REVIEW REVOLUTION

design specification



|  |  |
| --- | --- |
| Student Number | Name |
| 2017314380 | 윤성경 |
| 2014311058 | 김진태 |
| 2014314650 | 유종현 |
| 2013310608 | 정창호 |

Contents

[1. Preface 5](#_Toc24112817)

[1.1. Readership 5](#_Toc24112818)

[1.2. Document Structure 6](#_Toc24112819)

[A. Introduction 6](#_Toc24112820)

[B. System Architecture 6](#_Toc24112821)

[C. Protocol Design 6](#_Toc24112822)

[D. Database Design 6](#_Toc24112823)

[E. Testing Plan 6](#_Toc24112824)

[F. Development Plan 6](#_Toc24112825)

[G. Index 6](#_Toc24112826)

[2. Introduction 7](#_Toc24112827)

[2.1. Objectives 7](#_Toc24112828)

[2.2. Applied Diagram 7](#_Toc24112829)

[2.3. Applied Tool 10](#_Toc24112830)

[2.4. Project Scope 12](#_Toc24112831)

[3. System Architecture – Overall 13](#_Toc24112832)

[3.1. Objectives 13](#_Toc24112833)

[3.2. System Organization 13](#_Toc24112834)

[4. System Architecture – Frontend 17](#_Toc24112835)

[4.1. Objective 17](#_Toc24112836)

[4.2. Subcomponents 18](#_Toc24112837)

[A. Customized Search Result 18](#_Toc24112838)

[B. Item Page 21](#_Toc24112839)

[C. Keyword Visualizer 24](#_Toc24112840)

[D. Review Page 26](#_Toc24112841)

[E. Review Post 31](#_Toc24112842)

[F. User Page 32](#_Toc24112843)

[5. System Architecture – Backend 35](#_Toc24112844)

[5.1. Objectives 35](#_Toc24112845)

[5.2. Overall Architecture 35](#_Toc24112846)

[5.3. Subcomponents 35](#_Toc24112847)

[A. Login 35](#_Toc24112848)

[B. Review Analysis 37](#_Toc24112849)

[C. Similar Item Finder 40](#_Toc24112850)

[D. 상품 정보 제공 43](#_Toc24112851)

[E. 개인정보 변경 45](#_Toc24112852)

[B. Appendices 47](#_Toc24112853)

[1. Hardware requirements 47](#_Toc24112854)

[2. Database requirements 48](#_Toc24112855)

[A. User 48](#_Toc24112856)

[B. Item 48](#_Toc24112857)

[C. Review 49](#_Toc24112858)

[D. Keyword 49](#_Toc24112859)

[10. Index 50](#_Toc24112860)

[10.1. Tables 50](#_Toc24112861)

[10.2. Figures 51](#_Toc24112862)

[10.3. Diagrams 51](#_Toc24112863)

[11. References 53](#_Toc24112864)

# Preface

이 장에서는 예상 독자를 정의하고, 각 장의 내용을 요약한다. 또한 현재 version과 이전 version의 차이점에 대해 설명한다. 하지만 본 문서는 초기 버전이기 때문에 이 부분을 생략한다.

## Readership

본 문서는 다양한 독자에게 읽힐 것이라고 상정하고 있다. 따라서 각 부분을 서술하는 데 있어 어떠한 독자층을 상정하고 있는지를 설명한다.

## Document Structure

### Introduction

본 문서를 서술하는 데 사용된 다양한 다이어그램과 표현 도구들에 대해 설명하고, 본 소프트웨어 프로젝트가 다루는 시스템의 범위에 대해 서술한다.

### System Architecture

시스템과 각 서브시스템의 구조를 개괄적으로 기술하고, 시스템의 전체 기능이 각 서브시스템과 하드웨어에 어떻게 할당되었는지 설명한다.

### Protocol Design

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

### Database Design

Requirements 문서에서 규정된 데이터베이스 요구 사항을 기반으로, 각 데이터 엔티티의 속성과 관계를 ER diagram을 통해 표현하고 최종적으로 Relational Schema, SQL DDL를 작성한다.

### Testing Plan

미리 작성된 Test를 이용해, verification 과 validation을 시행한다. 이 Test 작성에 대한 계획을 설명한다.

### Development Plan

시스템을 구현하는 데 필요한 개발 도구와 프로그래밍 언어, 라이브러리 등의 개발 환경에 대해 설명하고, 시스템 개발 일정을 기술한다.

### Index

본 문서에서 사용된 그림, 표, 다이어그램 등의 색인을 기술한다.

# Introduction

## Objectives

이 장에서는 본 시스템이 등장하게 된 배경과 필요성에 대해 설명하고, 시스템의 대략적인 기능에 대해 설명한다. 또한 해당 시스템을 개발함으로써 기대되는 효과에 대해 설명한다

## Applied Diagram

1. **UML**

UML is a general purpose and developmental modelling language and technique that combines different aspects of a system to represent relations, processes or results of an overall model or system. It is essential to mention that we have used it thoroughly in this document to visualize the workflow of the system.

Since it provides different modelling techniques and a handful subset of diagrams. It can be efficiently used to provide means of communication between developers and users as it covers wide range of symbols and definitions and it consists of the following diagrams: Package Diagram, Deployment Diagram, Class Diagram, State Diagram, Sequence Diagram and ER Diagram.

1. **Package Diagram**

Package diagrams are kind of structural diagrams which show the arrangement and organization of model elements. Package diagram can show both structure and dependencies between sub-systems or modules in a more abstract way than other types of UML diagrams. This abstraction leads to the use of package diagrams in simplifying complex class diagrams by grouping them in packages.

1. **Deployment Diagram**

The deployment diagram describes the physical deployment of information generated by the software program (artifact)on hardware components.

Deployment diagrams are made up of several UML shapes. The three-dimensional boxes, known as nodes, represent the basic software or hardware elements, or nodes, in the system. Lines from node to node indicate relationships, and the smaller shapes contained within the boxes represent the software artifacts that are deployed.

1. **Class Diagram**

It is a diagram that is used to showcase the object classes of a system and the relationship between classes. One of the most fundamental reasons we are using it is because it provides a clear distinction between each class and show the hierarchy and dependency between them.

As far it goes for the inner structure of Class diagram, it consists of some fields indicating some variables, class methods and links or associations between classes.

1. **State Diagram**

State Diagram is a technique to represent different states of a system and all possible next states based on some particular stimuli which triggers the change of the state.

This kind of diagram is very important to analyze different scenarios of the system as the states are represented as nodes and events as arcs which helps in identifying the behavior of the object classes defined in class diagram.

