 보여주며 결핍된 영양소를 포함한 음식의 추천과, 사용자의 요청 시 해당 음식 판매자와 연결을 시행한다.

# System Architecture – Overall

## Objectives

해당 항목에서는 Balanced One 시스템의 전체적인 구조를 기술한다. 시스템 전체의 구조와 각 서브시스템의 개략적 구조 및 서브시스템 간의 관계를 서술한다.

## System Organization

1. Overall System

스크린샷이(가) 표시된 사진

자동 생성된 설명

Diagrams Frontend Overall System Diagram

본 시스템은 클라이언트-서버 모델을 적용하여 설계하였다. Frontend에 해당하는 Android Application에서 클라이언트와 상호작용하고, Backend에 해당하는 Node.js Application에서 Frontend로부터의 요청을 각 컨트롤러로 분배하고, 필요한 정보를 데이터베이스로부터 받아와 다시 Frontend로 전달한다. Frontend와 Backend간 통신은 HTTP로 이루어진다.

Frontend에서 Image Classification 요청을 보내면 Backend 컨트롤러에서 Image Classification System을 호출한다. Image Classification System에서는 Google Cloud Vision Image Classification Module에 접속하여 결과를 받아 전달한다.

사용자가 Nutrient State를 업데이트하면 Nutrient State Analysis System에서 해당 사용자의 Nutrient State가 Healthy한지 Unhealthy한지 여부를 분석한다. 분석한 결과는 Recommendation System과 Token Distribution System으로 전달된다.

Recommendation System에서는 Nutrient State Analysis System으로부터 전달받은 분석 결과에 따라 사용자에게 필요한 영양소를 가지고 있는 음식을 추천한다.

Token Distribution System에서는 Nutrient State Analysis System으로부터 전달받은 분석 결과에 따라 사용자가 구매에 사용할 수 있는 ERC20 Klaytn Token을 분배한다.

1. Frontend Application

스크린샷이(가) 표시된 사진

자동 생성된 설명

Diagrams Frontend System Diagram

전체적 시스템에서 클라이언트 부분을 담당하며, 사용자와 상호작용하여 필요한 요청을 Backend로 전송한다. 기본적인 Android의 MVVM 모델을 적용하였다. Sign Up, Login, Add Food, Recommendation, My Page, Order, Seller Sign Up, Seller Login, Add Product, Product Management 컴포넌트가 존재한다.

1. Backend Application

텍스트이(가) 표시된 사진

자동 생성된 설명

Diagrams Backend System Diagram

# System Architecture – Frontend

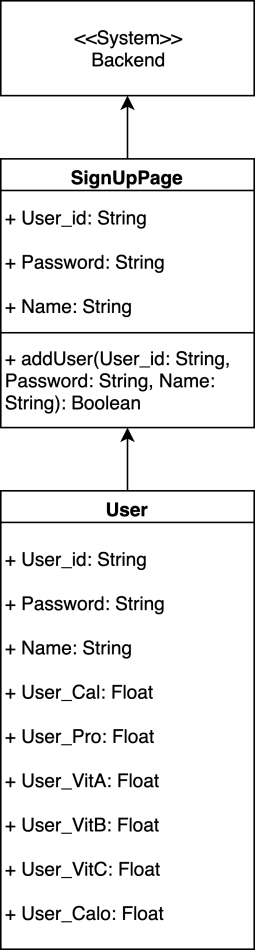
## Objectives

Frontend System의 구조에 관해 서술한다.

## Subcomponents

1. Sign up

A-1. Class Diagram



Diagrams 4 Subcomponent Sign Up Class Diagram

### 1. Sign Up – 회원가입 객체

### A. attributes

+ User\_id : 유저를 식별하는데 사용하는 고유 아이디

+ Password : 유저를 확인하는데 사용하는 암호, 아이디와 쌍으로 사용자 본인 확인을 하는데 사용.

### B. methods

+ addUser(User\_id: String, Password: String, Name: String) : 새로운 유저의 아이디 비밀번호를 입력 받아 데이터 베이스에 추가한다.