1. **Sequence Diagram**

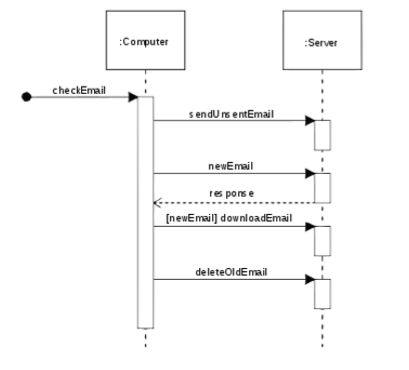


Figure : Example of Sequence Diagram

A sequence diagram to represent the interactions between the actors and objects of the system. To be more specific, the goal of this diagram is present the sequence of interaction and processes that take place in a specific use case instance so that a result could be generated. It is important to notice that the direction of the arrows here is essential to indicate the correct flow of actions.

1. **ER Diagram**



Figure : Example of ER Diagram

ER Diagram 은 Entity 가 가지고 있는 속성과 Entity 간의 관계를 나타낸 다이어그램이다. 이 다이어그램은 주로 데이터베이스를 설계하는 데 사용되며, 해당 다이어그램을 기반으로 Relational Schema, SQL DDL을 작성하게 된다.

## Applied Tool

* 1. **Draw.io**



Figure : Draw.io Logo

Draw.io는 온라인 모델링 툴로서 많은 기본 템플릿과 도형을 제공하기 때문에 사용자가 직접 다이어그램에 사용하기 위해 도형을 만들 필요가 없다. 또한 도형 간 연결선을 간단하게 만들 수 있고, 격자에 위치를 맞출 수 있기 때문에 도형을 정렬하기 편리하다. 이 문서에서 사용된 대부분의 다이어그램은 본 도구로 작성되었다.

* 1. **PowerPoint**



Figure : Powerpoint logo

Powerpoint는 그래픽 프레젠테이션 툴이다. 주로 발표용으로 사용되지만 내장된 도형 작성 기능이 매우 강력하기 때문에 draw.io에서 만들기 힘든 복잡한 다이어그램을 작성하기 위해 사용하였다.

* 1. **ERDPlus**



Figure : ERDPlus Logo

ERDPlus 는 ER Diagram 을 간단한 버튼 클릭으로 생성할 수 있게 해 주는 온라인 툴이다. 적은 노력으로 ER DIagram을 draw.io에 비해 간단하게 만들 수 있기 때문에 ER Diagram을 작성하는 데 사용하였다.

## Project Scope

본 시스템은 상품 평점의 신뢰도가 낮은 상황에서 사용자들이 일일이 직접 리뷰를 읽고 신뢰성을 측정해야 했던 기존의 오픈마켓 서비스들의 단점을 극복하기 위해 리뷰에 자연어 처리 시스템을 도입하여 리뷰의 신뢰성을 판단할 수 있는 기반 지식 없이도 사용자들이 믿을 수 있는 리뷰에 기반한 상품 선택을 도와주는 시스템이다. 본 시스템의 핵심 기능은 리뷰의 분석 기능이며, 해당 기능을 중심으로 각 서브시스템들이 상호작용하도록 설계하였다.

먼저 Frontend System 은 시스템과 사용자와의 상호작용을 담당하며, Backend System Frontend System 에서 오는 데이터 요청에 응답하고 Review Collecting System, Review Analysis System 를 실행시키는 역할을 담당한다. 사용자가 리뷰를 작성하거나 Review Collecting System 에서 정기적으로 오픈마켓에서 리뷰의 목록을 가져오면, Review Analysis System 은 리뷰를 분석해 리뷰의 긍정 평가 정도와 중요도를 측정한다. Review Analysis System이 분석한 리뷰를 데이터베이스에 전달하면, Item Ranking System과 Recommendation System은 추가된 정보를 바탕으로 각 상품의 평가 수치와 추천 카테고리를 수정한다.

# System Architecture – Overall

## Objectives

이번 장 에서는 본 시스템의 구조에 대해서 설명한다.

본 시스템의 전체적인 구조를 다이어그램으로 그리고 다이어그램에 포함된 각 서브시스템과 서브시스템 간의 상호작용에 대해서 설명한다.

## System Organization

* + 1. Overall System Architecture

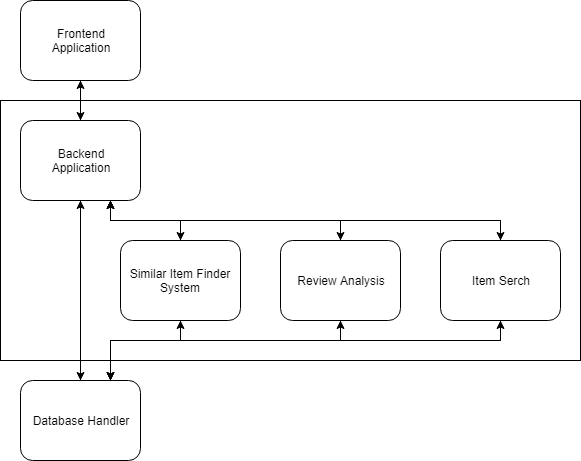


Diagram : Overall System Architecture

시스템은 크게 Frontend Application, Backend Application, Database 로 나누어져 있다.

Frontend Application은 사용자로부터 입력을 받아 Backend Application에 필요한 정보 혹은 정보 수정을 요청한다. 두 Application간의 통신은 HTTP 프로토콜 위에서 JSON 혹은 XML의 형식으로 이루어진다. Backend Application으로부터 필요한 정보를 받으면, Frontend Application은 이 정보를 이용해 사용자에게 보여지는 페이지를 구성하고 사용자에게 전달한다.

Backend Application은 Frontend Application에서 요청한 정보를 제공하거나 정보를 수정한다. 이런 요청들은 Backend Application 에서 실행되거나, 복잡한 요청의 경우 하위 시스템에 의해 실행된다. 각 시스템들은 데이터베이스와 통신하기 위해 Database Handler를 이용한다.

Database는 사용자에게 보여지는 페이지를 구성하는데 필요한 정보들을 저장하고 있다. Backend Application의 요청에 따라 정보의 추가, 수정, 삭제, 조회가 이루어진다.

.

* + 1. Frontend Architecutre

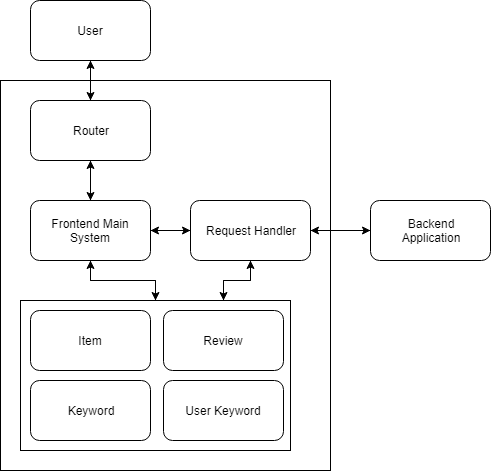
****

Diagram 2 Frontend Architecture

Frontend는 사용자와 직접 상호작용을 하는 시스템을 말한다. Frontend Main System은 사용자의 입력을 받고 적절한 페이지를 만들어 사용자에게 전송한다. Item, Review, Keyword, User Keyword 등 페이지를 구성하는데 필요한 정보는 Backend Application을 통해 제공받으며, Backend와의 통신은 Request Handler을 통한다. Request Handler는 REST API를 이용해 Backend Application과 정보를 주고 받는다.