### 2. User – 유저 객체

### A. attribute

+ User\_id : 유저 식별 고유 코드

+ Password : 유저 식별 암호

+ Name : 유저의 이름

+ User\_Cal : 유저가 섭취한 칼로리 양

+ User\_Pro : 유저가 섭취한 단백질의 양

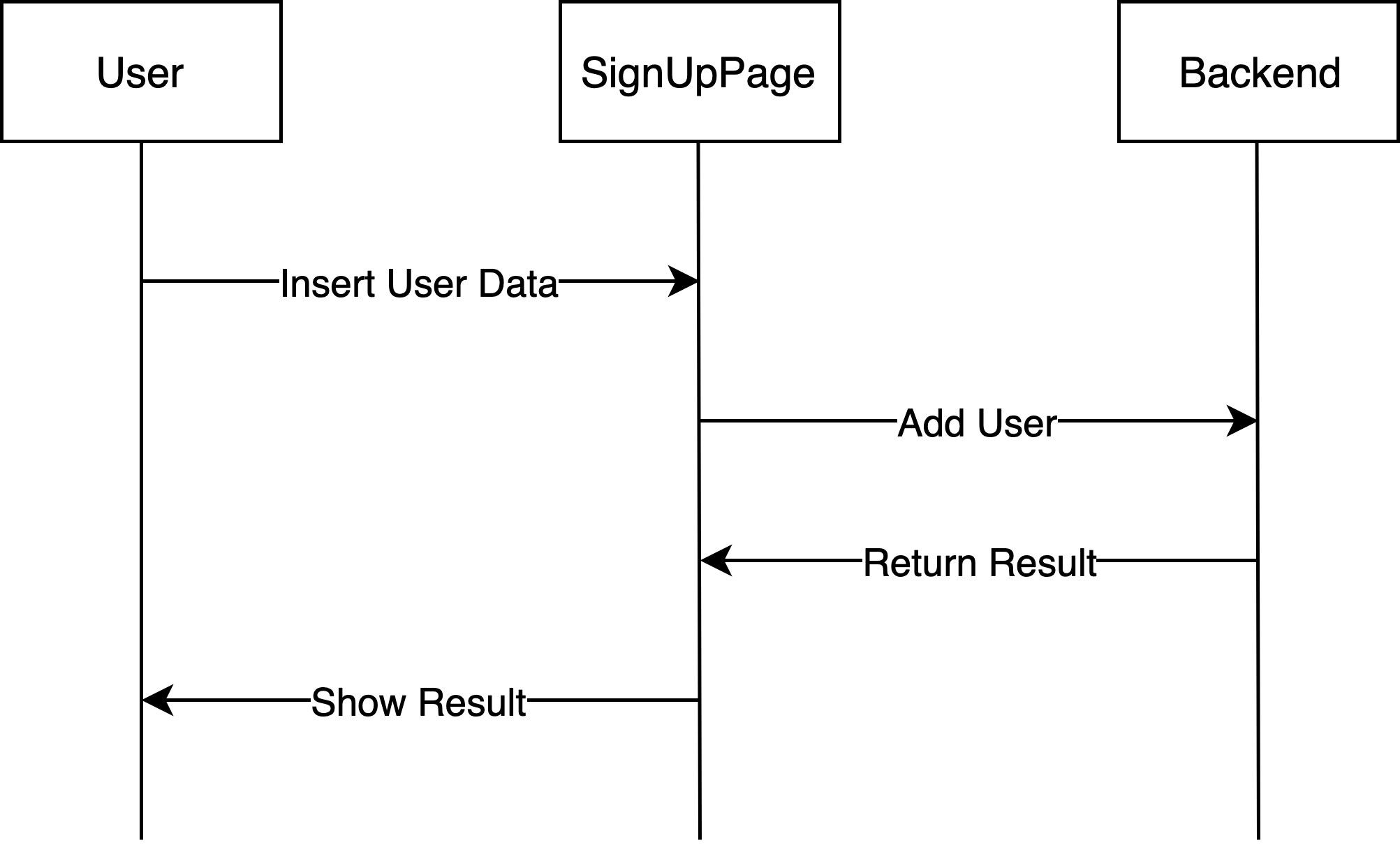
+ User\_VitA : 유저가 섭취한 비타민A의 양

+ User\_VitB : 유저가 섭취한 비타민B의 양

+ User\_VitC : 유저가 섭취한 비타민C의 양

+ User\_Calo : 유저가 섭취한 철분의 양

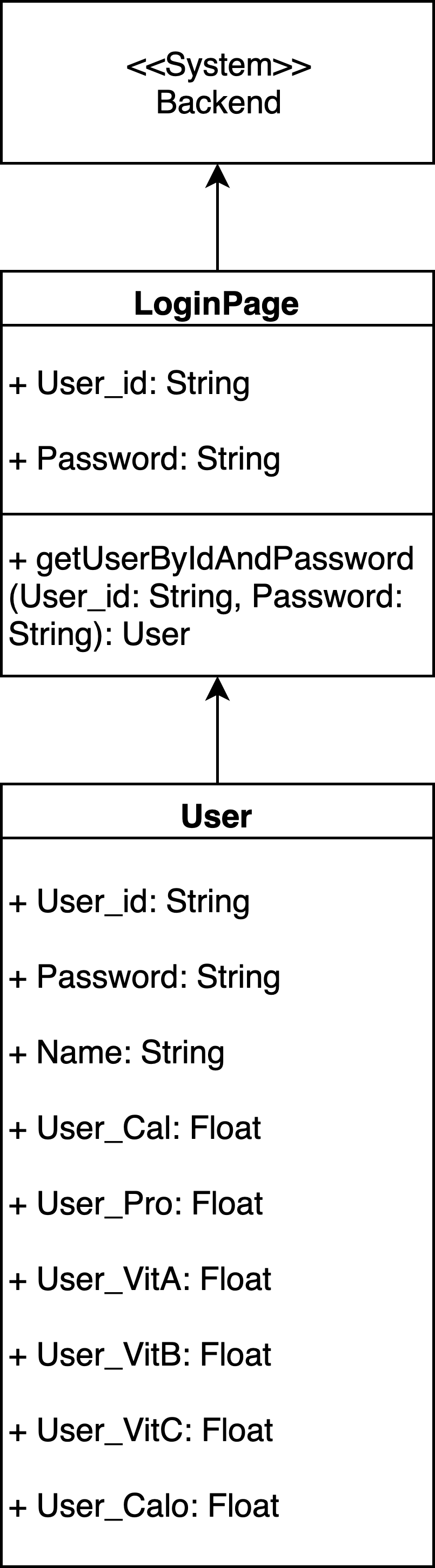
### A-2. Sequence Diagram



Diagrams 5 Subcomponent Sign Up Sequence Diagram

### B. Login

### B-1. Class Diagram



Diagrams Subcomponent Login Class Diagram

### 1. LoginPage – 로그인 페이지 객체

### A. attribute

+ User\_id : 유저 식별용 고유 아이디

+ Password : 유저가 설정한 암호

### B. methods

+ getUserByIdAndPassword : 유저의 아이디와 비밀번호를 입력 받아 일치하는 유저의 정보를 가져옴

### 2. User – 유저 객체

### A. attribute

+ User\_id : 유저 식별 고유 코드

+ Password : 유저 식별 암호

+ Name : 유저의 이름

+ User\_Cal : 유저가 섭취한 칼로리 양

+ User\_Pro : 유저가 섭취한 단백질의 양

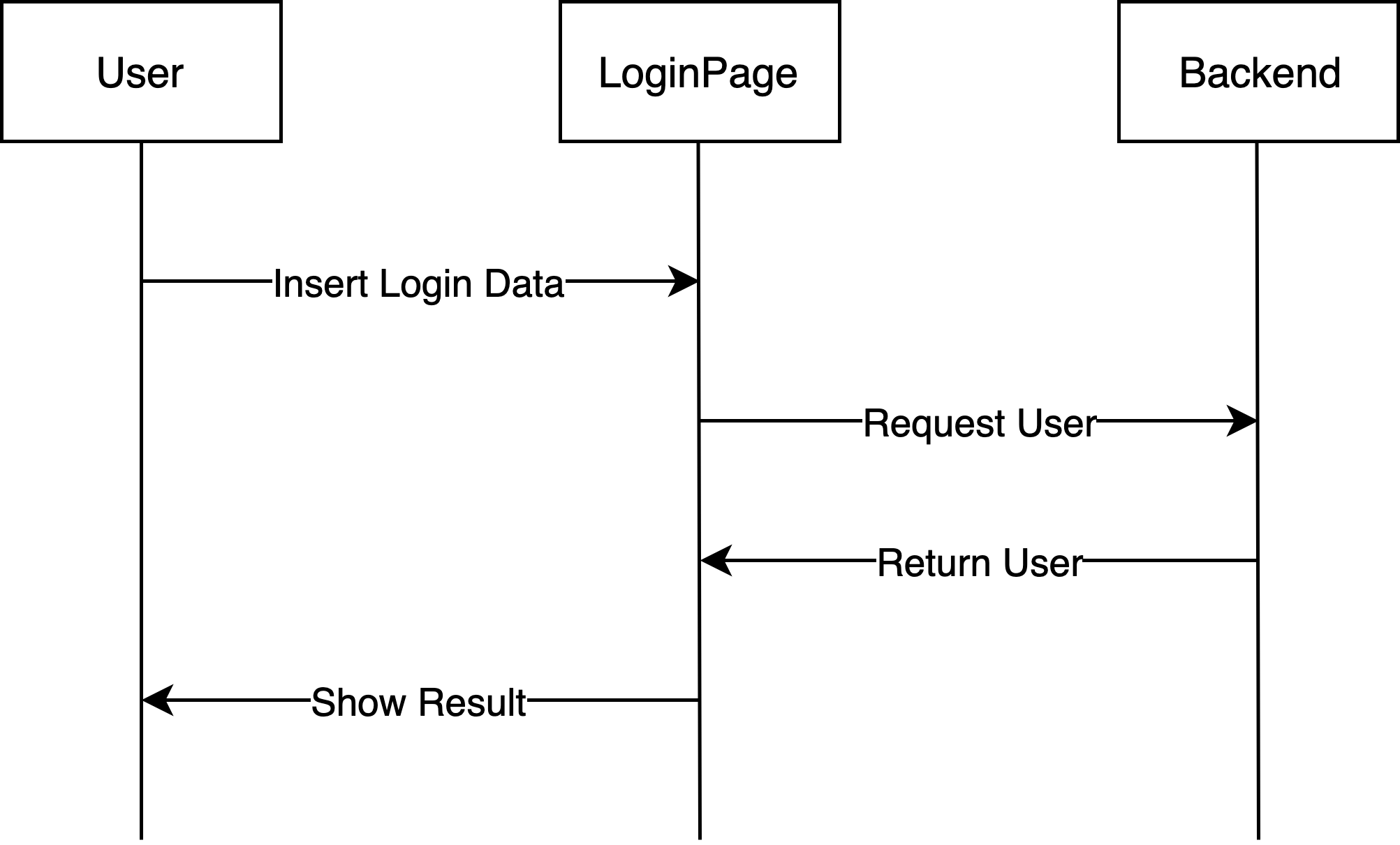
+ User\_VitA : 유저가 섭취한 비타민A의 양

+ User\_VitB : 유저가 섭취한 비타민B의 양

+ User\_VitC : 유저가 섭취한 비타민C의 양

+ User\_Calo : 유저가 섭취한 철분의 양

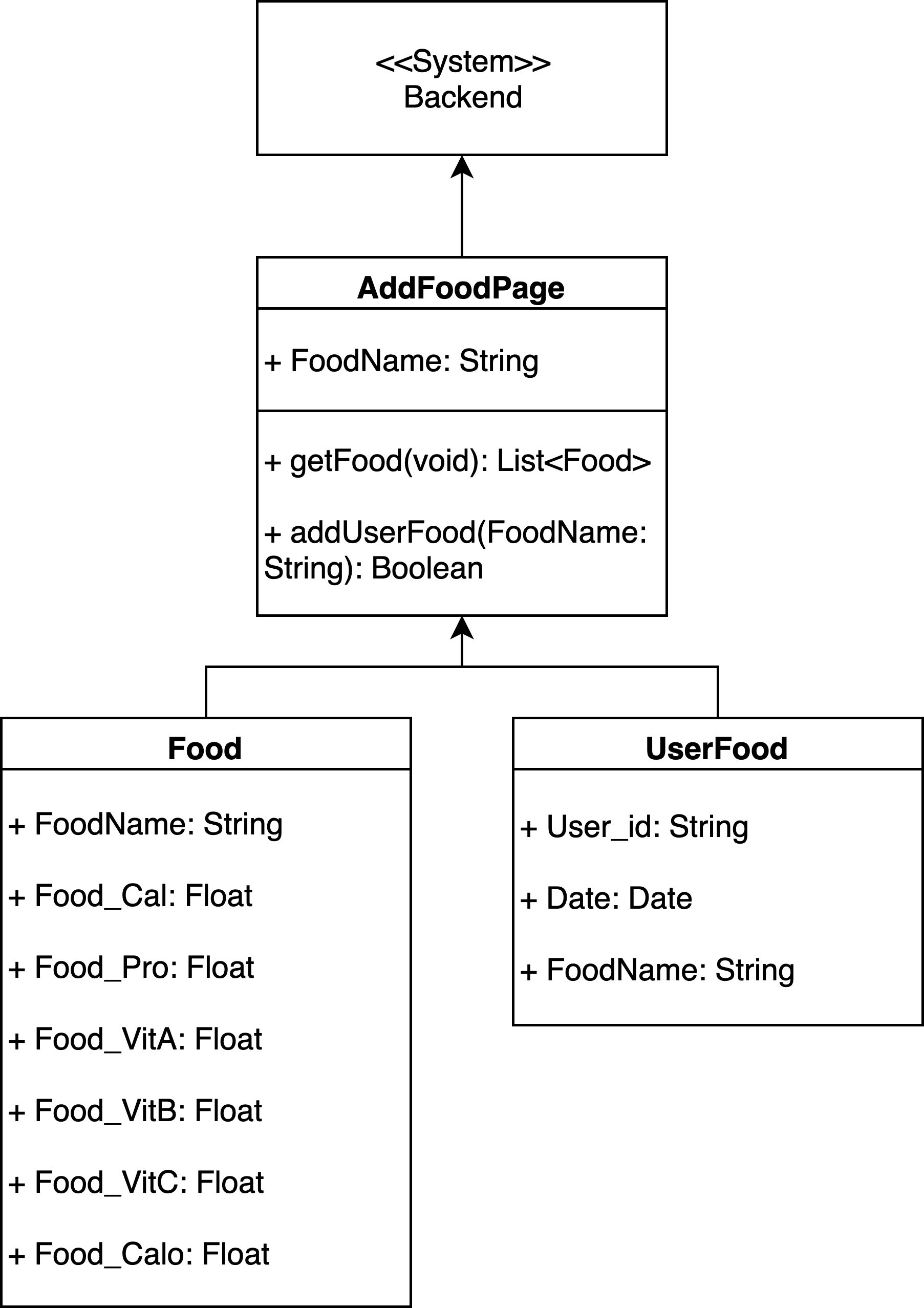
### B-2 Sequence Diagram



Diagrams Subcomponent Login Sequence Diagram

### C. AddFood

### C-1 Class Diagram



Diagrams 8 Subcomponent Add Food Class Diagram

### 1. AddFoodPage – 음식 추가 페이지 객체

### A. attribute

+ FoodName : 추가할 음식의 이름

### B. methods

+ getFood(void) : 촬영된 음식의 이름 후보 리스트 생성

+ addUserFood(FoodName) : 선택된 음식을 UserFood객체에 추가

### 2. Food – 음식 객체

### A. attribute

+ FoodName : 음식의 이름

+ Food\_Cal : 음식의 칼로리

+ Food\_Pro : 음식의 단백질 함유량

+ Food\_VitA : 음식의 비타민A 함유량

+ Food\_VitB : 음식의 비타민B 함유량

+ Food\_VitC : 음식의 비타민C 함유량

+ Food\_Calo : 음식의 철분 함유량

### 3. UserFood – 유저의 음식 정보 객체

### A. attribute

+ User\_id : 유저 식별 고유 문자

+ Date : 유저가 음식을 섭취한 날짜

+ FoodName : 유저가 먹은 음식 이름

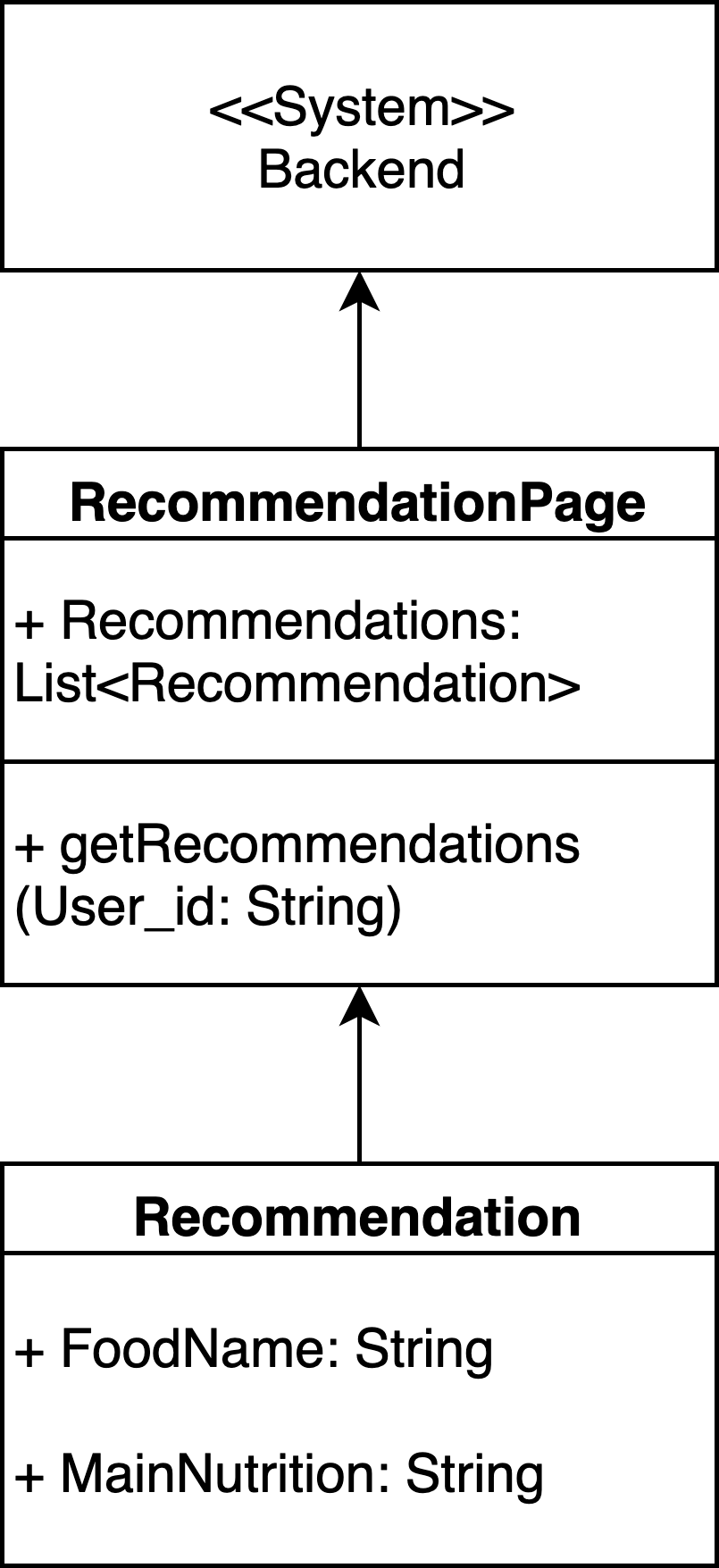
### C-2 Sequence Diagram



Diagrams 9 Subcomponent Add Food Sequence Diagram

### D. Recommendation

### D-1 Class Diagram



Diagrams Subcomponent Recommendation Class Diagram

### 1. RecommendationPage – 추천 페이지 객체

### A. attribute

+ Recommendations : 추천 음식 목록 리스트

### B. methods

+ getRecommendations : 추천 음식 목록 생성

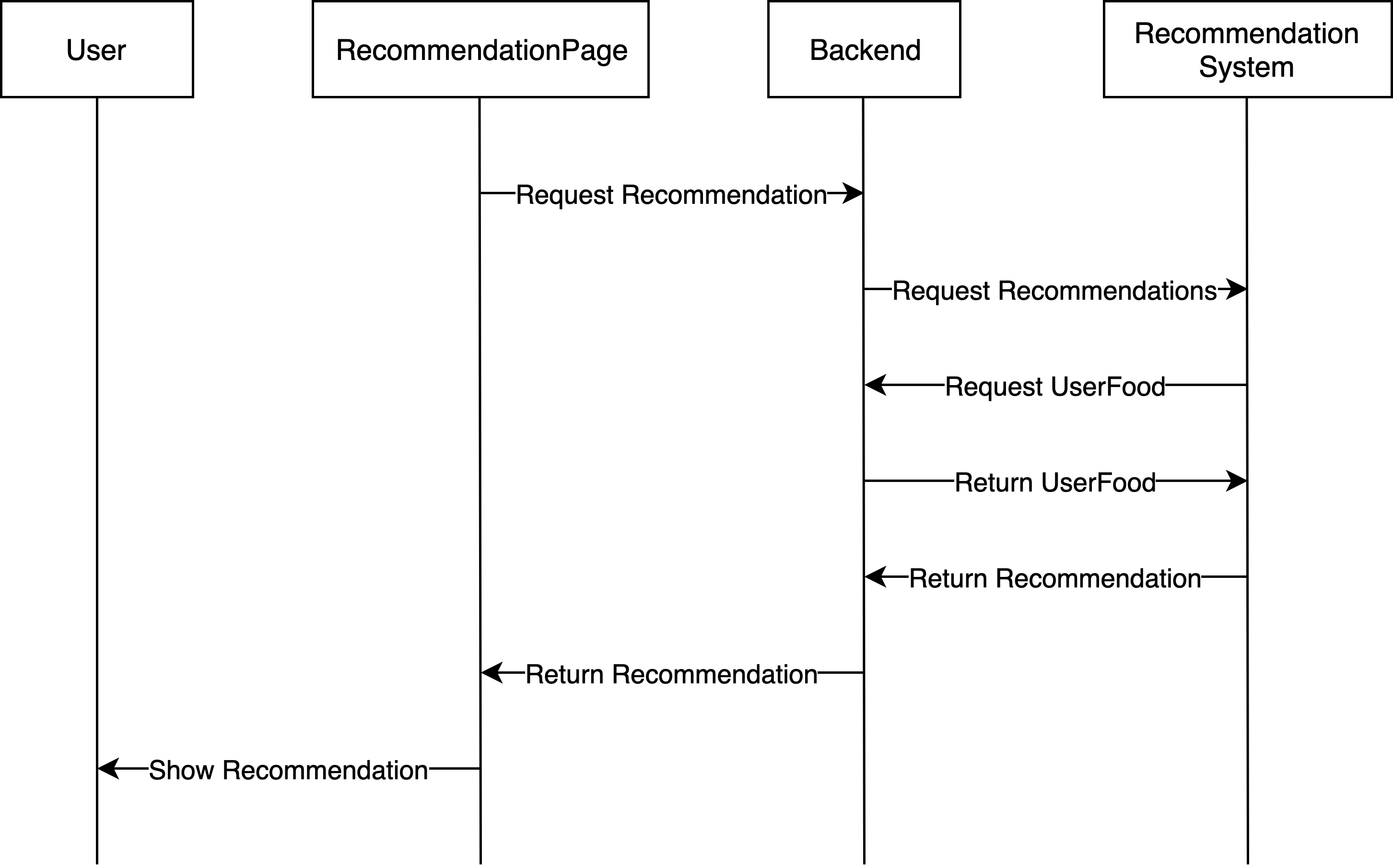
### 2. Recommendation – 추천 객체

### A. attribute

+ FoodName : 음식 이름

+ MainNutrition : 해당 음식의 대표 함유 영양소 이름

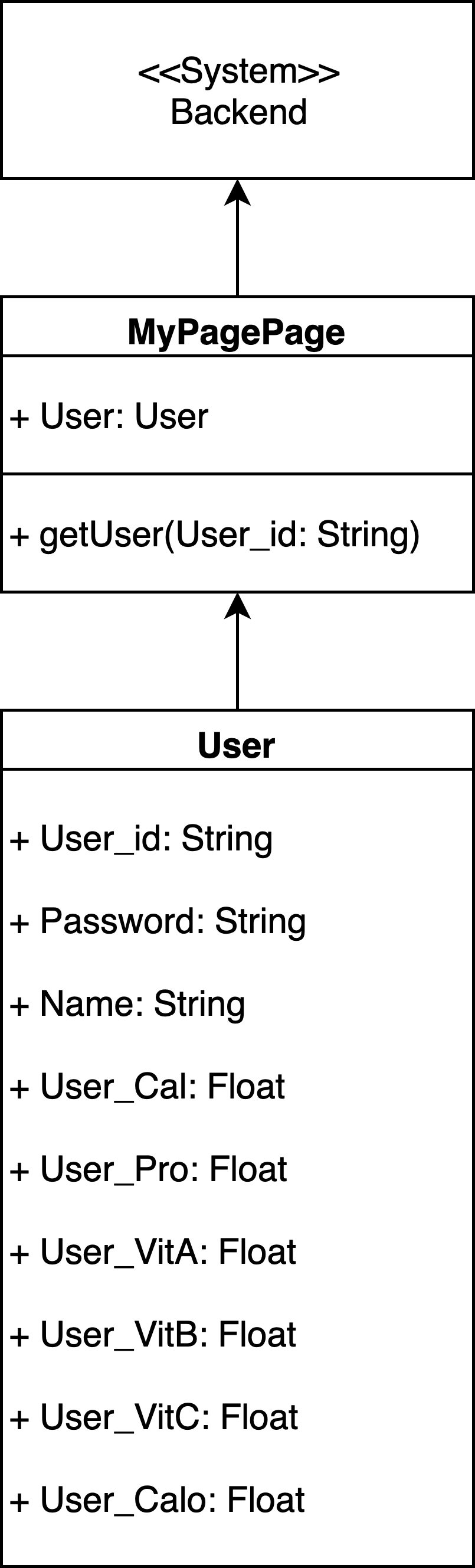
### D-2 Sequence Diagram



Diagrams Subcomponent Recommendation Sequence Diagram

### E. My page

### E-1 Class Diagram



Diagrams Subcomponent My Page Class Diagram

### 1. MyPagePage – 마이페이지 페이지 객체

### A. attribute

+ User : 유저 객체

### B. methods

+ getUser : 해당 유저의 유저 객체를 가져옴

### 2. User – 유저 객체

### A. attribute

+ User\_id : 유저 식별 고유 코드

+ Password : 유저 식별 암호

+ Name : 유저의 이름

+ User\_Cal : 유저가 섭취한 칼로리 양

+ User\_Pro : 유저가 섭취한 단백질의 양

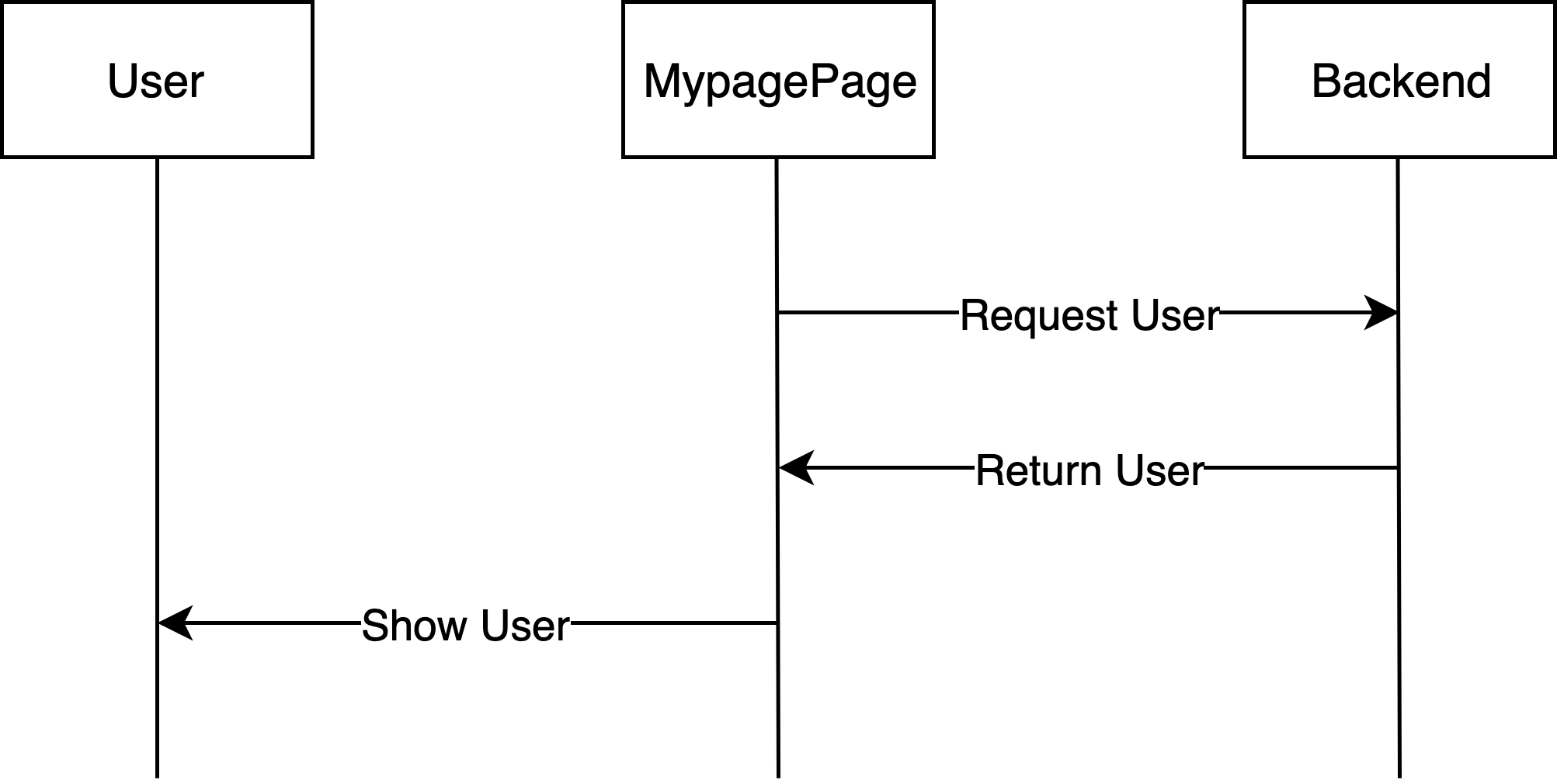
+ User\_VitA : 유저가 섭취한 비타민A의 양

+ User\_VitB : 유저가 섭취한 비타민B의 양

+ User\_VitC : 유저가 섭취한 비타민C의 양

+ User\_Calo : 유저가 섭취한 철분의 양

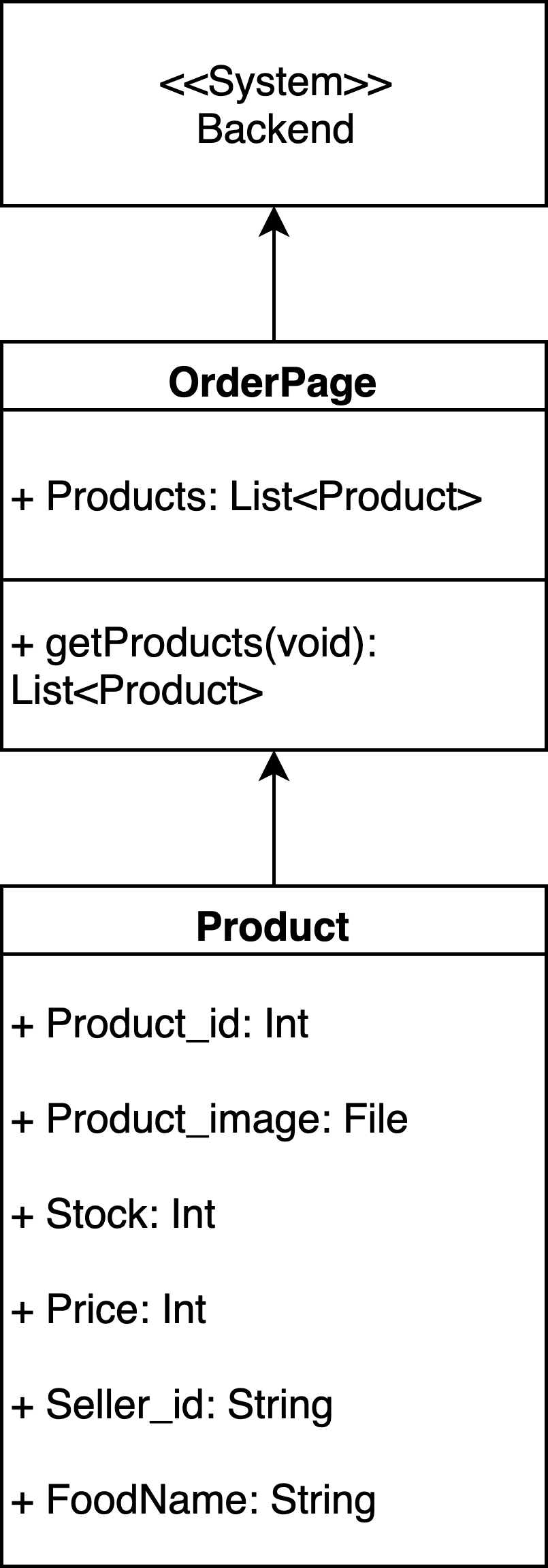
### E-2 Sequence Diagram



Diagrams Subcomponent My Page Sequence Diagram

### F. Order

### F-1 Class Diagram



Diagrams Subcomponent Order Class Diagram

### 1. OrderPage – 주문 페이지 객체

### A. attribute

+ Products : 구입 음식의 리스트

### B. methods

+ getProducts(void) : 구매 상품 리스트 가져오기

### 2. Product – 물품 객체

### A. attribute

+ Product\_id : 물품의 아이디

+ Product\_image : 물품의 사진

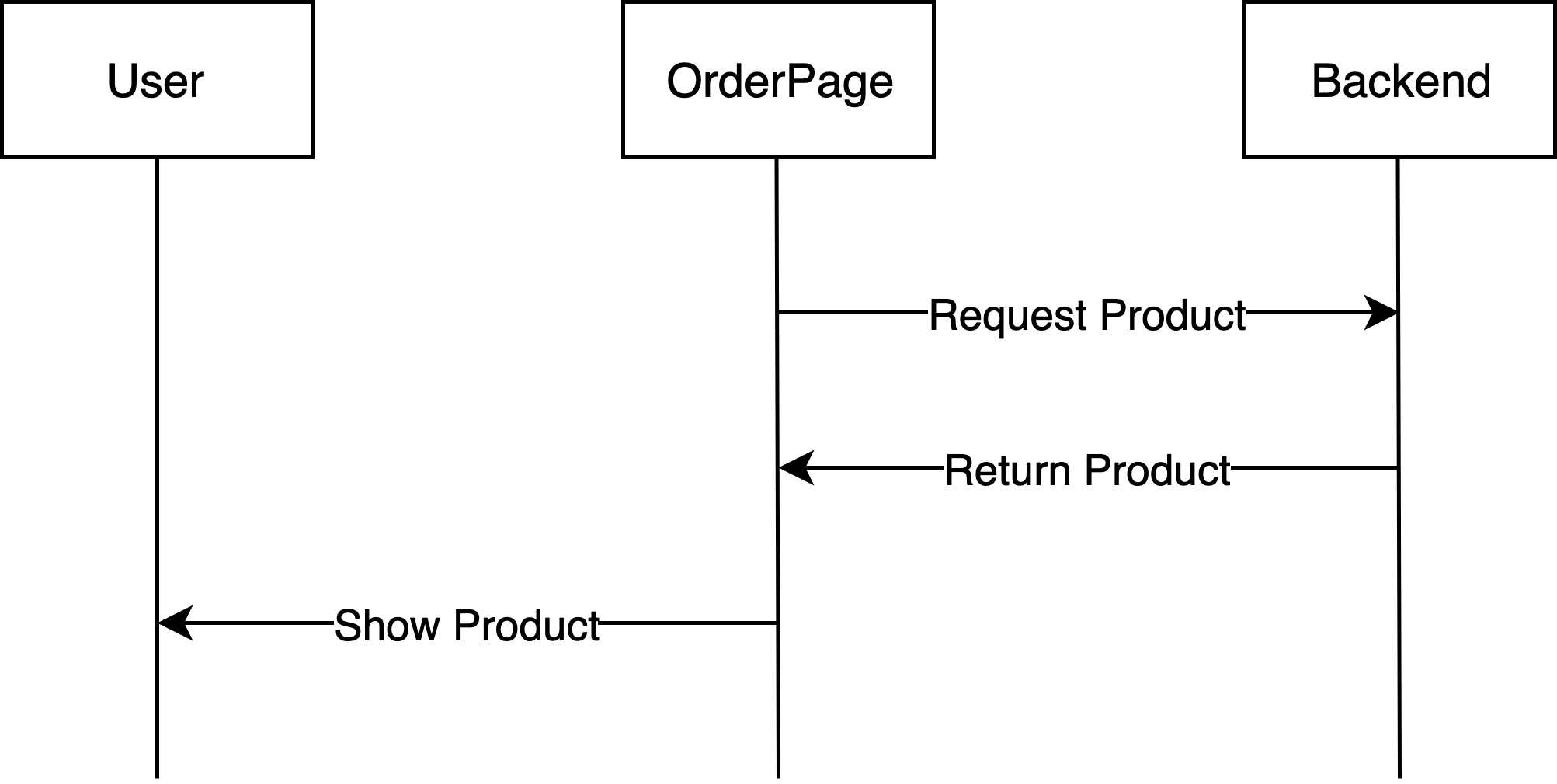
+ Stock : 해당 물품의 재고

+ Price : 해당 물품의 가격

+ Seller\_id : 해당 물품 판매자의 아이디

+ FoodName : 음식의 이름

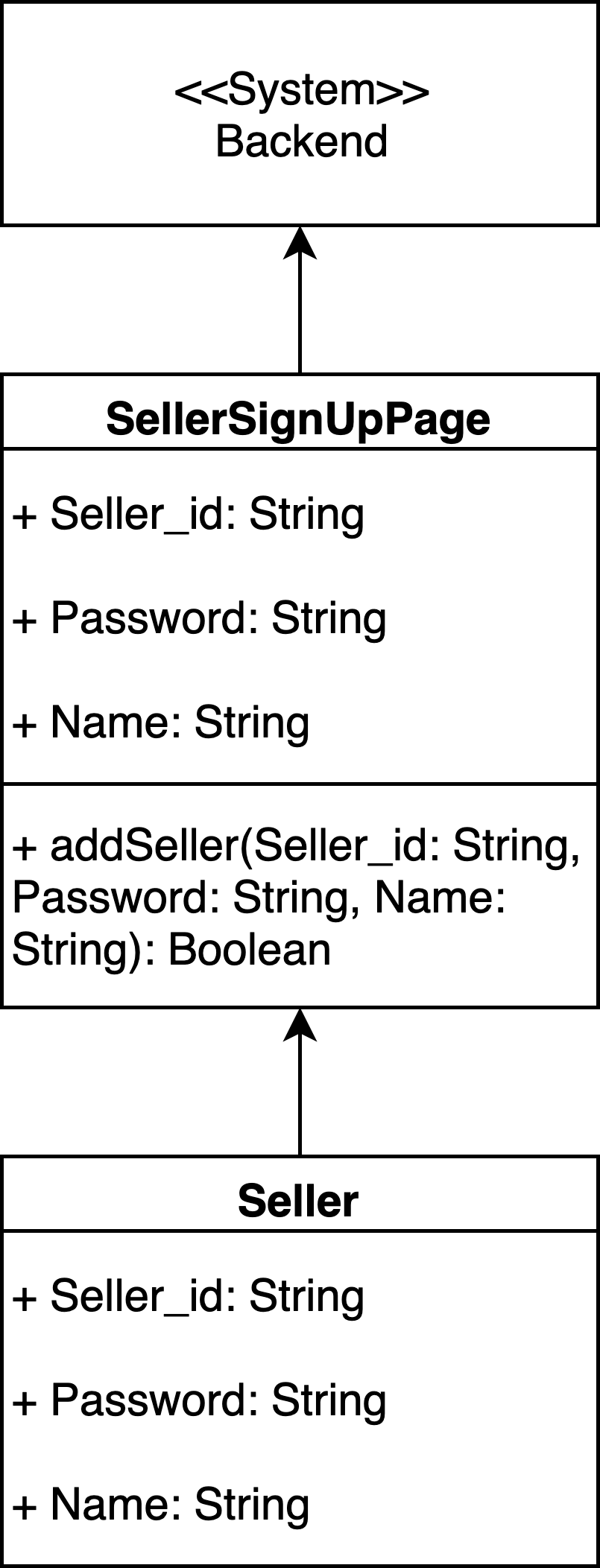
### F-2 Sequence Diagram



Diagrams Subcomponent Order Sequence Diagram

### G. Seller Sign Up

### G-1 Class Diagram



Diagrams 16 Subcomponent Seller Sign Up Class Diagram

### 1. SellerSignUpPage – 판매자 회원가입 페이지 객체

### A. attribute

+ Seller\_id : 판매자 고유 문자, 아이디

+ Password : 판매자가 설정한 비밀번호

+ Name : 판매자 이름

### B. methods

+ addSeller(Seller\_id, Password) : 셀러가 입력한 고유 아이디와 비밀번호를 데이터베이스에 추가

### 2. Seller – 판매자 객체

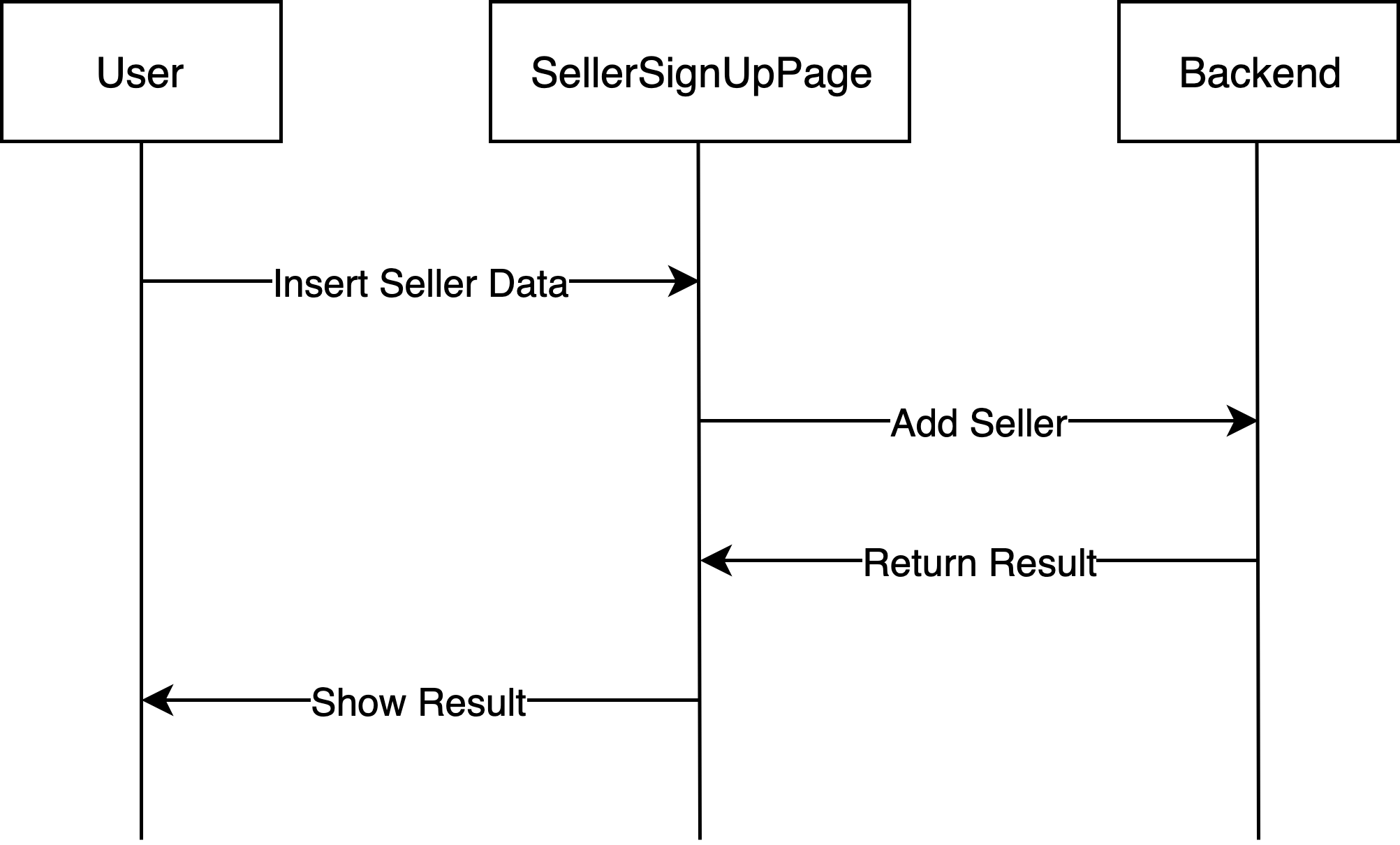
### A. attribute

+ Seller\_id : 판매자 고유 문자, 아이디

+ Password : 판매자가 설정한 비밀번호

+ Name : 판매자 이름

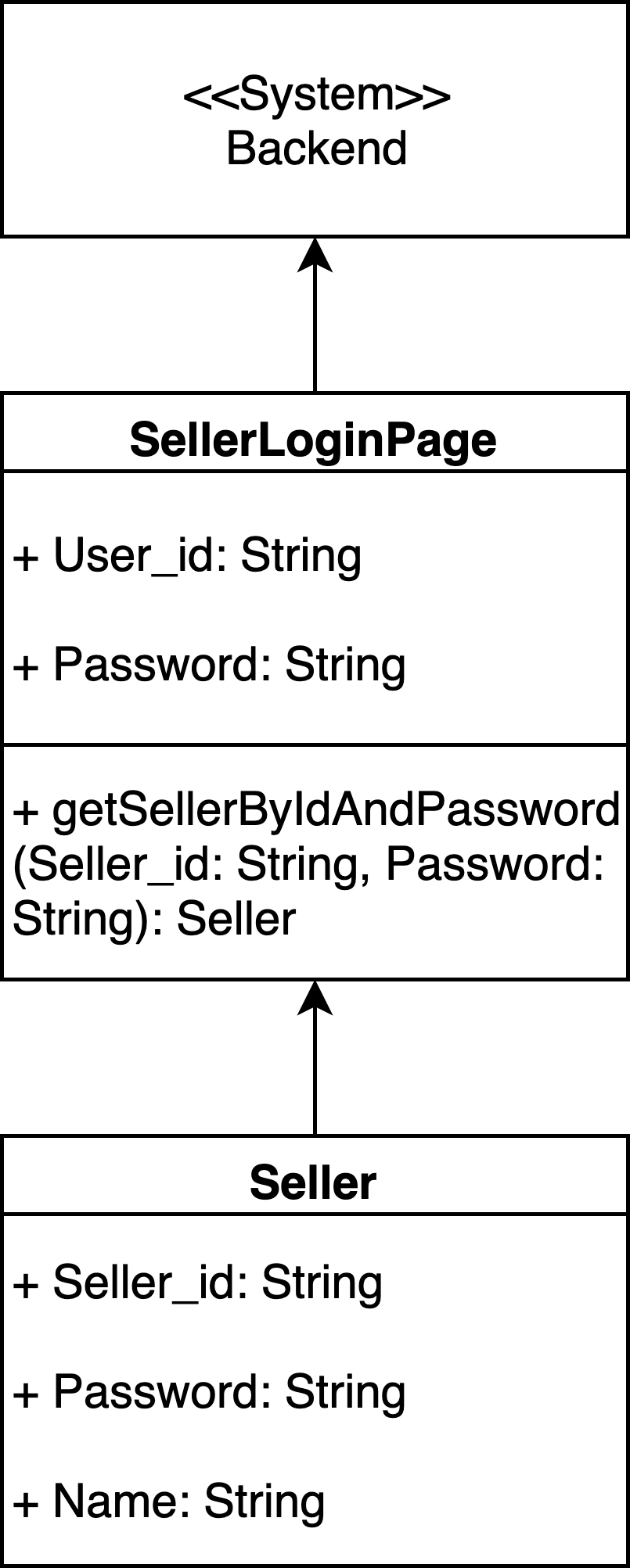
### G-2 Sequence Diagram



Diagrams 17 Subcomponent Seller Sign Up Sequence Diagram

### H. Seller Login

### H-1 Class Diagram



Diagrams Subcomponent Seller login Class Diagram

### 1. SellerLoginPage – 판매자 로그인 페이지 객체

### A. attribute

+ Uesr\_id : 판매자가 입력한 아이디

+ Password : 판매자가 입력한 비밀번호

### B. methods

+ getSellerByIdAndPassword(Seller\_id, Password) : 판매자가 입력한 아이디와 비밀번호를 통해 데이터베이스에서 검색, 일치 시 판매자 객체에 저장된 정보를 가져옴