* + 1. Backend Architecture

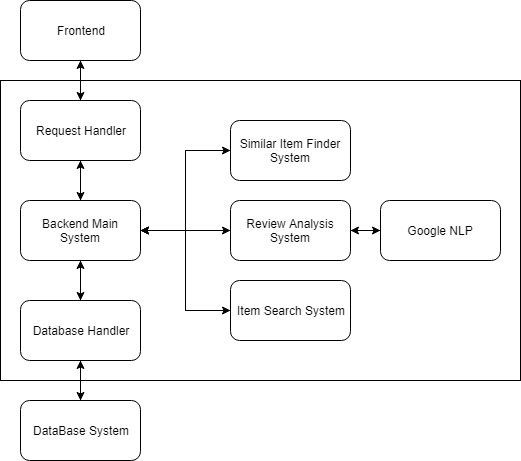


Diagram 3 Backend Architecture

Backend Application은 사용자와 직접 상호작용을 하지 않지만, Frontend Application이 요청하는 정보를 제공하는 방식으로 페이지 구성을 돕는다. 필요한 정보를 제공하기 위해 데이터베이스에서 Item, Review 등을 조회하며, 정보를 가공하기도 한다. 유사한 상품 검색, 리뷰 분석 등 복잡한 작업은 하위 시스템이 전담하여 처리하며, 리뷰 분석의 경우 Google NLP API를 이용한다.

# System Architecture – Frontend

## Objective

전체 시스템 아키텍처 중 사용자와의 상호작용을 담당하는 프론트엔드 시스템의 구조와 각 컴포넌트의 구성, 컴포넌트간의 관계를 서술한다.

## Subcomponents

### Customized Search Result

1. Class Diagram

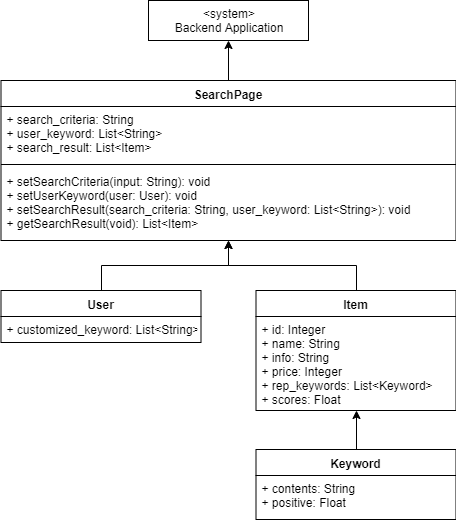


Diagram : System Architecture - Frontend – Customized Search Result

* 1. SearchPage – 검색 페이지 객체

1. attributes

* search\_criteria: SearchPage 객체에서 아이템을 검색할 때 기준이 되는 단어
* user\_keyword: user가 사전에 설정한 선호 키워드 목록
* search\_result: 아이템 검색 후, 해당하는 아이템에 대한 목록

1. methods

* setSerachCriteria(input: String): 유저가 입력한 검색어로 search\_criteria를 설정한다.
* setUserKeyword(user: User): user 객체에서 유저가 사전에 설정한 선호 키워드를 가져와 user\_keyword에 저장한다.
* setSearchResult(search\_criteria: String, user\_keyword: List<String>): 검색 조건에 해당하는 아이템 목록을 가져오고, 유저 키워드에 따라 상품 키워드를 정렬한 후 search\_result에 저장한다.
* getSearchResult(void): search\_result에 저장된 값을 반환한다.
  1. User – 유저 객체

A. attributes

* customized\_keyword: 유저가 사전에 설정해둔 선호 키워드 목록

B. 기타사항

* 실제 User 객체는 그림보다 더 많은 속성 값을 가지고 있지만 사용되어지는 속성만을 표현했다.
  1. Item – 상품 객체

A. attributes

* id: 상품 ID
* name: 상품 이름
* info: 상품 정보
* price: 상품 가격
* rep\_keywords: 상품 대표 키워드 목록
* scores: 상품 평점

B. 기타사항

* 실제 Item 객체는 그림보다 더 많은 속성 값을 가지고 있지만 사용되는 속성만을 표현했다.
  1. Keyword – 키워드 객체

A. attributes

* contents: 키워드 내용
* positive: 리뷰 긍정도

1. Sequence Diagram

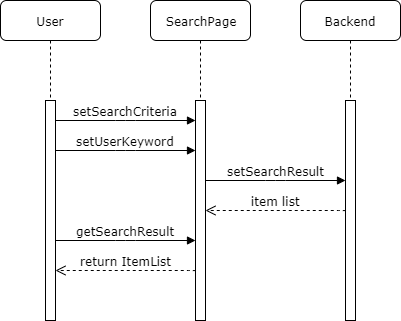


Diagram : System Architecture – Frontend - Customized Search Result - sequence

### Item Page

1. Class Diagram

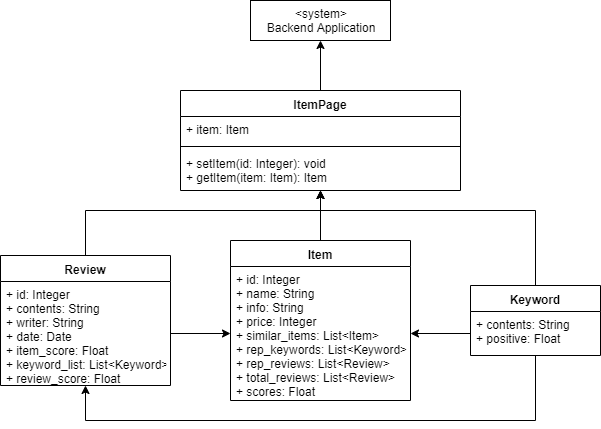


Diagram : System Architecture – Frontend – Item Page

1. ItemPage – 아이템 상세 페이지 객체
   1. Attributes
      * item: 상세 페이지에서 보여줄 아이템 객체
   2. methods
      * setItem(id: Integer): 유저가 선택한 아이템을 id 조회를 통해 item에 설정
      * getItem\*(item: Item): getItem으로 시작하는 모든 함수는 item에 저장되어 있는 객체에서 정보를 추출
2. Review – 리뷰 객체
   1. Attributes
      * id: 리뷰의 id
      * contents: 리뷰 내용
      * writer: 리뷰 작성자
      * date: 리뷰 작성 날짜
      * item\_score: 리뷰 작성자가 매긴 아이템의 점수
      * keyword\_list: 리뷰의 키워드 목록
      * review\_score: 리뷰 자체 점수
3. Item – 상품 객체