### 2. Seller – 판매자 객체

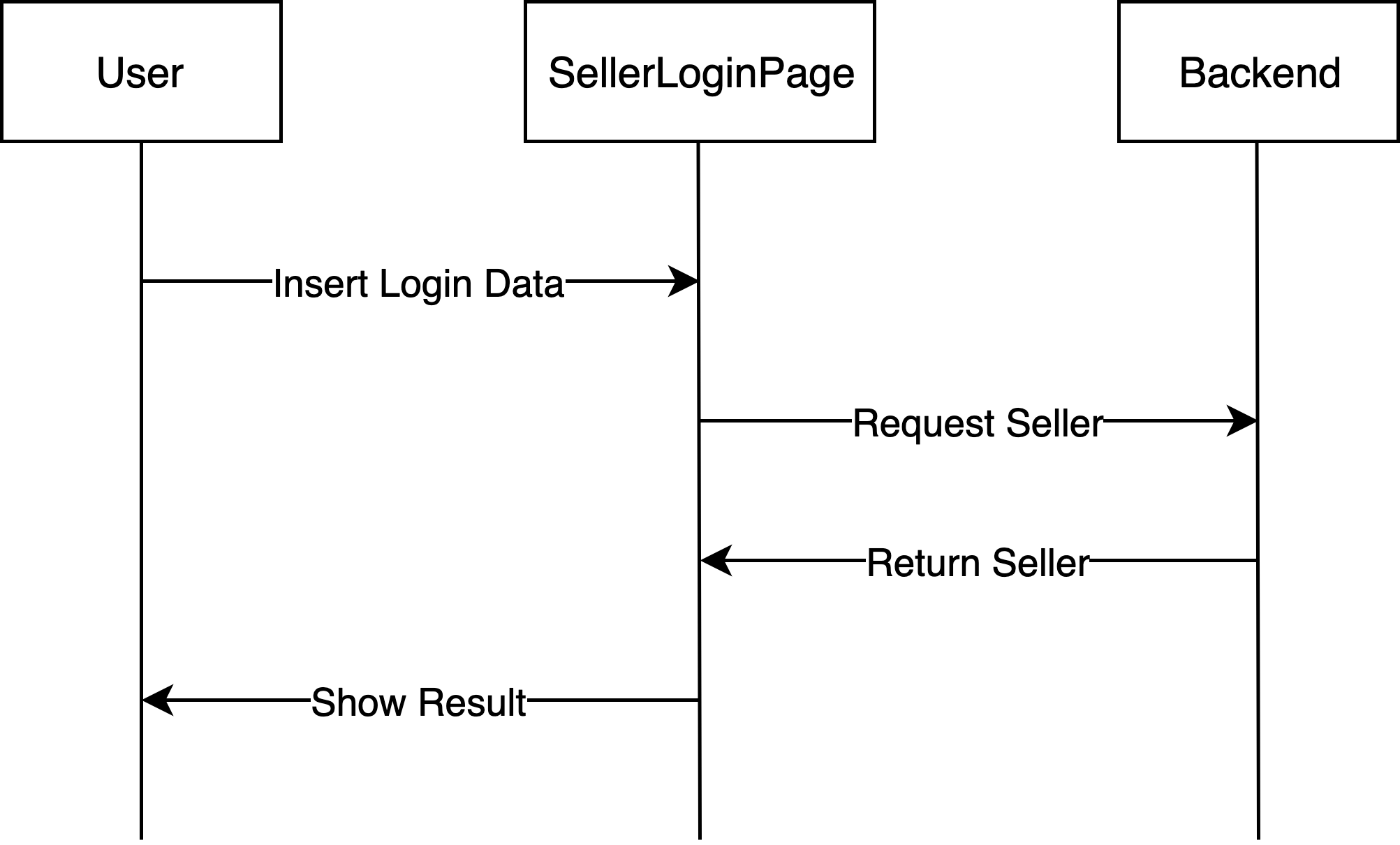
### A. attribute

+ Seller\_id : 판매자 고유 문자, 아이디

+ Password : 판매자가 설정한 비밀번호

+ Name : 판매자 이름

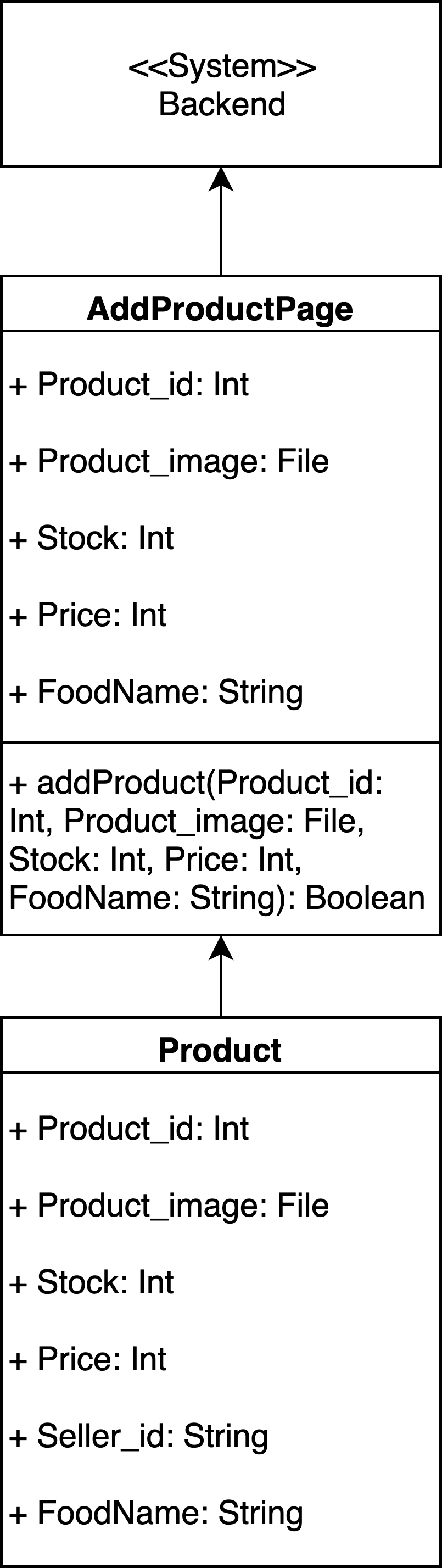
### H-2 Sequence Diagram



Diagrams Subcomponent Seller Login Sequence Diagram

### I. AddProduct

### I-1 Class Diagram



Diagrams 20 Subcomponent Add Product Page Class Diagram

### 1. AddProductPage – 물품 추가 페이지 객체

### A. attribute

+ Product\_id : 판매할 물품의 아이디

+ Product\_image : 판매할 물품의 사진

+ Stock : 판매할 물품의 재고

+ Price : 판매할 물품의 가격

+ FoodName : 판매할 물품의 음식 이름

### B. method

+ addProduct(Product\_id, Product\_image, Stock, Price, FoodName) : 판매자가 입력한 물품 정보를 판매 물품 리스트에 추가

### 2. Product – 물품 객체

### A. attribute

+ Product\_id : 물품의 아이디

+ Product\_image : 물품의 사진

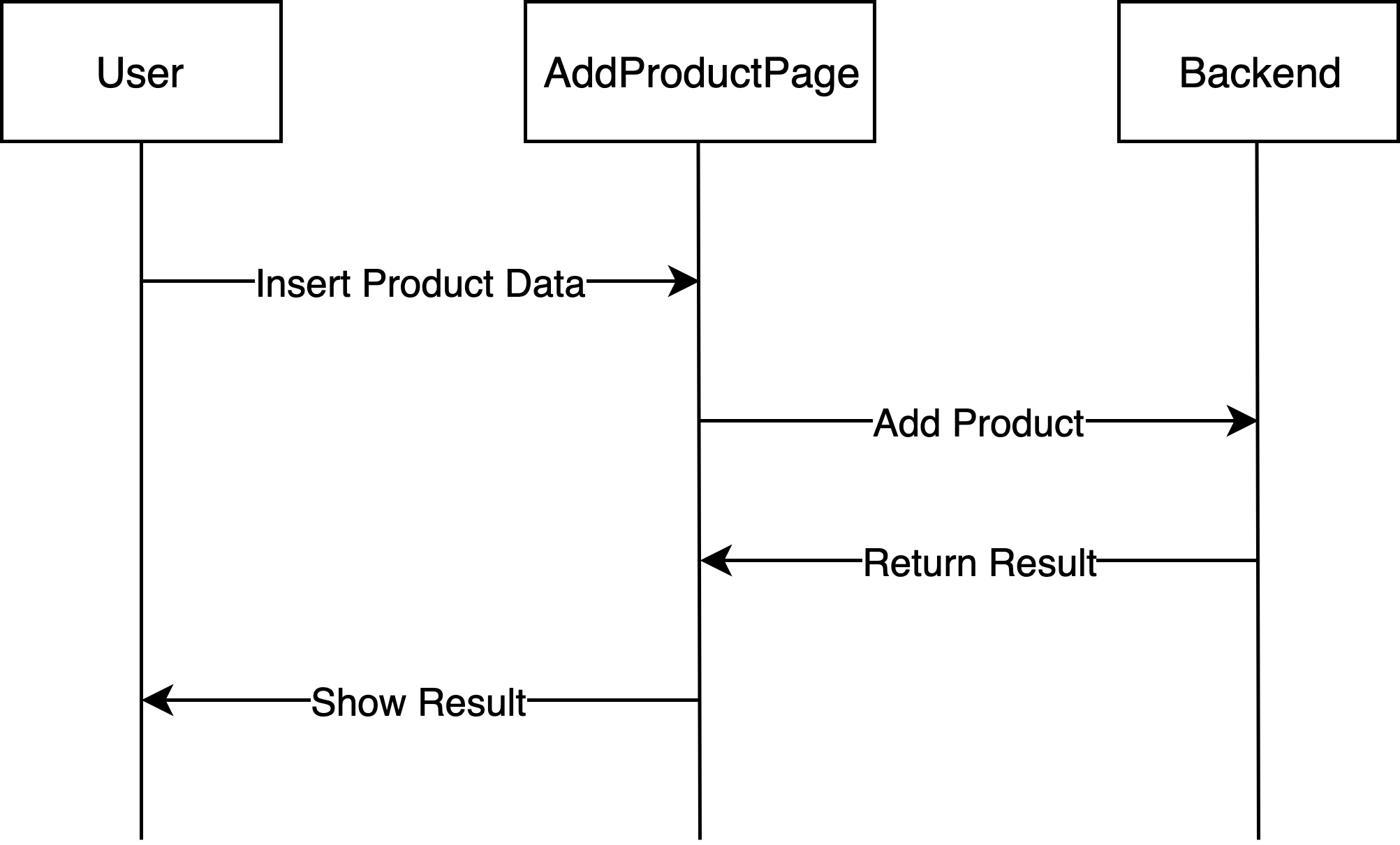
+ Stock : 해당 물품의 재고

+ Price : 해당 물품의 가격

+ Seller\_id : 해당 물품 판매자의 아이디

+ FoodName : 음식의 이름

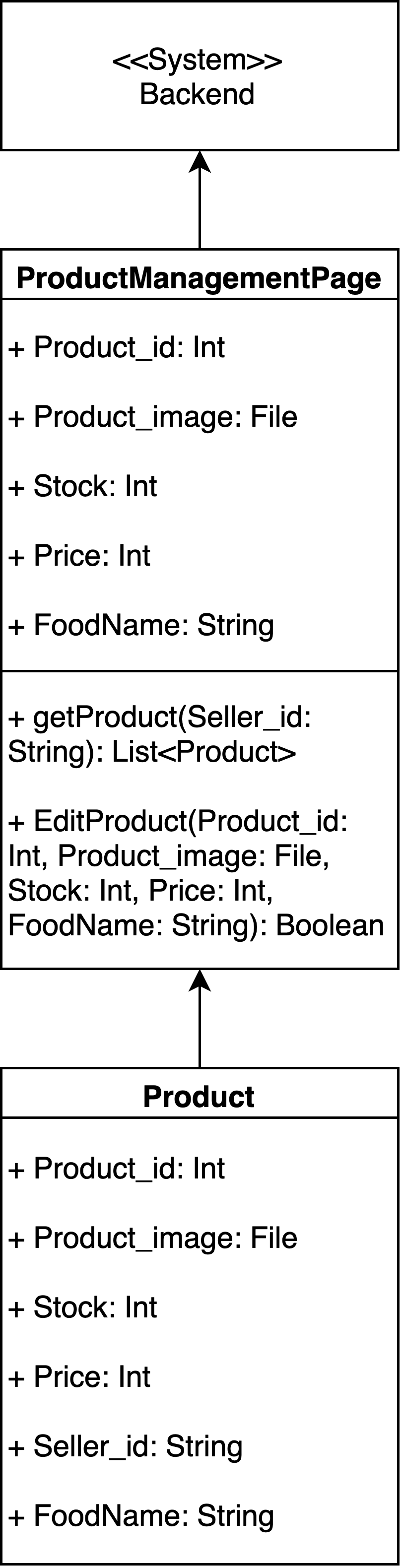
### I-2 Sequence Diagram



Diagrams 21 Subcomponent Add Product Page Sequence Diagram

### J. Product Management

### J-1 Class Diagram



Diagrams Subcomponent Product Management Class Diagram

### 1. ProductManagementPage – 물품 관지 페이지 객체

### A. attribute

+ Product\_id : 판매할 물품의 아이디

+ Product\_image : 판매할 물품의 사진

+ Stock : 판매할 물품의 재고

+ Price : 판매할 물품의 가격

+ FoodName : 판매할 물품의 음식 이름

### 2. Product – 물품 객체

### A. attribute

+ Product\_id : 물품의 아이디

+ Product\_image : 물품의 사진

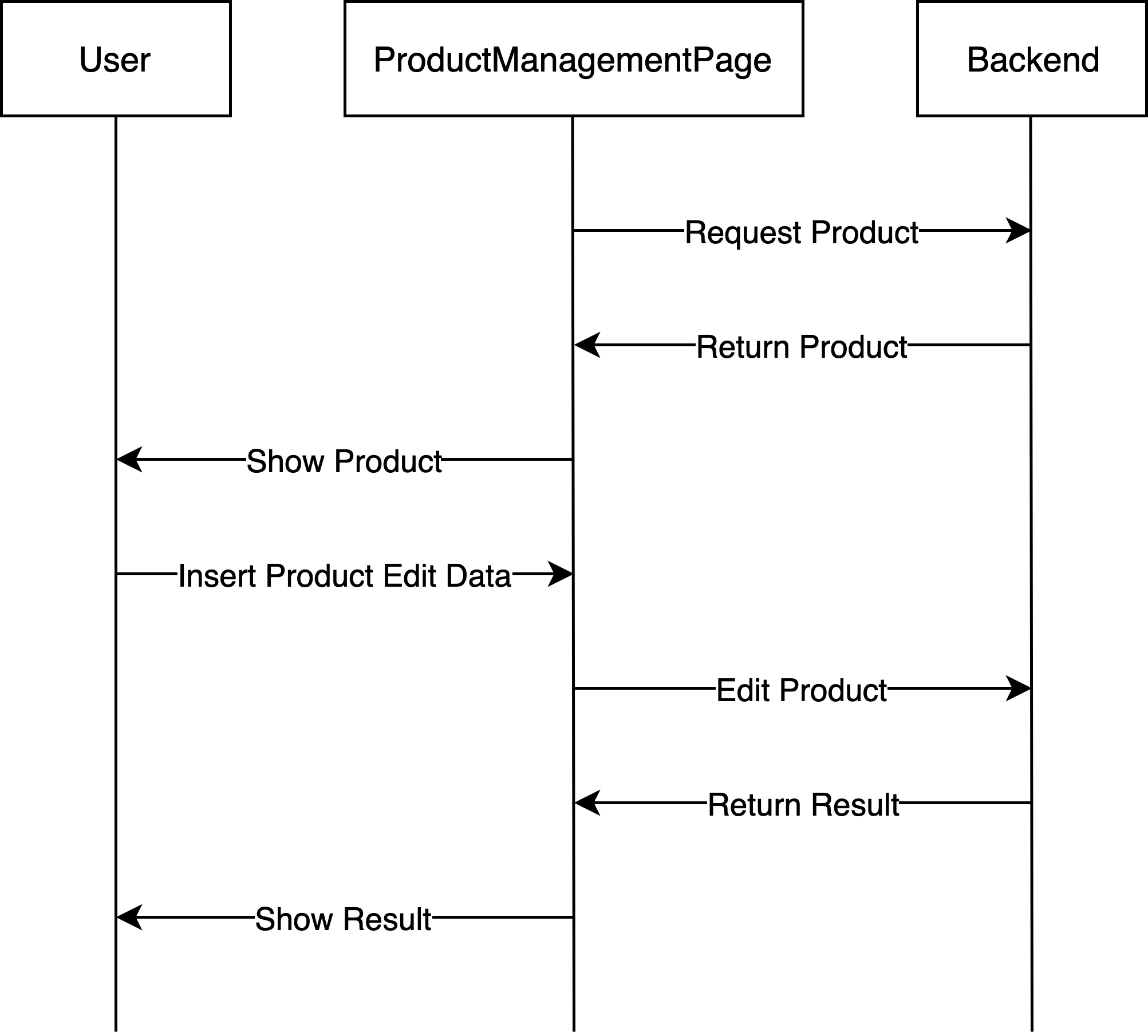
+ Stock : 해당 물품의 재고

+ Price : 해당 물품의 가격

+ Seller\_id : 해당 물품 판매자의 아이디

+ FoodName : 음식의 이름

### J-2 Sequence Diagram



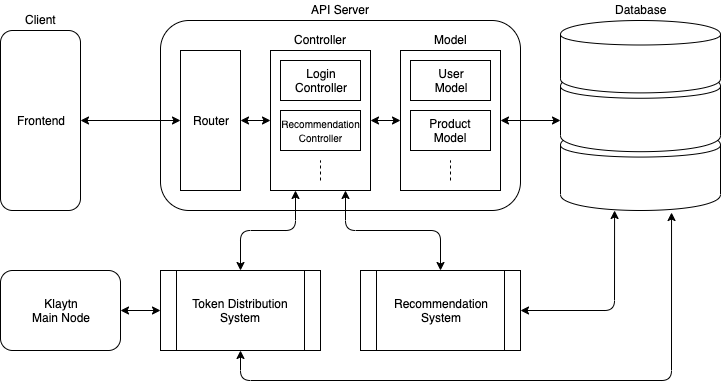
Diagrams Subcomponent Product Management Sequence Diagram

# System Architecture – Backend

## Objectives

Backend 시스템과 각 서브시스템에 대해 서술한다.