A. attributes

* id: 상품 ID
* name: 상품 이름
* info: 상품 정보
* price: 상품 가격
* similar\_items: 해당 상품과 관련 있는 상품 목록
* rep\_keywords: 상품 대표 키워드 목록
* rep\_reviews: 상품 대표 리뷰 목록
* total\_reviews: 상품 전체 리뷰 목록
  + - scores: 상품 평점

1. Keyword – 키워드 객체
   1. Attributes
      * contents: 키워드 내용
      * positive: 리뷰 긍정도

2. Sequence Diagram

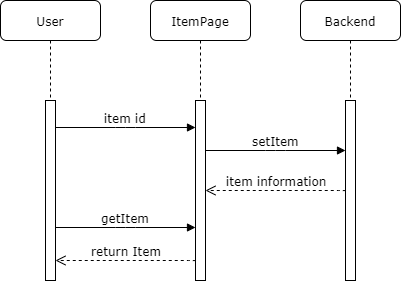


Diagram : System Architecture – Frontend – Item Page - Sequence

### Keyword Visualizer

1. Class Diagram

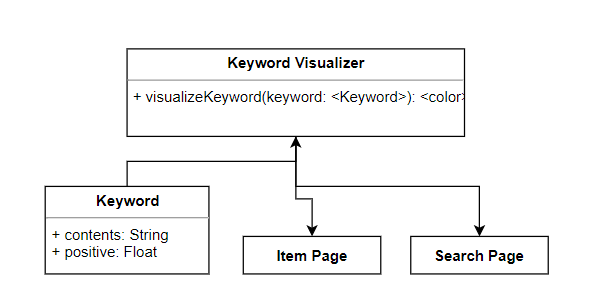


Diagram : System Architecture - Frontend – Keyword Visualizer

1. Keyword Visualizer – 키워드 시각화 객체
2. attributes
3. methods

* visualizeKeyword(keyword: <Keyword>): 키워드의 positive value를 이용해서 키워드의 긍정도를 색으로 시각화해준다. Item Page와 Search Page에서 키워드를 표시할 때에 사용된다.

1. Keyword – 키워드 객체
   * + - 1. attributes

* contents: 키워드 내용
* positive: 리뷰 긍정도

1. Item Page – 상품 페이지 객체

A. 기타사항

* 실제 상품 페이지 객체를 추상화해서 해당 페이지에서 키워드 시각화 기능이 사용됨만을 나타냈다.

1. Search Page – 검색 페이지 객체

A. 기타사항

* 실제 검색 페이지 객체를 추상화해서 해당 페이지에서 키워드 시각화 기능이 사용됨만을 나타냈다.

1. Sequence Diagram

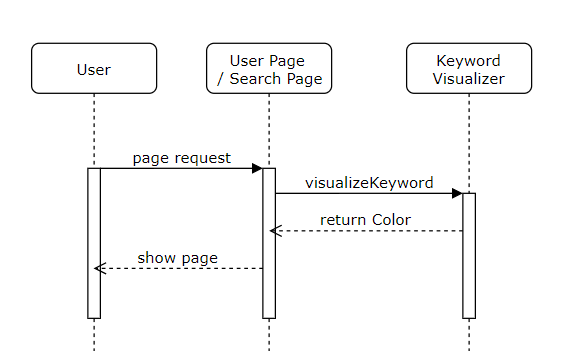


Diagram : System Architecture – Frontend – keyword Visualizer - sequence

### Review Page

1. Class Diagram

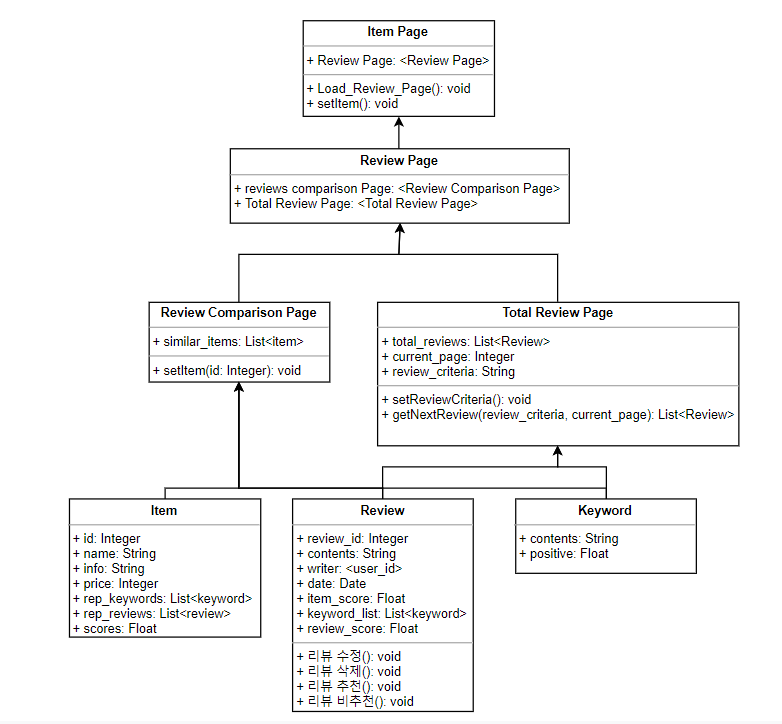


Diagram : System Architecture - Frontend – Review Page

1. Item Page – 상품 페이지
2. attributes

* Review Page: 상품에 해당하는 리뷰 정보 페이지

1. methods

* Load\_Review\_Page(): 상품의 리뷰 페이지를 로드한다.
* setItem(): 상품 페이지에서 정보를 표시할 대상 상품을 설정한다.

1. 기타사항

실제 Item Page 객체를 Review Page입장에서 필요한 요소로만 나눠서 표현하였다.

1. Review Page – 리뷰 페이지 객체

A. attributes

* Reviews comparison Page: 유사상품 리뷰 비교 페이지
* Total Reviews Page: 전체 리뷰 페이지

B. 기타사항

* Review Page가 수행하는 두 가지 기능을 두 가지 객체로 나눠서 표현한다.

1. Review Comparison Page – 리뷰 비교 페이지

A. attributes

* Similar\_items: 유사 상품 목록

B. methods

* setItem(id: integer): 사용자가 관심 유사상품을 클릭하면 해당 상품의 상품 페이지로 넘어간다.

B. 기타사항

* 상품 페이지에서 원래 갖고 있는 속성, method들을 리뷰 페이지 관점으로 구체화해서 정의했다.