## Overall Architecture



Diagrams Backend Overall Architecture

Frontend에서 REST API를 통해 API Server에 요청을 보내면 Router가 각각의 요청에 해당하는 Controller를 호출한다. Controller에서는 요청에 해당하는 동작을 수행하며, 필요에 따라 외부 시스템인 Recommendation System과 Token Distribution System을 호출한다.

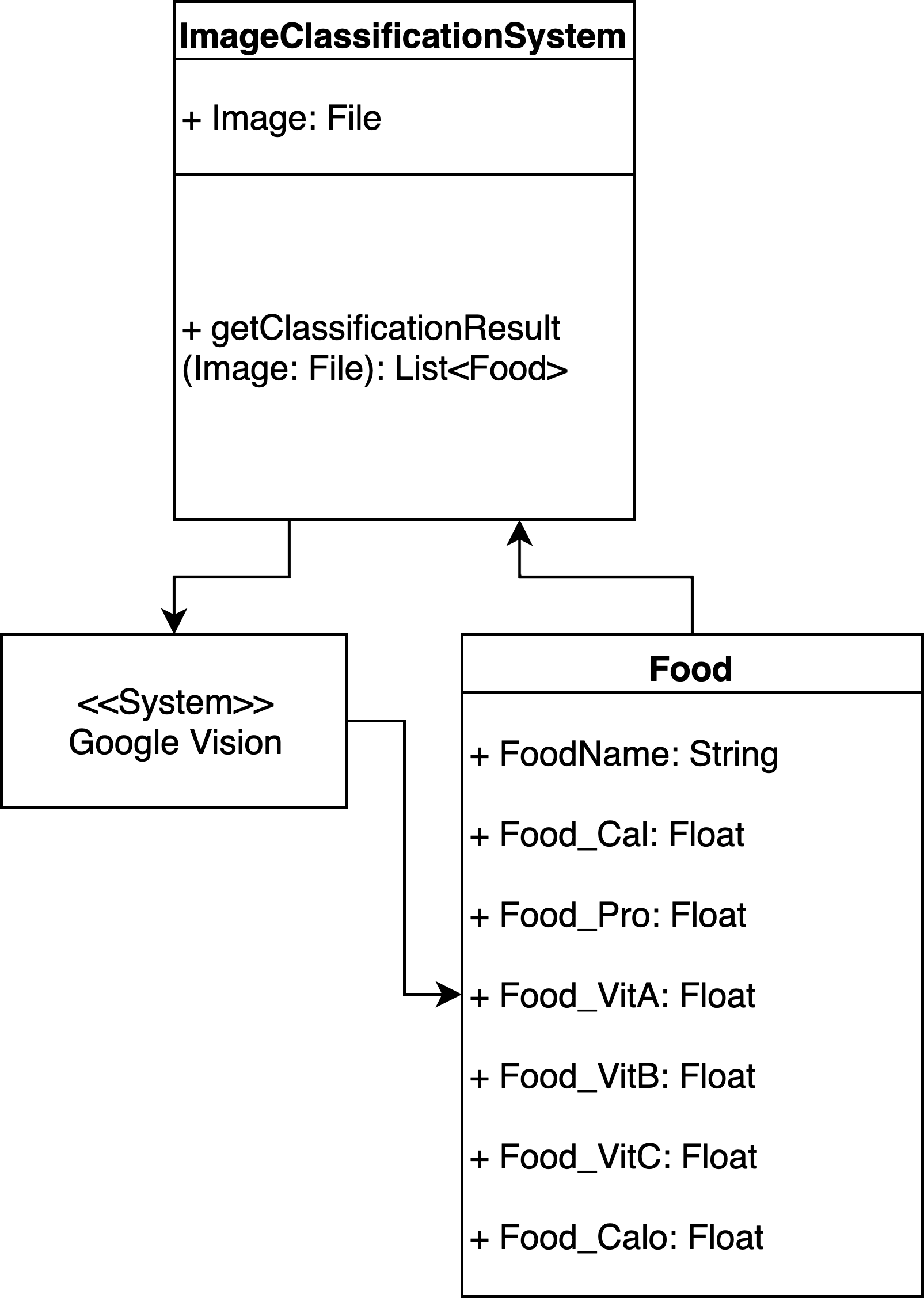
Controller에서는 데이터가 필요할 경우 필요에 따라 Model을 통해 DB와 통신하며, 통신 결과를 또한 필요에 따라 가공하여 Router를 통해 Frontend에 전송한다.

Backend System은 Main API Server 이외에도 Image Classification System, Nutrient State Analysis System, Recommendation System, Token Distribution System으로 구성되어 있다.