1. Total Review Page – 전체 리뷰 페이지
   1. attributes

* Total\_reviews: 전체 리뷰 목록
* Current\_page: 현재 리뷰 페이지 위치 (한번에 모든 리뷰를 요청하지 않고 사용자가 원하는 리뷰만 요청해서 자원 활용을 효율적으로 한다.)
* Review\_criteria: 리뷰 요청 시의 기준을 말한다. 특정 키워드 혹은 평점 순 등의 기준이 있다.
  1. methods
* setReviewCriteria(): 리뷰 요청 시의 기준을 설정한다.
* getNextReview(review\_criteria, current\_page): 설정한 리뷰 요청 기준으로 더 볼 리뷰를 불러온다. 키워드 별 리뷰 모아보기, 평점 높은 리뷰 순으로 보기, 리뷰 더 보기 기능을 구현할 수 있다.
  1. 기타사항
* 상품 페이지에서 원래 갖고 있는 속성, method들을 리뷰 페이지 관점으로 구체화해서 정의했다.

1. Item – 상품 객체

A. attributes

* id: 상품 ID
* name: 상품 이름
* info: 상품 정보
* price: 상품 가격
* rep\_keywords: 상품 대표 키워드 목록
* scores: 상품 평점

B. 기타사항

* 실제 Item 객체는 그림보다 더 많은 속성 값을 가지고 있지만 사용되는 속성만을 표현했다.

1. Review – 리뷰 객체
   * + - 1. attributes

* review\_id: 리뷰 아이디
* contents: 리뷰 내용
* writer: 작성자
* date: 마지막 수정 시각
* item\_score: 상품 평점
* keyword\_list: 키워드 목록
* review\_score: 리뷰 추천수
  + - * 1. methods
* 리뷰 수정( ): 작성자가 리뷰를 수정한다.
* 리뷰 삭제( ): 작성자가 리뷰를 삭제한다.
* 리뷰 추천( ): 자신이 구매한 상품의 리뷰를 최대 한번 추천한다. 추천수를 1 늘린다.
* 리뷰 비추천( ): 자신이 구매한 상품의 리뷰를 최대 한번 비추천한다. 추천수를 1 줄인다.

1. Keyword – 키워드 객체

A. attributes

* contents: 키워드 내용
* positive: 리뷰 긍정도

1. Sequence Diagram

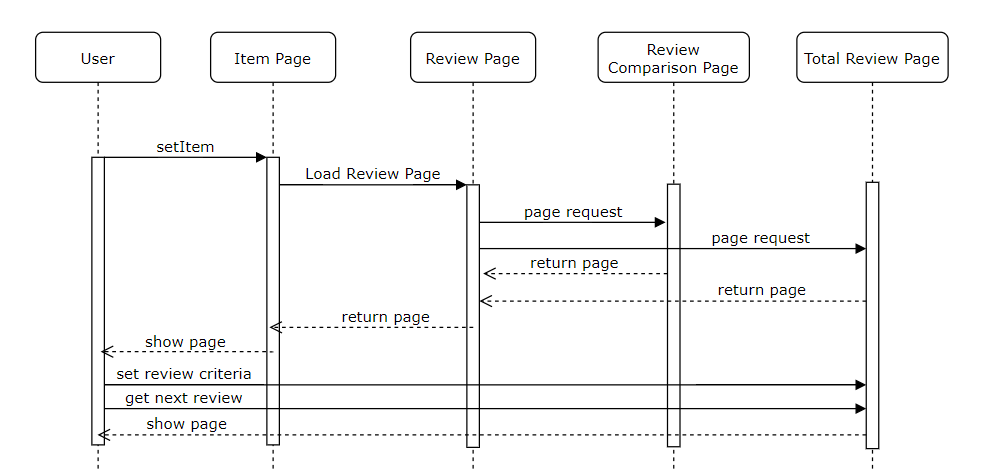


Diagram : System Architecture – Frontend - Review Page - sequence

### Review Post

1. Class Diagram

스크린샷이(가) 표시된 사진

자동 생성된 설명

Diagram : System Architecture – Frontend - Review Post

* 1. ReviewPoster : 리뷰 등록 객체
     1. Methods
        + PostReview(review: Review, item Item)

입력으로 들어온 리뷰를 상품의 리뷰로 등록 시킨다.

* 1. Review : Review 객체
     1. Attributes
        + Contents:사용자가 직접 입력한 리뷰 내용
        + User: 해당 리뷰를 등록한 유저
        + 기타사항: Review에는 keyword 등의 attribute가 더 있으나, 등록 시점에는 설정되지 않으므로 클래스 다이어그램에서 제외하였음
  2. User : 리뷰를 등록한 유저 객체
     1. Attributes
        + userId: 사용자의 ID
        + 기타사항: Review에 사용자의 다른 정보가 필요하지 않으므로 다른 attribute를 적지 아니하였음
  3. Item – 리뷰의 대상이 되는 상품
     1. Attributes
        + itemId: 상품의 ID
        + name: 상품의 이름
        + reviews: 해당 상품에 작성된 리뷰들의 리스트

### User Page

1. Class Diagram

스크린샷이(가) 표시된 사진

자동 생성된 설명

Diagram : System Architecture – Frontend - User Page

* 1. User Page
     1. Attributes
        + User: 해당 사용자 페이지를 방문하는 사용자 객체
     2. Methods
        + enterUserPage(password: String):해당 유저의 유저페이지에 접근하기 위해 비밀번호를 다시 입력하여 사용자를 인증. 인증 실패 시 접근 불가.
        + changePassword(password String): 사용자 계정의 비밀번호를 바꿈
        + changeKeyword(keywords: Lisng<Keyword>):사용자의 선호 키워드의 순서, 내용등을 바꿈
  2. User
     1. Attributes
        + userId: 사용자의 ID
        + name: 표시 될 사용자의 이름
        + password: 사용자의 비밀번호. 이 값은 해시 된 형태로 저장된다.
        + customized\_keyword: 사용자가 사전에 설정한 키워드들의 목록. 사용자가 중요하게 생각하는 순서대로 정렬되어 있다.
  3. Keyword
     1. Attributes
        + Contents: keyword의 이름. 리뷰 분석을 통해 얻어지며, 사용자가 중요하게 생각하는 내용과 관련이 있다.
        + Score: Keyword의 점수. 높을수록 리뷰에서 좋은 뜻으로 쓰였음을 의미하지만 사용자 페이지 기능에서는 쓰이지 않는다.

# System Architecture – Backend

## Objectives

## Overall Architecture

## Subcomponents

### Login

1. Class Diagram

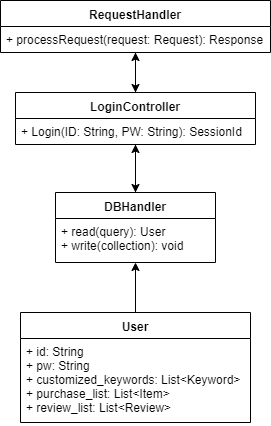


Diagram : System Architecture – Backend – Login

* + - 1. RequestHandler – request 처리 객체
         1. methods

processRequest(request: Request): request 요청이 오면 요청받은 내용에 따라 알맞은 controller에 넘겨주는 메서드. 요청을 처리한 후 적절한 값을 반환

* + - 1. LoginController – 로그인 처리 객체
         1. methods

Login(ID: String, PW: String): ID와 PW를 이용해 database에 해당 유저 조회를 요청하는 메서드

* + - 1. DBHandler – database 제어 객체
         1. Methods

Read(query): 요청받은 qeury문을 사용해 databse에서 조회, User 객체를 반환

Write(collection): colleciton에 있는 데이터를 database에 저장

* + - 1. User – 유저 객체

1. Sequence Diagram

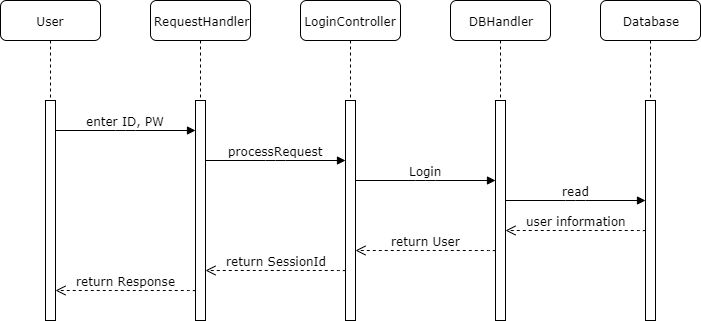


Diagram : System Architecture – Backend – Login - Sequence

### B. Review Analysis

1. Class Diagram

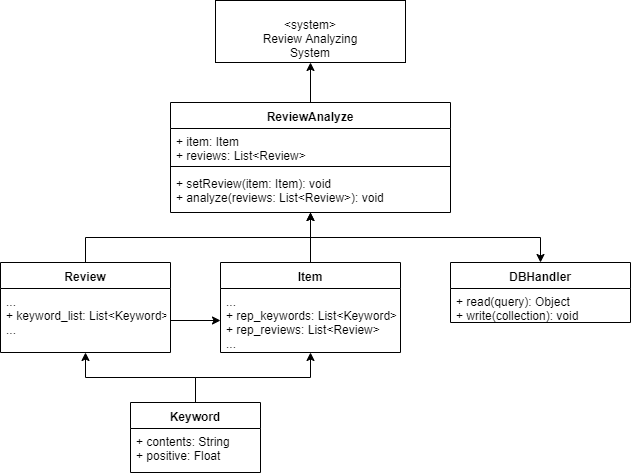


Diagram : System Architecture – Backend – Review Analysis

* + - 1. ReviewAnalyze – Review 분석 객체
         1. Attributes

item: 리뷰 분석을 진행할 상품

reviews: 상품에서 추출한 리뷰 목록

* + - * 1. methods

setReview(item: Item): 상품에서 리뷰 목록을 추출해 reviews에 저장

analyze(reviews: List<Review>): Google NLP API를 사용해 리뷰를 분석, 분석 후 변경된 정보를 database에 저장

* + - 1. DBHandler – database 제어 객체
         1. Methods

Read(query): 요청받은 qeury문을 사용해 databse에서 조회

Write(collection): colleciton에 있는 데이터를 database에 저장

1. Sequence Diagram

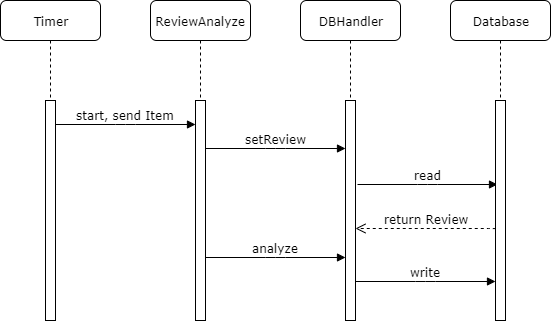


Diagram : System Architecture – Backend – Review Analysis - Sequence

### C. Similar Item Finder

1. Class Diagram

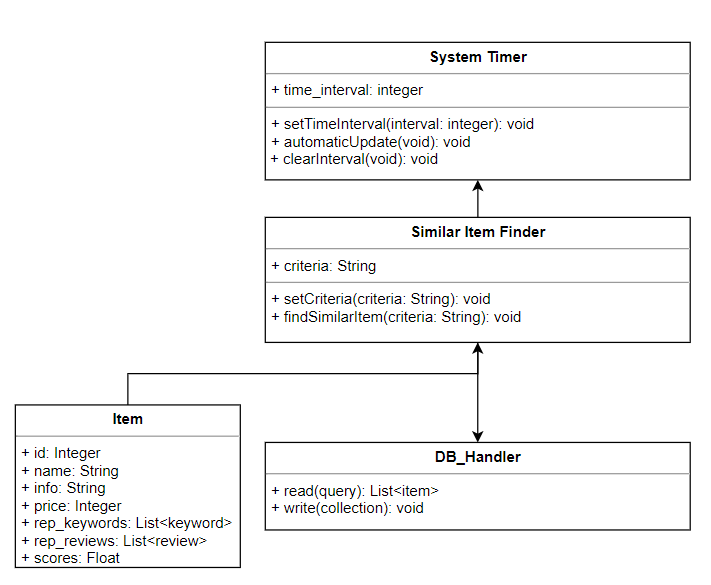


Diagram : System Architecture – Backend – Similar Item Finder

* + - 1. System Timer: 시스템 자동 실행 시스템
         1. Attributes

time\_interval: 자동 실행 주기

* + - * 1. methods

setTimeInterval(interval: Integer): 자동 실행 주기 설정

automaticUpdate(void): 일정 주기마다 update 진행, Similar Item Finder 실행

clearTimeInterval(void): 자동 실행 취소

* + - 1. Similar Item Finder: 유사 상품 탐색 시스템
         1. Attributes

criteria: 유사 상품 탐색의 기준

* + - * 1. Methods

setCriteria(criteria:String): 유사 상품 탐색의 기준을 설정

findSimilarItem(criteria:String): criteria를 기준으로 유사 상품을 탐색하고 상품 DB에 업데이트한다.

* + - 1. DBHandler – database 제어 객체
         1. Methods

Read(query): 요청받은 qeury문을 사용해 databse에서 조회

Write(collection): colleciton에 있는 데이터를 database에 저장

* + - 1. Item – 상품 객체

A. attributes

* id: 상품 ID
* name: 상품 이름
* info: 상품 정보
* price: 상품 가격
* rep\_keywords: 상품 대표 키워드 목록
* scores: 상품 평점

B. 기타사항

* 실제 Item 객체는 그림보다 더 많은 속성 값을 가지고 있지만 사용되는 속성만을 표현했다.