## Subcomponents

### A. ImageClassificationSystem

### A-1 Class Diagram



Diagrams Subcomponent Image Classification System Class Diagram

1. ImageClassificationSystem – 이미지 분류 시스템 객체

A. attribute

+ Image : 사용자가 촬영해 입력된 사진의 파일

B. methods

+ getClassificationResult(Image) : 입력된 이미지의 음식 후보 이름 리스트를 생성

2. Food – 음식 객체

A. attribute

+ FoodName : 음식의 이름

+ Food\_Cal : 음식의 칼로리

+ Food\_Pro : 음식의 단백질 함유량

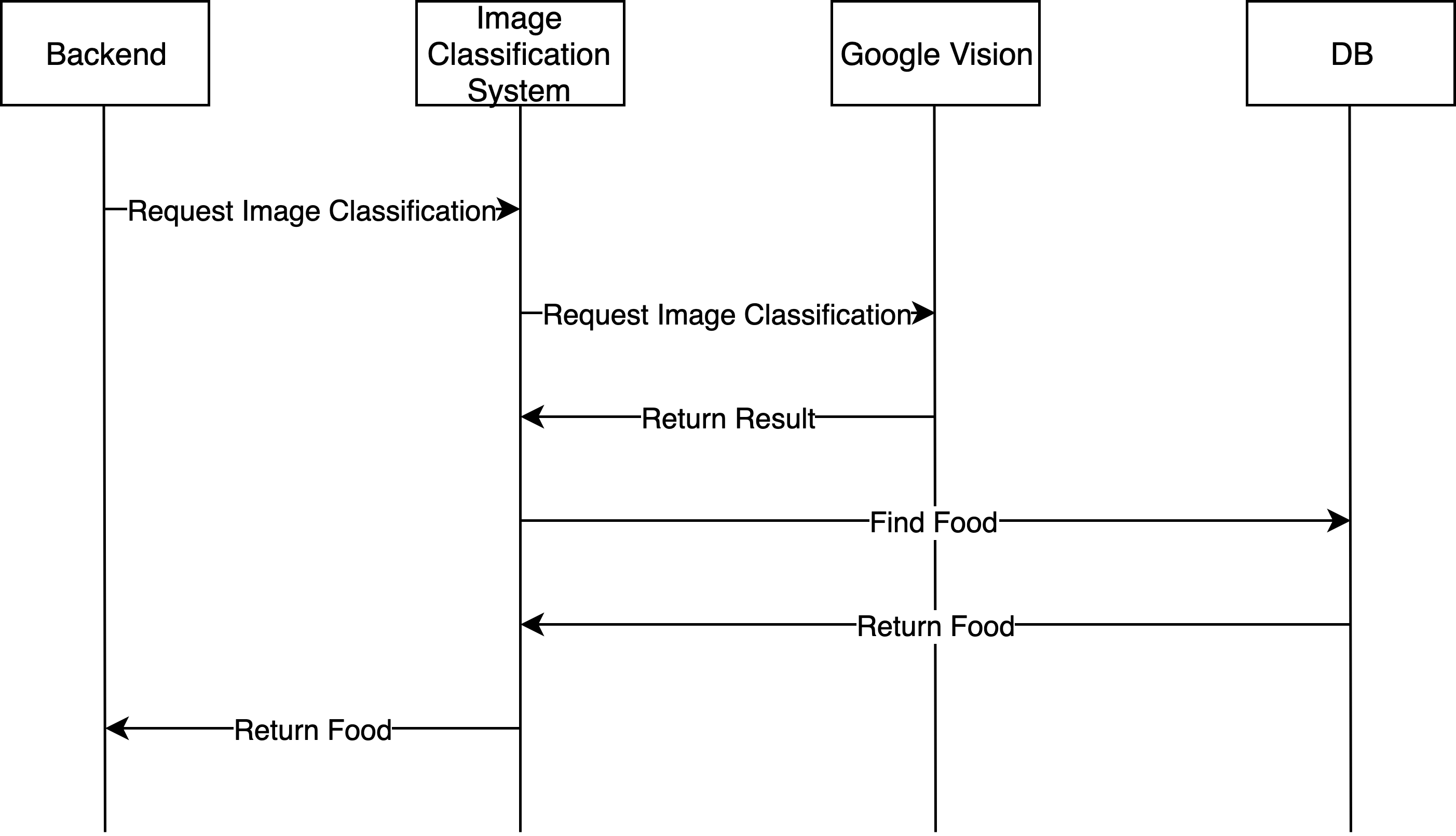
+ Food\_VitA : 음식의 비타민A 함유량

+ Food\_VitB : 음식의 비타민B 함유량

+ Food\_VitC : 음식의 비타민C 함유량

+ Food\_Calo : 음식의 철분 함유량

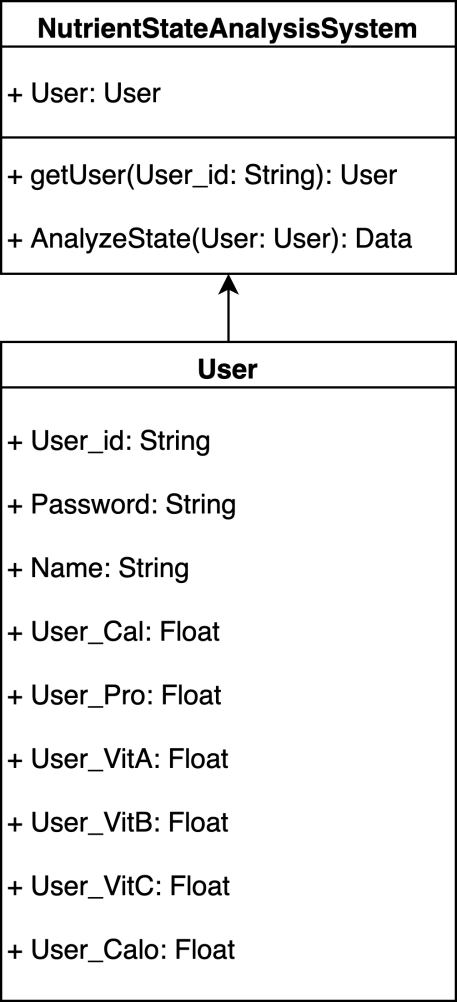
### A-2 Sequence Diagram



Diagrams Subcomponent Image Classification System Sequence Diagram

B. Nutrient State Analysis System

### B-1 Class Diagram



Diagrams Subcomponent Nutrient State Analysis System Class Diagram

1. NutrientStateAnalisysSystem – 영양상태 분석 시스템 객체

A. attribute

+ User : 분석 대상이 되는 유저의 User객체

B. methods

+ getUser(User\_id, Password) : 분석 대상이 되는 유저의 정보 가져오기

+ AnalysisStatae(User) : 유저 정보를 바탕으로 영양 상태 분석

2. User – 유저 객체

A. attribute

+ User\_id : 유저 식별 고유 코드

+ Password : 유저 식별 암호

+ Name : 유저의 이름

+ User\_Cal : 유저가 섭취한 칼로리 양

+ User\_Pro : 유저가 섭취한 단백질의 양

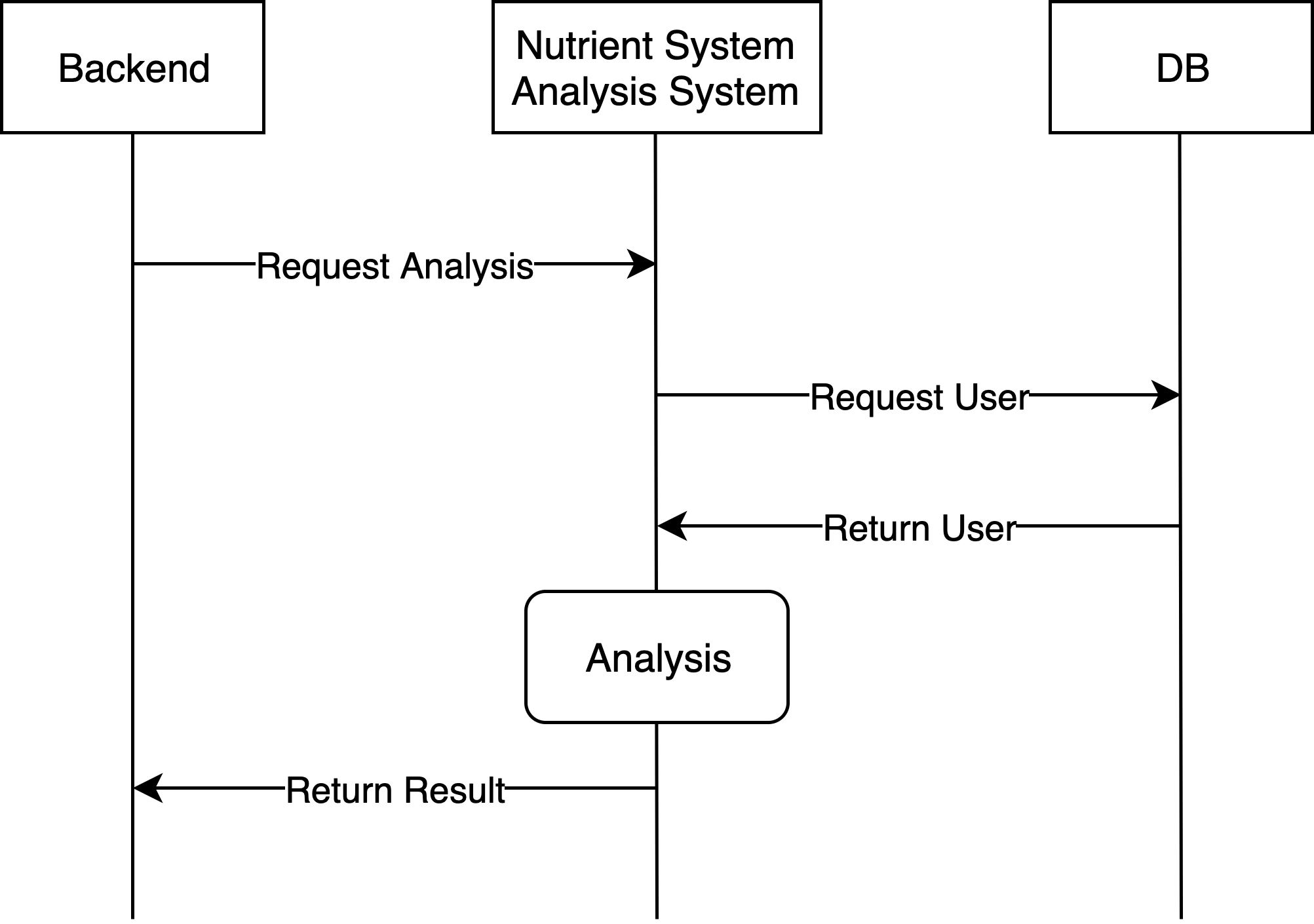
+ User\_VitA : 유저가 섭취한 비타민A의 양

+ User\_VitB : 유저가 섭취한 비타민B의 양

+ User\_VitC : 유저가 섭취한 비타민C의 양

+ User\_Calo : 유저가 섭취한 철분의 양

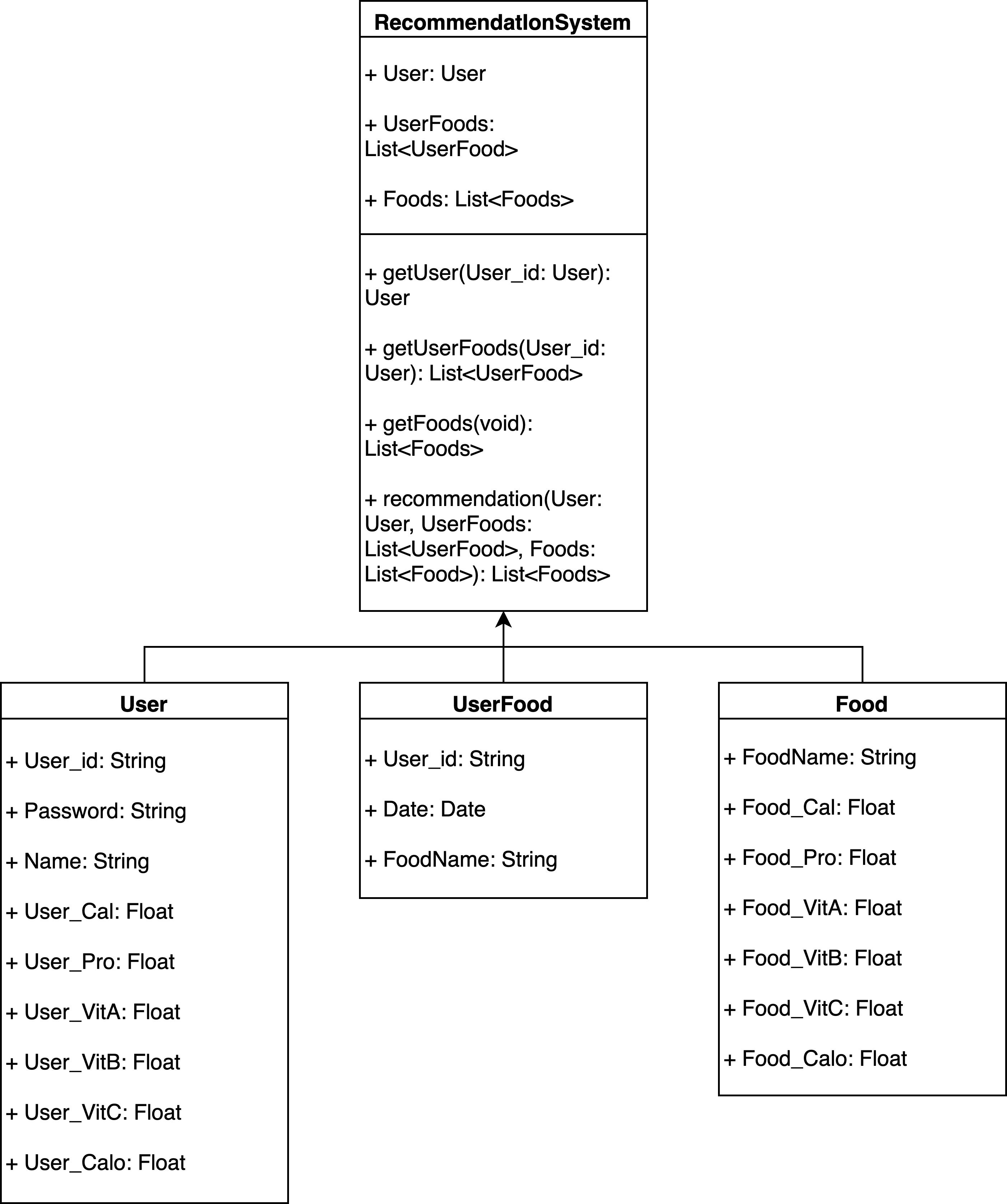
### B-2 Sequence Diagram



Diagrams Subcomponent Nutrient State Analysis System Sequence Diagram

### C. Recommendation System

### C-1 Class Diagram



Diagrams Subcomponent Recommendation System Class Diagram

1. RecommendationSystem – 추천 시스템 객체

A. attribute

+ User : 추천 대상이 되는 유저의 User 객체

+ UserFood : 추천 대상이 되는 유저가 등록한 음식 목록

+ Foods : 음식 정보가 담긴 리스트형의 Food객체

B. methods

+ getUser(User\_id) : 추천 대상이 되는 유저의 정보 가져오기

+ getUserFood(User\_id) : 유저 아이디로 UserFood에 접근하여 섭취한 음식 목록 가져오기

+ getFoods(void) : 해당 유저에게 추천할 음식 목록 생성

+ recommendation(User, UserFood)

2. User – 유저 객체

A. attribute

+ User\_id : 유저 식별 고유 코드

+ Password : 유저 식별 암호

+ Name : 유저의 이름

+ User\_Cal : 유저가 섭취한 칼로리 양

+ User\_Pro : 유저가 섭취한 단백질의 양

+ User\_VitA : 유저가 섭취한 비타민A의 양

+ User\_VitB : 유저가 섭취한 비타민B의 양

+ User\_VitC : 유저가 섭취한 비타민C의 양

+ User\_Calo : 유저가 섭취한 철분의 양

3. UserFood – 유저 음식 객체

A. attribute

+ User\_id : 유저 식별 고유 문자

+ Date : 유저가 음식을 섭취한 날짜

+ FoodName : 유저가 먹은 음식 이름

4. Food – 음식 객체

A. attribute

+ FoodName : 음식의 이름

+ Food\_Cal : 음식의 칼로리

+ Food\_Pro : 음식의 단백질 함유량

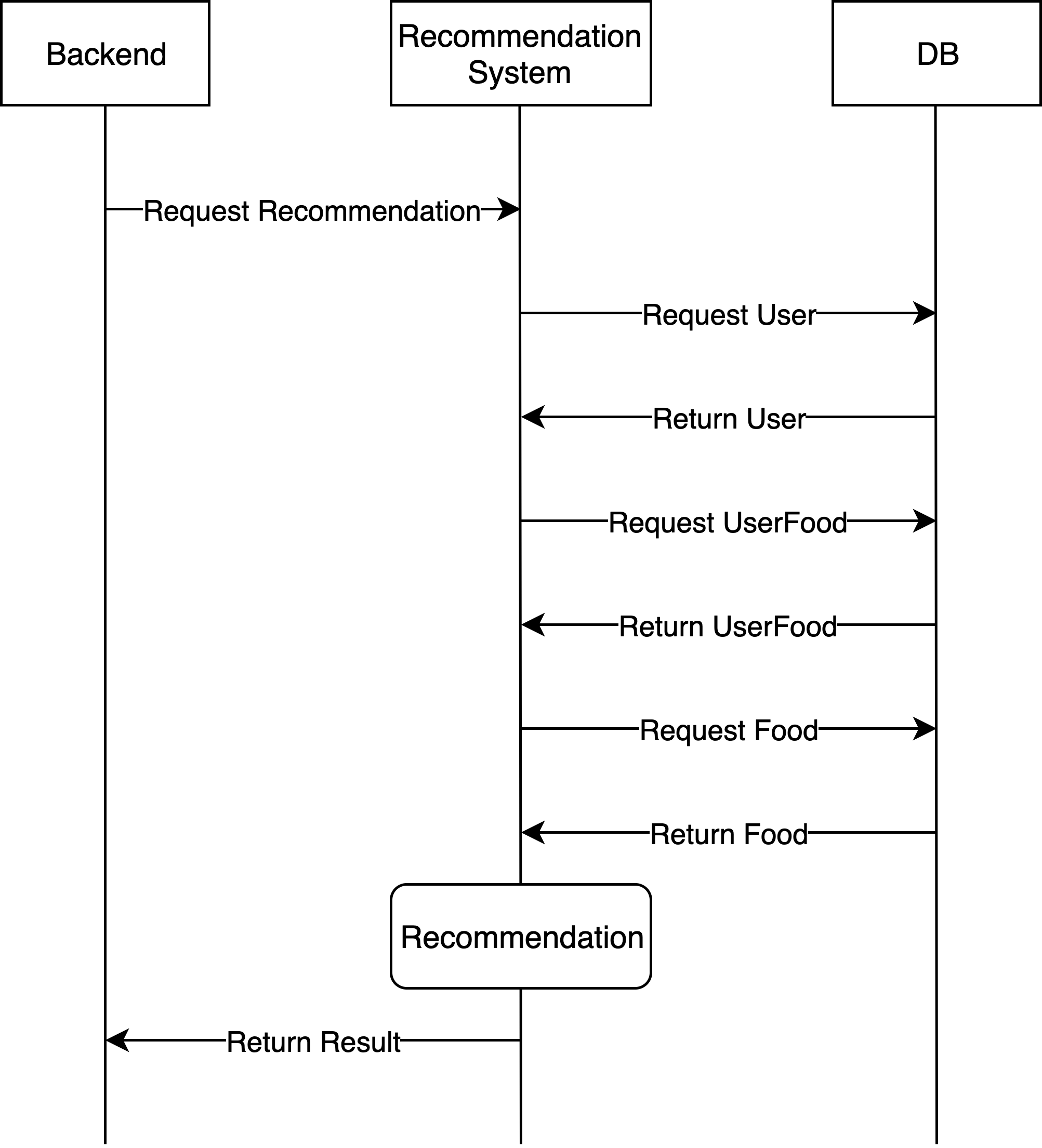
+ Food\_VitA : 음식의 비타민A 함유량

+ Food\_VitB : 음식의 비타민B 함유량

+ Food\_VitC : 음식의 비타민C 함유량

+ Food\_Calo : 음식의 철분 함유량

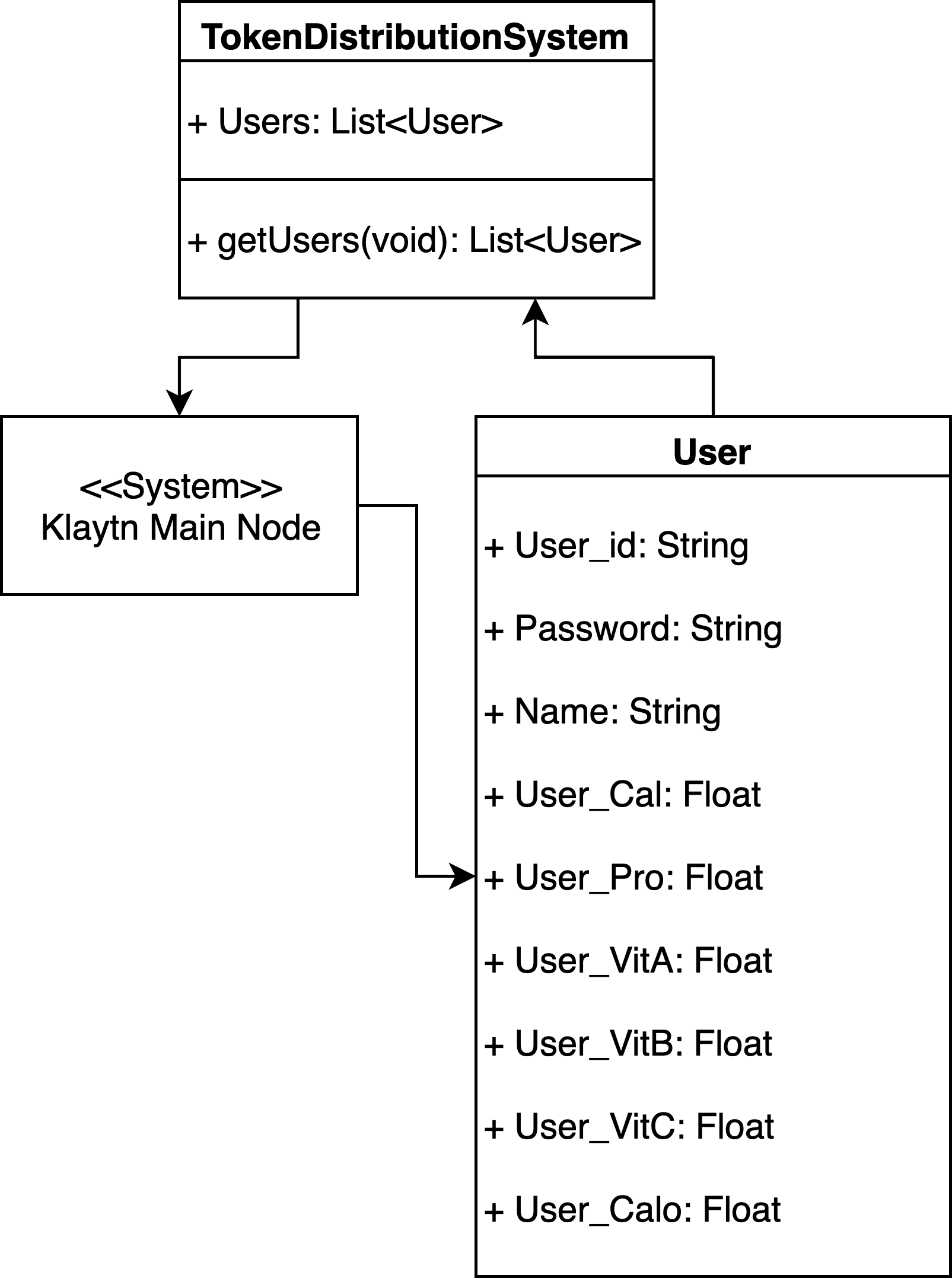
### C-2 Sequence Diagram



Diagrams Subcomponent Recommendation System Sequence Diagram

### D. Token Distribution System

### D-1 Class Diagram



Diagrams Subcomponent Token Distribution System Class Diagram

1. TokenDistributionSystem

A. attribute

+ Users : 유저 객체들의 리스트

B. methods

+ getUsers(void) : 유저 객체들을 리스트로 가져오기

2. User – 유저 객체

A. attribute

+ User\_id : 유저 식별 고유 코드

+ Password : 유저 식별 암호

+ Name : 유저의 이름

+ User\_Cal : 유저가 섭취한 칼로리 양

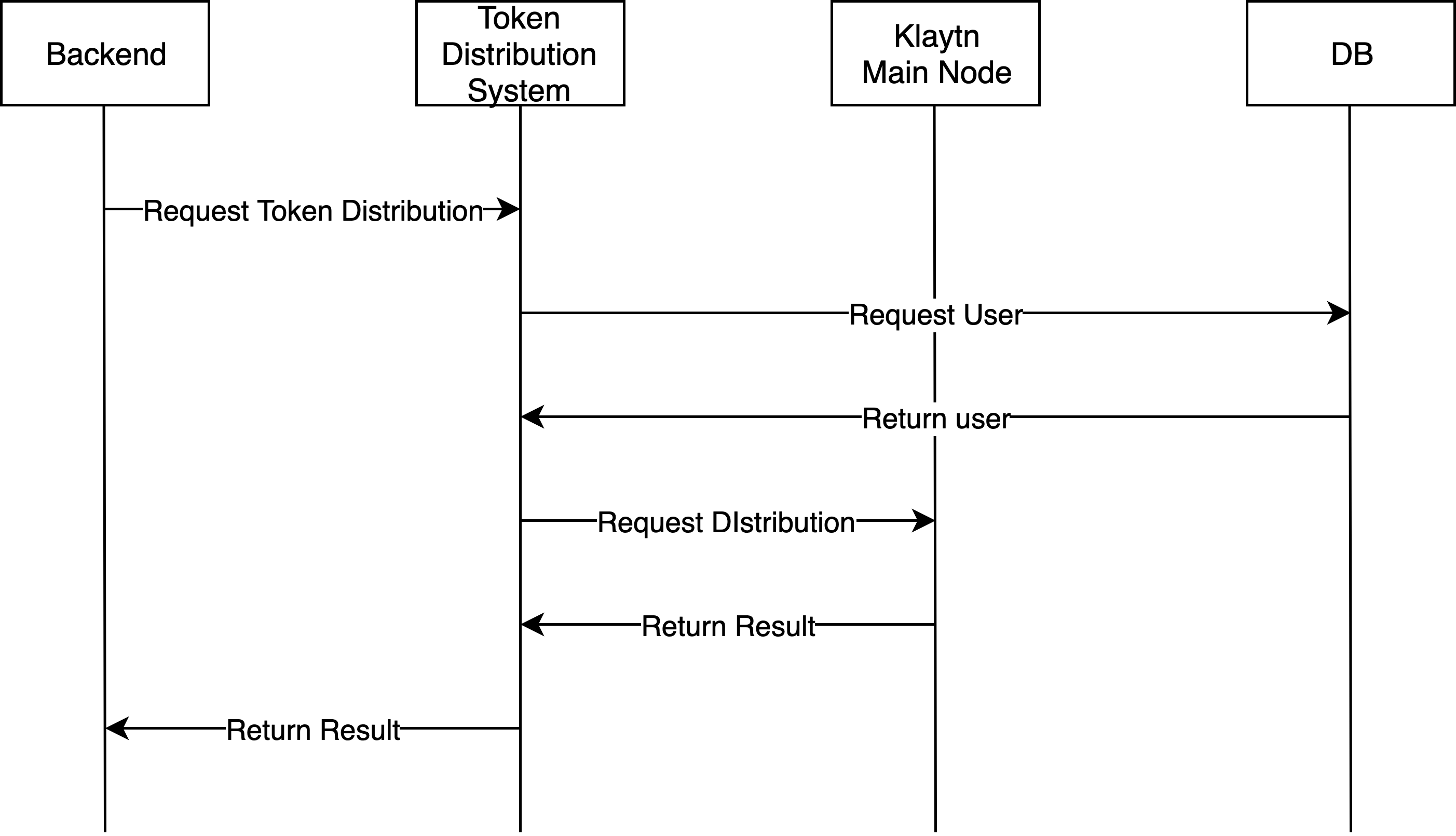
+ User\_Pro : 유저가 섭취한 단백질의 양

+ User\_VitA : 유저가 섭취한 비타민A의 양

+ User\_VitB : 유저가 섭취한 비타민B의 양

+ User\_VitC : 유저가 섭취한 비타민C의 양

### D-2 Sequence Diagram



Diagrams Subcomponent Token Distribution System Sequence Diagram

# 6. Protocol Design

## 6.1 Objectives

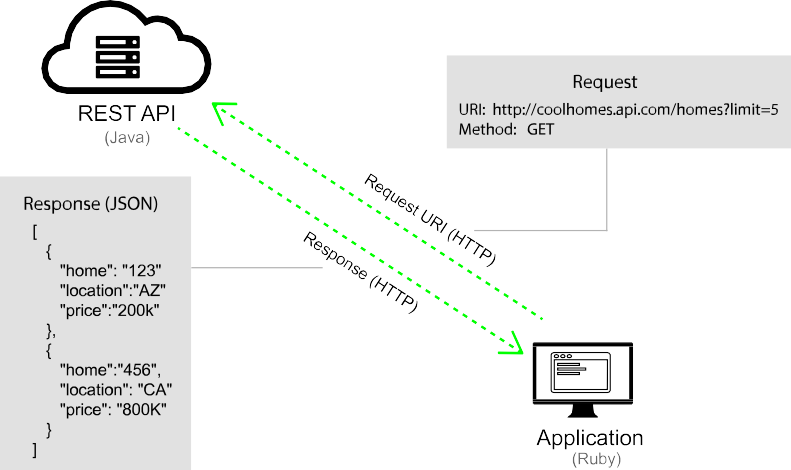
이번 챕터에서는 각 서브시스템 간의 상호작용, 특히 프론트 엔드 시스템과 백 엔드 애플리케이션 서버 시스템간 상호작용에 이용되는 프로토콜에 어떤 구조가 사용되는지 설명하고, 각 인터페이스가 어떻게 정의되어 있는지를 기술한다.