1. Sequence Diagram

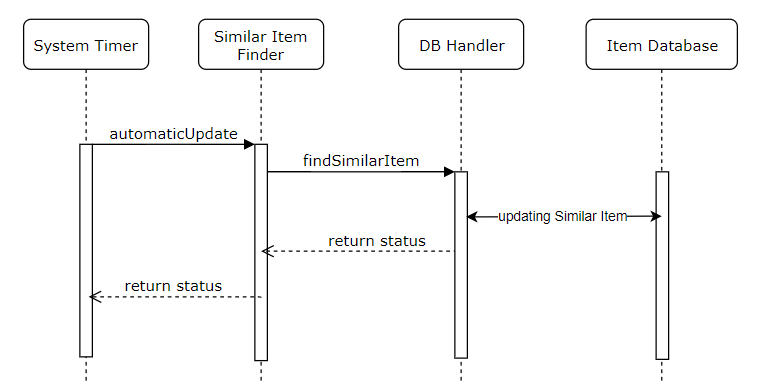


Diagram : System Architecture – Backend – Similar Item Finder - Sequence

### D. 상품 정보 제공

1. Class Diagram

스크린샷이(가) 표시된 사진

자동 생성된 설명

Diagram : System Architecture – Backend – 상품 정보 제공

* 1. RequestHandler
     1. Methods
        + processRequest(request: Request): 사용자 Frontend에서 들어오는 요청을 처리하고 그에 따른 응답을 사용자 Frontend로 보내는 역할. 상품 정보 제공에서는 상품에 대한 요청이 들어왔을 때 DB로부터 상품정보를 받아 해당 정보를 응답으로 보낸다.
  2. DBHandler
     1. Methods
        + getItem(query): 상품을 DB에서 검색하여 Item 객체를 반환한다.
        + putItem(query, input: List<Item>): 상품 객체를 DB에 등록하는 기능. 그러나 이 기능에서는 쓰이지 않는다.
        + 기타사항: 그 외에 다른 method들이 많으나 해당 기능과 관련이 적어 적지 아니하였다.
  3. Item
     1. Attributes
        + Item\_id: 상품의 ID
        + name: 상품의 이름
        + price: 상품의 가격
        + keywords: 상품이 가지고 있는 keyword들의 목록
        + img\_src: 상품의 사진 주소
  4. Keywords
     1. Attributes
        + Contents: keyword의 이름. 리뷰 분석을 통해 얻어지며, 사용자가 중요하게 생각하는 내용과 관련이 있다.
        + Score: Keyword의 점수. 높을수록 리뷰에서 좋은 뜻으로 쓰였음을 의미한다.

### E. 개인정보 변경

1. Class Diagram

스크린샷이(가) 표시된 사진

자동 생성된 설명

Diagram : System Architecture – Backend – 개인 정보 변경

* 1. RequestHandler
     1. Methods
        + processRequest(request: Request): 사용자 Frontend에서 들어오는 요청을 처리하고 그에 따른 응답을 사용자 Frontend로 보내는 역할. 개인정보 변경 기능에서는 비밀번호 또는 사용자 설정 키워드를 변경하고자 하는 응답을 받아 처리하게 된다.
  2. DBHandler
     1. Methods
        + getUser(query): 상품을 DB에서 검색하여 User 객체를 반환한다.
        + putUser(query, input: List<Item>): User 객체를 DB에 등록하는 기능. 이 기능에서는 단순 등록이 아닌 기존에 있는 User 객체를 Update하는 것을 의미한다.
        + 그 외에 다른 method들이 많으나 해당 기능과 관련이 적어 적지 아니하였다.
  3. User
     1. Attributes
        + userId: 사용자의 ID
        + name: 표시 될 사용자의 이름
        + password: 사용자의 비밀번호. 이 값은 해시 된 형태로 저장된다.
        + customized\_keyword: 사용자가 사전에 설정한 키워드. 사용자가 중요하게 생각하는 순서대로 정렬되어 있다.
  4. Keywords
     1. Attributes
        + Contents: keyword의 이름. 리뷰 분석을 통해 얻어지며, 사용자가 중요하게 생각하는 내용과 관련이 있다.
        + Score: Keyword의 점수. 높을수록 리뷰에서 좋은 뜻으로 쓰였음을 의미하지만 개인정보 변경 기능에서는 쓰이지 않는다.

# Protocol Design

이 장에는 하드웨어 요구사항, 데이터베이스 요구사항, 개발환경 요구사항 등 개발 중인 시스템과 관련된 항목의 상세하고 구체적인 정보가 서술된다.

## Objectives

## REST API

## JSON

## Details

# Database Design

## Objectives

## ER Diagram

## JSON

# Testing Plan

## Objectives

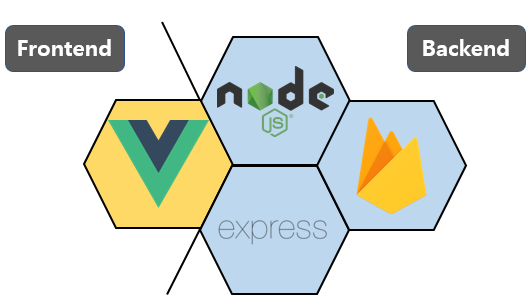
## ER Diagram

# Development Plan

## Objectives

본 챕터에서는 개발 시 실제로 사용할 플랫폼과 환경에 대해 설명할 것이다. 각 플랫폼의 특징에 대해 설명하고 최종적으로 어떤 구조를 가지게 되는지 기술한다. 또한 개발 일정을 간트 차트 형식을 통해 기술할 것이다.

## Overall Framework Architecture



우리 서비스에서 최종적으로 사용할 프레임워크 구조이다. 프론트엔드로는 vue.js를 사용했으며 백엔드로는 Node.js, Express, Firebase를 사용할 것이다. 각각의 프레임워크에 대한 설명을 아래 파트에 상세히 기술했다.

## Frontend Environment

### Vue.js



Vue.js는 최근 계속해서 상승세를 타고 있는 프론트엔드 자바스크립트 프레임워크이다. Angular, React에 비해 매우 작고 가벼우며 복잡도도 낮기 때문에 사용하기에 매우 간편하고 처음 배우는 사람이 시작하기에 다른 프레임워크보다 쉽다. 또한 성능이 다른 프론트엔드 프렘워크에 비해 우수하고 빠른편이며, Angular의 장점과 React의 장점을 모두 가지고 있다.

Vue.js의 특징은 다음과 같다.