## 6.2. REST API





Figure REST API model



본 시스템은 프론트 엔드와 백 엔드 사이의 통신에 웹 인터페이스, 즉 HTTP 를 이용하며, 요청과 응답 형식은 REST API 에 따른다. REST API 란 Representational State Transfer의 약자로서, 서버에 저장되어 있는 각 자원을 이름으로 구분하여 해당 자원의 상태를 주고받는 API의 설계 형식을 의미한다. REST API는 크게 다음 세 부분으로 구성되어 있다.

* 1. 자원(Resources): URI

- 서버가 보관하고 있는 데이터를 나타내며, 각 자원은 고유한 URI를 가진다.

Design Specification

* 1. 행위(Verb): HTTP Method

- 서버의 자원에 접근해 상태를 조작하기 위한 요청 행위로서, 각 조작 행위는 HTTP Method를 통해 표현된다. (POST, GET, PUT, DELETE)

* 1. 표현(Representation): JSON

- 클라이언트의 요청에 대한 서버의 응답 형식으로, 주로 JSON 이 사용된다.

클라이언트와 서버 간의 통신에 REST API를 사용할 경우 서버와 클라이언트 간의 의존성이 줄어들고, 프로토콜이 이해하기 쉬워지는 장점이 있으며, 이를 통해 본 시스템에서 개발하는 프론트 엔드 클라이언트 뿐 아니라 다른 시스템에서 서버의 자원을 쉽게 이용할 수 있기 때문에 확장성이 높다.

## 6.3. JSON



Figure JSON Logo

Design Specification

JSON (JavaScript Object Notation)은 속성-값 쌍 또는 키 값 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷이다. 비동기 브라우저/서버 통신 (AJAX)을 위해, 넓게는 XML을 대체하는 주요 데이터 포맷이다. 특히, 인터넷에서 자료를 주고받을 때 그 자료를 표현하는 방법으로 알려져 있다. 자료의 종류에 큰 제한은 없으며, 특히 컴퓨터 프로그램의 변수 값을 표현하는 데 적합하다.

## 6.4. Details

### Authentication

##### Login

- Request

|  |  |  |
| --- | --- | --- |
| Method | POST | |
| URI | /authentication/login (REST 예외) | |
| Request Body | id | 사용자 ID |
| password | 사용자 패스워드 |

Table Authentication Login Request Details

- Response

|  |  |  |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | 400 Bad Request (ID/PW 불일치인 경우) | |
| Success  Response Body | success | true |
| Failure Response Body | success | false |
| message | reason of failure |

Table Authentication Login Response Details

##### Signup

* Request

|  |  |  |
| --- | --- | --- |
| Method | POST | |
| URI | /authentication/signup(REST 예외) | |
| Request Body | id | 사용자 ID |
| name | 사용자 이름 |
| password | 사용자 패스워드 |

Table Authentication Signup Request Details

* Response

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Success Code | 200 | OK |  | |  |  |  |
| Failure Code | 400 | Bad | Request | | (ID | 중복인 | 경우) |
| Success  Response Body | - | | | - | | | |
| Failure  Response Body | message | | | reason of failure | | | |

able Authentication Sign Up Response Details

### User

##### Get

* + - * + Request

|  |  |  |
| --- | --- | --- |
| Method | GET | |
| URI | /users/:id | |
| Parameters | - | - |
| Header | Authorization | 사용자 인증 토큰 |

Table User Get Request Details

* + - * + Response

|  |  |  |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | 404 Not Found (id에 해당하는 User가 없음) | |
| Success  Response Body | user | User Object |
| Failure  Response Body | - | - |

Table User Get Response Details

##### Search

* Request

|  |  |  |
| --- | --- | --- |
| Method | GET | |
| URI | /users | |
| Parameters | id | User ID |
| name | User Name |
| authority | User Authority |
| Header | Authorization | 사용자 인증 토큰 |

Table User Search Request Details

* Response

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Success Code | 200 | OK |  | |  |  |  |  |  |
| Failure Code | 404 | Not | Found | | (검색 | 조건에 | 해당하는 | User 가 | 없음) |
| Success  Response Body | users | | | User Object List | | | | | |
| Failure  Response Body | - | | | - | | | | | |

Table User Search Response Details

##### **UserFood**

##### Get

* + - * + Request

|  |  |  |
| --- | --- | --- |
| Method | GET | |
| URI | /users/:id | |
| Parameters | - | - |
| Header | Authorization | 사용자 인증 토큰 |

Table User Food Get Request Details

* + - * + Response

|  |  |  |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | 404 Not Found (id 에 해당하는 User 가 없음) | |
| Success  Response Body | userfood | Userfood Object |
| Failure  Response Body | - | - |

Table User Food Get Response Details

##### Search

* Request

|  |  |  |
| --- | --- | --- |
| Method | GET | |
| URI | /users | |
| Parameters | id | Food ID |
| name | Food Name |
| authority | User Authority |
| Header | Authorization | 사용자 인증 토큰 |

Table User Food Search Request Details

* Response

|  |  |  |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | 404 Not Found(검색 조건에 해당하는 UserFood 없음) | |
| Success  Response Body | userFood | User Food Object List |
| Failure  Response Body | - | - |

Table User Food Search Response Details

### Food

##### Get

* + - Request

|  |  |  |
| --- | --- | --- |
| Method | GET | |
| URI | /food/:id | |
| Parameters | - | - |
| Header | - | - |

Table Food Get Request Details

* + - Response

|  |  |  |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | 404 Not Found(검색 조건에 해당하는 Food 없음) | |
| Success  Response Body | food | Food Object |
| Failure  Response Body | - | - |

Table Food Get Response Details

##### Search

* + - Request

|  |  |  |
| --- | --- | --- |
| Method | GET | |
| URI | /food | |
| Parameters | id | food ID |
| name | food Name |
| Header | Authorization | 사용자 인증 토큰 |

Table Food Search Request Details

* + - Response

|  |  |  |
| --- | --- | --- |
| Success Code |  | |
| Failure Code |  | |
| Success  Response Body | food | Food Object List |

Table Food Search Response Details

### Product

##### Get

* + Request

|  |  |  |
| --- | --- | --- |
| Method | GET | |
| URI | /product/:id | |
| Parameters | - | - |
| Header | - | - |

Table Product Get Request Details

* + Response

|  |  |  |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | 404 Not Found(검색 조건에 해당하는 Product 없음) | |
| Success  Response Body | product | product Object |
| Failure  Response Body | - | - |

Table Product Get Response Details

##### Search

* + Request

|  |  |  |
| --- | --- | --- |
| Method | GET | |
| URI | /product | |
| Parameters | Product\_id | Product ID |
| Food\_Name | Product name |
| price | Product price |
| Header | Authorization | 사용자 인증 토큰 |

Table Product Search Request Details

* + Response

|  |  |  |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code |  | |
| Success  Response Body | products | Product Object List |

Table Product Get Response Details

##### Write

* Request

|  |  |  |
| --- | --- | --- |
| Method | POST | |
| URI | /product | |
| Parameters | Food\_Name | Product name |
| Product\_image | Product image |
| Price | Product price |
| Stock | Prodcut stock |
| Header | Authorization | 사용자 인증 토큰 |

Table Product Write Request Details

* Response

|  |  |  |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | 400 Bad Request | |
| Success  Response Body | product\_id | Created product id |
| Failure  Response Body | message | fail reasons  - 해당 item에 대한 product가 이미 있음 |

Table Product Write Response Details

##### Delete

* + Request

|  |  |  |
| --- | --- | --- |
| Method | DELETE | |
| URI | /product/:id | |
| Header | Authorization | 사용자 인증 토큰 |

Table Product Delete Request Details

* + Response

|  |  |
| --- | --- |
| Success Code | 200 OK |
| Failure Code | 403 Forbidden (삭제 권한 없음) |

Table Product Delete Response Details

### Recommendation

##### Get

* + - Request

|  |  |  |
| --- | --- | --- |
| Method | GET | |
| URI | /recommendations/:id | |
| Parameters | -Food Name | -Food name |
| -Main Nutrition | -Main nutrition of food |
| Header | - | - |

Table Recommendation Get Request Details

* + - Response

|  |  |  |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | 404 Not Found(id 해당하는 recommendation 없음) | |
| Success  Response Body | recommend | Recommendation category object |
| Failure  Response Body | - | - |

Table Recommendation Get Response Details

##### Search

- Request

|  |  |  |
| --- | --- | --- |
| Method | GET | |
| URI | /recommendations | |
| Parameters | id | Recommendation id |
| category | Item category |
| Header | Authorization | 사용자 인증 토큰 |

Table Recommendation Search Request Details

* + - * + Response

|  |  |  |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | 404 Not Found (검색 조건에 해당하는 Recommendation없음) | |
| Success  Response Body | recommend | Recommendation category object |
| Failure  Response Body | - | - |

Table Recommendation Search Response Details

### Seller

##### Get

* + - * + Request

|  |  |  |
| --- | --- | --- |
| Method | GET | |
| URI | /sellers/:id | |
| Parameters | - | - |
| Header | Authorization | 사용자 인증 토큰 |

Table Seller Get Request Details

* + - * + Response

|  |  |  |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | 404 Not Found (id에 해당하는 Seller 없음) | |
| Success  Response Body | seller | Seller Object |
| Failure  Response Body | - | - |

Table Seller Get Response Details

##### Search

* Request

|  |  |  |
| --- | --- | --- |
| Method | GET | |
| URI | /users | |
| Parameters | id | Seller ID |
| name | Seller Name |
| authority | Seller Authority |
| Header | Authorization | 사용자 인증 토큰 |

Table Seller Search Request Details

* Response

|  |  |  |
| --- | --- | --- |
| Success Code | 200 OK | |
| Failure Code | 404 Not Found (검색 조건에 해당하는 Seller 없음) | |
| Success  Response Body | sellers | Seller Object List |
| Failure  Response Body | - | - |

Table Seller Search Response Details

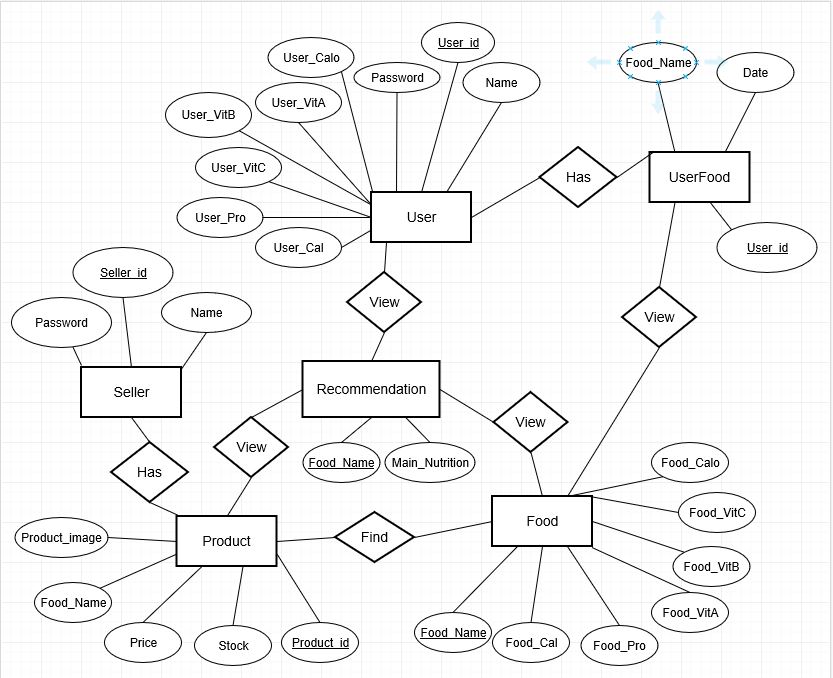
# 7. Database Design

## 7.1. Objectives

본 챕터에서는 시스템이Requirement specification에서 작성한 데이터베이스 요구사항을 기반으로 하여 세부적인 데이터베이스 설계를 기술한다. Entity-Relation Diagram을 통해 개괄적인 Entity 간의 관계를 기술하고, Relational Schema 명세를 만든다.

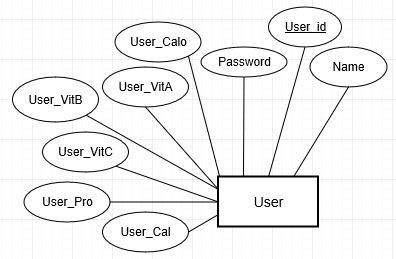
## 7.2. Entity-Relation Diagram

본 시스템에는User, User Food, Seller, Product, Food, Recommendation의 6개의 Entity가 존재한다. Entity-Relation Diagram에서 Entity는 직사각형으로 표현하며, Entity가 가지는 Attribute는 타원형으로 표현하고, Entity간의 관계는 마름모로 표현한다. 이때, 각 Entity를 식별하는 Primary key가 되는 Attribute 밑줄을 그어 표시한다.



Diagrams System ER Diagram

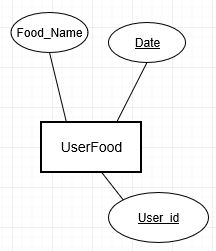
1. Entity
2. User



Diagrams User ER Diagram

User Entity는 사용자 정보를 가진다. Primary key는 user\_id이며, password와 이름, 6종류의 영양소 정보를 가지고 있다.

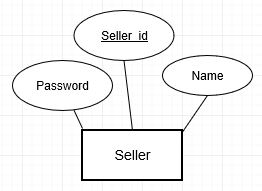
1. User Food



Diagrams User Food ER Diagram

User Food Entity는 사용자가 먹은 음식 정보를 가진다. Date와 User\_id를 composite key로 이용하며, 음식 이름 정보를 가지고 있다.

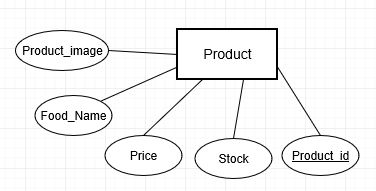
1. Seller



Diagrams Seller ER Diagram

Seller Entity는 판매자 정보를 가진다. Primary key는 seller\_id이며, password와 이름 정보를 가지고 있다.

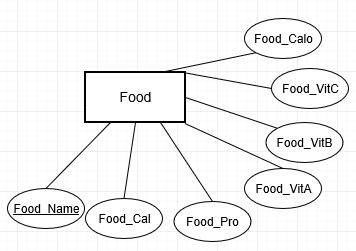
1. Product



Diagrams Product ER Diagram

Product Entity는 판매자가 판매하는 제품 정보를 가진다. Primary key는 Product\_id이며, 이름, 가격, 재고, 제품 이미지 정보를 가지고 있다.

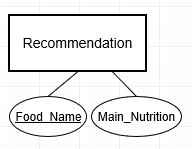
1. Food



Diagrams Food ER Diagram

Food Entity는 시스템 관리자가 지정해 놓은 음식의 영양성분 정보를 가진다. Primary key는 Food\_name이며, 칼로리, 단백질, 비타민A, 비타민B, 비타민C, 칼슘함량 정보를 가지고 있다.

1. Recommendation



Diagrams Recommendation ER Diagram

Recommendation Entity는 음식 추천 알고리즘에 사용되는 정보를 가진다. Primary key는 Food\_Name이며, 주 영양소 정보를 가지고 있다.

1. Relational Schema

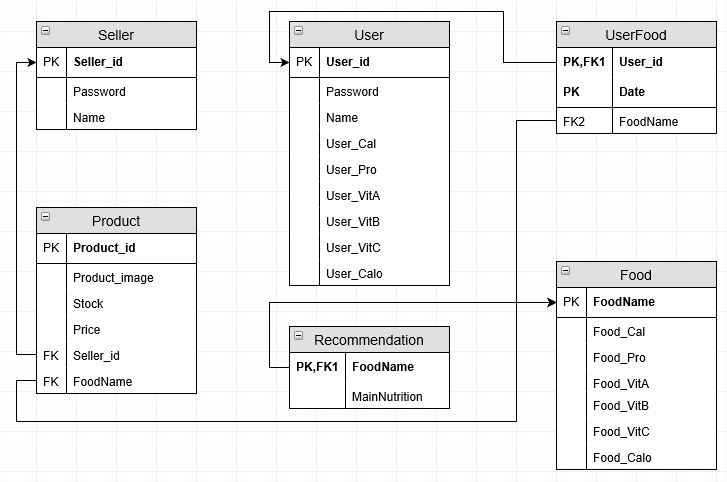


Figure 3 System Relational Schema

## 7.3. SQL DDL

1. User

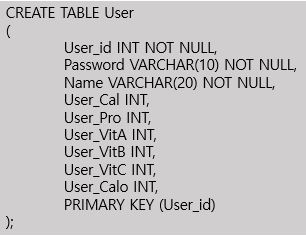


Figure 4 User SQL DDL

1. User Food

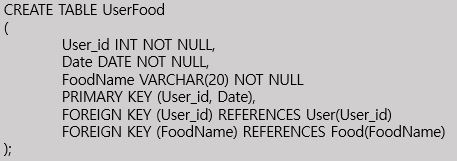


Figure 5 User Food SQL DDL

1. Seller



Figure 6 Seller SQL DDL

1. Product

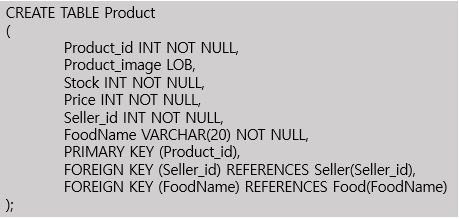


Figure 7 Product SQL DDL

1. Food

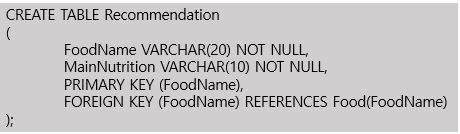


Figure 8 Product SQL DDL

1. Recommendation



Figure 9 Recommendation SQL DDL

# 8. Testing Plan

## 8.1 Objectives

Testing Plan에서는 Test Policy와 Test Case에 대해 설명한다. 해당 챕터의 목적은 전체 시스템이 의도한대로 실행되는지를 확인, 검증(verification, validation)하기 위한 과정을 사전에 계획하는 데에 있다.

## 8.2 Testing Policy

### A. Development Testing

Development testing은 소프트웨어 개발 리스트를 줄이기 위한 넓은 범위의 탐색 전력, 방어 전략, 공격 예방 기능들을 목적으로 두고 있다.

이 단계에서, 정적인 코드 분석, 데이터 흐름 분석, 동료 코드 리뷰, 유닛 테스팅을 진행한다. Reliability, security, performance, regulatory compliance에 초점을 두고 테스팅 해야 한다.

1) Reliability:

다양한 음식 사진에 대한 데이터베이스가 등록되어 있어야, 유저가 입력한 사진을 분석했을 때 올바른 결과가 나올 수 있다. 또한, 음식 데이터베이스 속 음식 정보가 믿을 수 있는 정보여야하며, 수집된 결과를 각 영양소 별 필요한 양에 맞춰 가공해서 보여줘야 한다. 마지막으로 결제를 진행하는 과정에서 오류가 없어야 한다.

2) Security

사용자가 개인 카카오 계정을 통해 회원가입을 할 때 보안 시스템을 통해 데이터베이스가 개인 정보를 보호해야 한다. 이는 해싱 알고리즘을 통해 달성될 수 있다. 또한, 데이터베이스는 외부의 의도적인 침입에 대해 접근을 막고 중요한 정보를 잃지 않아야 한다. 특히, 결제 서비스가 포함된 본 프로젝트는 해당 과정의 보안도 중요시 여겨야 한다.

3) Performance

Performance는 속도, 정확성을 통해 평가 가능하다. 음식 사진을 분석하는데 필요한 속도는 빠를수록 좋으며, 그렇게 추출된 후보 안에 정확한 음식이 나와야 한다. 또한, 이러한 모든 작업은 데이터가 굉장히 많이 쌓여있을 때에도 좋은 퍼포먼스가 유지 되어야 한다.

### B. Release Testing

Release Testing은 새로운 버전/소프트웨어의 설치/ 어플리케이션을 의도하지 않은 문제나 오류 없이 작동하는지 확인하는 단계다. 이 작업은 출시 전에 반드시 진행되어야 한다.

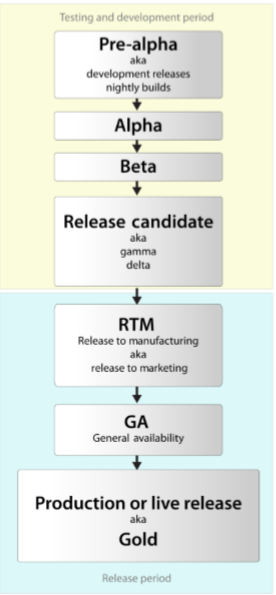


Figure Releasing Testing & Development Period

위의 테스팅 & 개발 주기처럼, 우리는 먼저 개발자들을 통해 알파 버전 테스팅을 진행하고, 최종 점검을 위해 베타 버전을 출시해야 한다. 베타 버전 터스팅을 통해, 사용자의 피드백을 받아 다음 개발 사이클에 사용할 수 있다.

### C. User Testing

User Testing은 Usability testing과 같은 개념으로 설명할 수 있다. 시나리오를 설명하고 유저 테스트가 필요한 실재 상황을 설정한다. 우리는 20명의 유저와 500개의 음식 정보가 있다고 가정했다. 이 모든 상황을 설정하고, 우리는 위에서 설명한 각 컨셉을 확인하고 테스트 안에서 그 값들을 비교해야 하다.

### D. Testing Case

Testing Case는 정확한 숫자를 통해 설정할 수 있다. 우리는 20명의 유저와 500개의 음식 리스트를 가지고 테스팅을 진행하며, 각 음식 별 1~3개의 판매자를 지정하기로 했다.

# 9. Development Plan

## 9.1 Objectives

Development Plan에서는 개발 계획에 대해 서술한다. 시스템을 구현하는 데 필요한 개발 도구와 프로그래밍 언어, 라이브러리 등의 개발 환경에 대해 설명하고, 도표를 이용해 시스템 개발 일정을 기술한다.

## 9.2 Frontend Environment

### A. Android

Android Studio는 자바로 쓰인 JetBrains의 IntelliJ IDEA 소프트웨어를 바탕으로 안드로이드 개발을 위해 설계된 구글의 안드로이드 운영체제를 위한 통합 개발 환경(IDE, Integrated Development Environment)이다. 다양한 기본 템플릿들을 기반으로 일반적인 안드로이드 디자인과 구성요소들을 만들 수 있으며 풍부하게 만들어진 레이아웃 편집기를 통해 드래그 앤 드롭 UI 구성요소와 멀티 스크린을 구성하는 데에도 편리하다. 또한, 안드로이드 가상 디바이스 기능을 제공해, 안드로이드 스튜디오 자체에서 코드를 실행하고 디버깅할 수 있다.

### B. React

React는 사용자 인터페이스(UI, User Interface)를 만들기 위해 사용하는 자바스크립트 라이브러리 중 하나이다. 리액트는 [싱글 페이지](https://ko.wikipedia.org/wiki/%EC%8B%B1%EA%B8%80_%ED%8E%98%EC%9D%B4%EC%A7%80_%EC%95%A0%ED%94%8C%EB%A6%AC%EC%BC%80%EC%9D%B4%EC%85%98)나 모바일 애플리케이션의 개발 시 토대로 사용할 수 있으며, 다른 라이브러리와 함께 사용함으로써 복잡한 리액트 애플리케이션들은 [상태 관리, 라우팅, API와의 통신을](https://ko.wikipedia.org/w/index.php?title=%EC%83%81%ED%83%9C_%EA%B4%80%EB%A6%AC&action=edit&redlink=1) 할 수 있다.

## 9.3 Backend Environment

### A. MySQL

MySQL은 오라클이 관리 및 지원하고 있는 관계형 데이터베이스 관리 시스템(RDBMS, Relational Database Management System)이다. 데이터베이스를 만들고, 관리하는데, 데이터를 백업하는데, 상태를 검사하고, 데이터베이스 구조를 생성하는데, 또는 데이터 레코더를 작성하는데 있어서 MySQL 프론트엔드 데스크톱 소프트웨어나 웹 애플리케이션을 사용하거나 명령 줄 인터페이스를 사용한다.

### B. Node.js

Node.js는 확장성 있는 네트워크 애플리케이션, 특히 서버 사이드 개발에 사용되는 소프트웨어 플랫폼이다. 자바스크립트를 활용하며, Non-Blocking I/O와 단일 스레드 이벤트 루프를 통한 높은 처리 성능을 가지고 있다. 내장 HTTP 서버 라이브러리를 포함하고 있어 웹 서버에서 아파피 등의 별도의 소프트웨어 없이 동작하는 것이 가능하고, 이를 통해 웹 서버의 동작에 있어 더 많은 통제를 가능하게 한다.

### C. Klaytn

Klaytn은 카카오의 그라운드X에서 출시한 블록체인 플랫폼으로 강력한 보안성과 투명성을 제공하면서 엔터프라이즈급 성능 및 안전성을 제공하는 퍼블릭 블록체인을 지향한다. 약 1초의 Block Interval과 300TPS의 빠른 속도를 가져 거래의 완료성을 높혔다. 또한, IBFT(이스탄불 비잔티움 결함 허용)을 합의 알고리즘으로 적용해 강력한 보안성과 높은 성능, 안전성을 제공하고, 동시에 생성된 블록을 누구나 검증 할 수 있도록 공개해 신뢰성과 투명성도 확보했다.

### D. Flask

Flask는 파이썬으로 구동되는 웹 어플리케이션 프레임워크다. 즉, 서버와 웹 어플리케이션의 통신을 하기 위해 필요한 인터페이스이다. 플라스크는 프레임워크들 중 매우 심플하고 가벼우며, 중요하고 핵심적인 내용과 기능을 갖고 있다.

### E. Google Cloud Vision API

Vision API는 google cloud에서 제공하는 REST 및 RPC API를 통해 선행 학습된 강력한 머신러닝 모델이다. 이미지에 라벨을 할당하고 사전 정의된 수백만 개의 카테고리로 빠르게 분류할 수 있다. 객체와 얼굴을 감지하고 인쇄 및 필기 텍스트를 읽으며 유용한 메타데이터를 이미지 카탈로그에 구축해 준다.

이미지 내에 있는 각 객체의 위치를 포함하여 여러 객체를 감지하고 분류하며, AutoML Vision Edge를 사용하면 속도와 정확성이 우수한 모델을 빌드 및 베포하여 에지의 이미지를 분류하고 로컬 데이터를 바탕으로 실시간 작업을 트리거할 수 있다.

## 9.4 Schedule

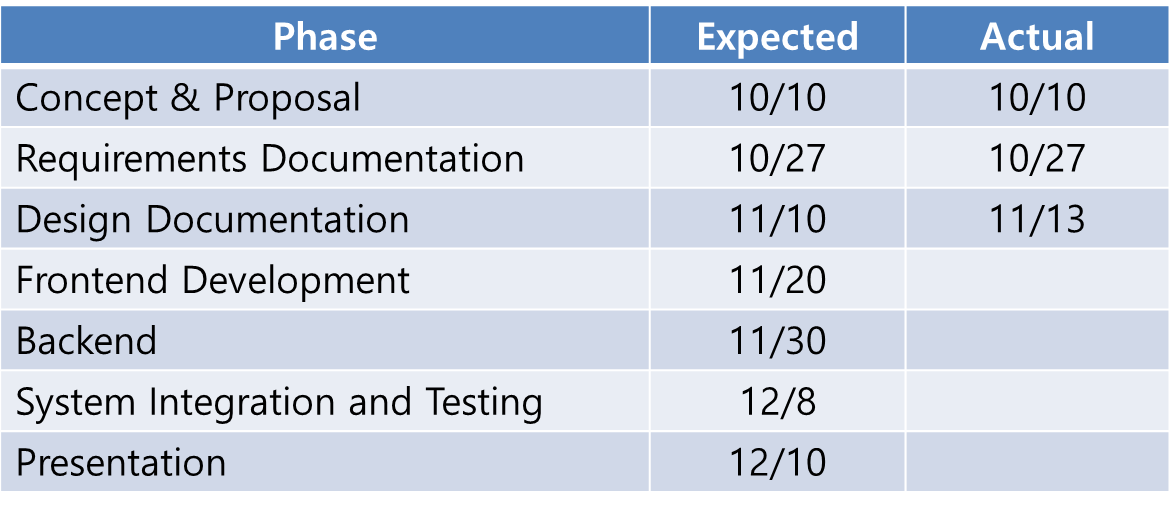


Figure Development Schedule

수업 스케줄 변경으로 Design Documentation부터 조금씩 날짜가 밀렸다. 병렬 개발을 통해 부족한 시간을 채워 시스템 개발을 진행해야 한다.

# 10. INDEX

## 10.1 Table

[Table 1 Authentication Login Request Details 53](#_Toc24579242)

[Table 2 Authentication Login Response Details 53](#_Toc24579243)

[Table 3 Authentication Signup Request Details 53](#_Toc24579244)

[able 4 Authentication Sign Up Response Details 54](#_Toc24579245)

[Table 5 User Get Request Details 55](#_Toc24579246)

[Table 6 User Get Response Details 55](#_Toc24579247)

[Table 7 User Search Request Details 55](#_Toc24579248)

[Table 8 User Search Response Details 56](#_Toc24579249)

[Table 9 User Food Get Request Details 57](#_Toc24579250)

[Table 10 User Food Get Response Details 57](#_Toc24579251)

[Table 11 User Food Search Request Details 57](#_Toc24579252)

[Table 12 User Food Search Response Details 57](#_Toc24579253)

[Table 13 Food Get Request Details 58](#_Toc24579254)

[Table 14 Food Get Response Details 58](#_Toc24579255)

[Table 15 Food Search Request Details 58](#_Toc24579256)

[Table 16 Food Search Response Details 58](#_Toc24579257)

[Table 17 Product Get Request Details 59](#_Toc24579258)

[Table 18 Product Get Response Details 59](#_Toc24579259)

[Table 19 Product Search Request Details 59](#_Toc24579260)

[Table 20 Product Get Response Details 60](#_Toc24579261)

[Table 21 Product Write Request Details 61](#_Toc24579262)

[Table 22 Product Write Response Details 61](#_Toc24579263)

[Table 23 Product Delete Request Details 61](#_Toc24579264)

[Table 24 Product Delete Response Details 61](#_Toc24579265)

[Table 25 Recommendation Get Request Details 62](#_Toc24579266)

[Table 26 Recommendation Get Response Details 62](#_Toc24579267)

[Table 27 Recommendation Search Request Details 62](#_Toc24579268)

[Table 28 Recommendation Search Response Details 63](#_Toc24579269)

[Table 29 Seller Get Request Details 63](#_Toc24579270)

[Table 30 Seller Get Response Details 63](#_Toc24579271)

[Table 31 Seller Search Request Details 63](#_Toc24579272)

[Table 32 Seller Search Response Details 64](#_Toc24579273)

## 10.2 Figures

[Figure 1 REST API model 51](file:///C:\Users\onesm\Desktop\4학년\소프트웨어공학개론\project\design%20specification\Team1-design_specification.docx#_Toc24579229)

[Figure 2 JSON Logo 52](file:///C:\Users\onesm\Desktop\4학년\소프트웨어공학개론\project\design%20specification\Team1-design_specification.docx#_Toc24579230)

[Figure 3 System Relational Schema 69](#_Toc24579231)

[Figure 4 User SQL DDL 70](#_Toc24579232)

[Figure 5 User Food SQL DDL 70](#_Toc24579233)

[Figure 6 Seller SQL DDL 70](#_Toc24579234)

[Figure 7 Product SQL DDL 72](#_Toc24579235)

[Figure 8 Product SQL DDL 72](#_Toc24579236)

[Figure 9 Recommendation SQL DDL 72](#_Toc24579237)

[Figure 10 Releasing Testing & Development Period 74](#_Toc24579238)

[Figure 11 Development Schedule 78](#_Toc24579239)

## 10.3 Diagrams

[Diagrams 1 Frontend Overall System Diagram 16](#_Toc24579274)

[Diagrams 2 Frontend System Diagram 18](#_Toc24579275)

[Diagrams 3 Backend System Diagram 19](#_Toc24579276)

[Diagrams 4 Subcomponent Sign Up Class Diagram 20](#_Toc24579277)

[Diagrams 5 Subcomponent Sign Up Sequence Diagram 22](#_Toc24579278)

[Diagrams 6 Subcomponent Login Class Diagram 23](#_Toc24579279)

[Diagrams 7 Subcomponent Login Sequence Diagram 25](#_Toc24579280)

[Diagrams 8 Subcomponent Add Food Class Diagram 26](#_Toc24579281)

[Diagrams 9 Subcomponent Add Food Sequence Diagram 28](#_Toc24579282)

[Diagrams 10 Subcomponent Recommendation Class Diagram 29](#_Toc24579283)

[Diagrams 11 Subcomponent Recommendation Sequence Diagram 30](#_Toc24579284)

[Diagrams 12 Subcomponent My Page Class Diagram 31](#_Toc24579285)

[Diagrams 13 Subcomponent My Page Sequence Diagram 32](#_Toc24579286)

[Diagrams 14 Subcomponent Order Class Diagram 33](#_Toc24579287)

[Diagrams 15 Subcomponent Order Sequence Diagram 34](#_Toc24579288)

[Diagrams 16 Subcomponent Seller Sign Up Class Diagram 35](#_Toc24579289)

[Diagrams 17 Subcomponent Seller Sign Up Sequence Diagram 36](#_Toc24579290)

[Diagrams 18 Subcomponent Seller login Class Diagram 37](#_Toc24579291)

[Diagrams 19 Subcomponent Seller Login Sequence Diagram 38](#_Toc24579292)

[Diagrams 20 Subcomponent Add Product Page Class Diagram 39](#_Toc24579293)

[Diagrams 21 Subcomponent Add Product Page Sequence Diagram 40](#_Toc24579294)

[Diagrams 22 Subcomponent Product Management Class Diagram 41](#_Toc24579295)

[Diagrams 23 Subcomponent Product Management Sequence Diagram 42](#_Toc24579296)

[Diagrams 24 Backend Overall Architecture 43](#_Toc24579297)

[Diagrams 25 Subcomponent Image Classification System Class Diagram 44](#_Toc24579298)

[Diagrams 26 Subcomponent Image Classification System Sequence Diagram 45](#_Toc24579299)

[Diagrams 27 Subcomponent Nutrient State Analysis System Class Diagram 46](#_Toc24579300)

[Diagrams 28 Subcomponent Nutrient State Analysis System Sequence Diagram 46](#_Toc24579301)

[Diagrams 29 Subcomponent Recommendation System Class Diagram 47](#_Toc24579302)

[Diagrams 30 Subcomponent Recommendation System Sequence Diagram 48](#_Toc24579303)

[Diagrams 31 Subcomponent Token Distribution System Class Diagram 49](#_Toc24579304)

[Diagrams 32 Subcomponent Token Distribution System Sequence Diagram 50](#_Toc24579305)

[Diagrams 33 System ER Diagram 65](#_Toc24579306)

[Diagrams 34 User ER Diagram 66](#_Toc24579307)

[Diagrams 35 User Food ER Diagram 66](#_Toc24579308)

[Diagrams 36 Seller ER Diagram 67](#_Toc24579309)

[Diagrams 37 Product ER Diagram 67](#_Toc24579310)

[Diagrams 38 Food ER Diagram 68](#_Toc24579311)

[Diagrams 39 Recommendation ER Diagram 68](#_Toc24579312)