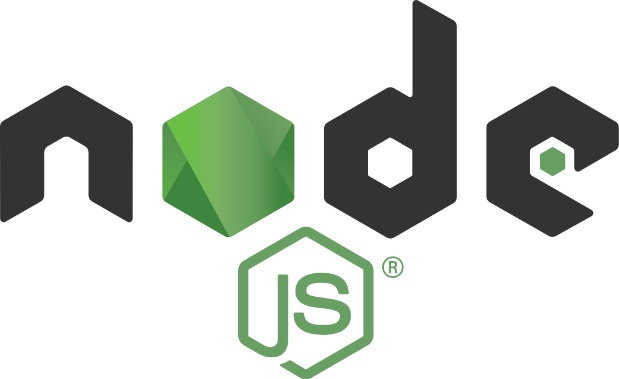
Vue.js는 MVVM(Model-View-ViewModel) 패턴을 사용한다. MVVM 패턴이란 화면을 모델(Model) – 뷰(View) – 뷰 모델(ViewModel)로 구조화하여 개발하는 방식이다. 여기서 모델은 어플리케이션에서 사용되는 데이터와 그 데이터를 처리하는 부분이며, 뷰는 사용자에게 보여지는 UI 부분, 마지막으로 뷰 모델은 뷰를 표현하기 위해 만든 뷰를 위한 모델이다. 뷰를 나타내 주기 위한 모델이자 뷰를 나타내기 위한 데이터를 처리하는 부분인 것이다. 이 MVVM패턴을 사용함으로써 뷰와 모델 사이의 의존성을 없애 각각을 모듈화하여 개발할 수 있게 된다.

Vue.js는 컴포넌트 기반의 프레임워크이다. 각각의 컴포넌트는 HTML, CSS, 자바스크립트를 모두 가지고 있어 코드 집적도와 유지보수성을 높일 수 있으며, 코드의 재사용성을 높일 수 있다.

이외에 가상돔(Virtual DOM) 렌더링 방식, 양방향 데이터 바인딩 등 많은 좋은 특징을 가지고 있다.

## Backend Environment

### Node.js



Node.js는 구글 크롬의 자바 크롬의 자바스크립트 엔진(V8 Engine)으로 빌드 된 자바스크립트 런타임이다. 이 이야기를 쉽게 하면 다음과 같다. 자바스크립트는 일반적으로 크롬과 같은 브라우저에 내장되어 있다. 그래서 자바스크립트는 브라우저에 종속되어 있었지만 이를 크롬 같은 브라우저에서만 쓰는 것이 아닌 브라우저 밖, 즉 내 컴퓨터에서 다양한 용도로 확장하기 위해 만들어진 것이 바로 Node.js이다. Node.js를 이용하면 python과 같이 내 컴퓨터에서 File System을 이용할 수도 있고, 서버를 만들 수도 있고, 크롤링도 할 수 있게 된다.

Node.js의 특징은 다음과 같다.

비동기식 I/O 처리를 지원한다. Node.js의 모든 라이브러리는 비동기식(async), 즉 Non-blocking 방식으로 작업을 처리한다. 하나의 API가 실행되고 데이터를 반환할 때까지 기다리는 것이 아니라 바로 다음 API를 계속해서 실행하는 것이다. 만약 이전에 실행했던 API가 결과값을 반환한다면 Node.js의 이벤트 알림 메커니즘을 통해 결과값을 받아오게 된다.

이 외 구글 크롬의 자바스크립트 엔진으로 인한 빠른 속도, 단일 쓰레드 사용, 뛰어난 확장성 등의 특징을 가지고 있다.

### Express



Express는 Node.js의 개발을 더 쉽게 하기위해 만들어진 간편한 웹 프레임워크이다. Node.js의 핵심 모듈인 http와 Connect 컴포넌트를 기반으로 한다. 쉽게 말하면 Node.js개발 시 개발을 빠르고 손쉽게 할 수 있도록 도와주는 역할을 한다. Python의 Django, java의 spring과 같은 역할을 한다고 생각하면 된다.

### Firebase



Firebase는 웹과 모바일 개발에 필요한 기능을 제공하는 BaaS(Backend as a Service)이다. 쉽게 말해 백엔드 개발을 통해 서버를 따로 설계, 구현하지 않고도 클라우드와 연동해 웹, 모바일 응용 프로그램의 프론트엔드 개발에 집중할 수 있도록 도와주는 서비스이다. 즉 하나의 서버를 개발하기 위한 인증, 데이터베이스, 푸시 알람, 스토리지, API 등을 따로 개발하지 않아도 Friebse가 이 모든 기능을 프로젝트 구축 시 자동적으로 만들어 주는 것이다. 또한 서버를 구축하기 위해 리눅스 명령어를 알 필요도 없으며 도메인을 구입할 필요도 없고 개발하는 동안에는 서버를 구입할 필요도 없게 해준다. Firebase가 가지고 있는 대표적인 기능은 다음과 같다.

* 실시간 데이터베이스
* 간편한 사용자 인증
* 클라우드 저장소
* 호스팅
* 앱 테스트
* 성능 모니터링
* 오류 보고

이 외에도 많은 기능을 가지고 있으며, 이는 Firebase 프로젝트를 생성하기만 하면 모두 사용할 수 있다.

## Schedule

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1. 항목 | | 11월 | | | 12월 | |
| 3주차 | 4주차 | 5주차 | 1주차 | 2주차 |
| 프론트엔드 | 쇼핑몰 첫 페이지 |  |  |  |  |  |
|  | 검색 결과 페이지 |  |  |  |  |  |
|  | 마이 페이지 |  |  |  |  |  |
|  | 상품비교 페이지 |  |  |  |  |  |
| 백엔드 | 로그인 |  |  |  |  |  |
|  | 개인정보 수정 |  |  |  |  |  |
|  | 상품 검색 |  |  |  |  |  |
|  | 연관상품 찾기 |  |  |  |  |  |
|  | 리뷰 분석 |  |  |  |  |  |
| 시스템 통합/ 테스트 | |  |  |  |  |  |
| 최종 발표 준비 | |  |  |  |  |  |

앞으로의 일정은 위의 표와 같다. 프론트엔드와 백엔드 개발을 병렬적으로 진행해 개발 속도를 높일 것이다.

# 10. Index

## 10.1. Tables

[Table 1: Glossary 17](#_Toc23624792)

[Table 2: Customized Search result 30](#_Toc23624793)

[Table 3: Item Page 31](#_Toc23624794)

[Table 4: Key word 31](#_Toc23624795)

[Table 5: Review Page 32](#_Toc23624796)

[Table 6: Review Post 33](#_Toc23624797)

[Table 7: User Page 33](#_Toc23624798)

[Table 8: Login Function 34](#_Toc23624799)

[Table 9: Review Analysis System 35](#_Toc23624800)

[Table 10: 연관상품 finder system 35](#_Toc23624801)

[Table 11: 품 정보 제공 36](#_Toc23624802)

[Table 12: 개인정보 변경 시 기능 36](#_Toc23624803)

[Table 13: 로그인 44](#_Toc23624804)

[Table 14: 사용자 정보 수정 44](#_Toc23624805)

[Table 15: 리뷰 작성, 키워드 분석 45](#_Toc23624806)

[Table 16: Customized Search-1 45](#_Toc23624807)

[Table 17: Customized Search-2 46](#_Toc23624808)

[Table 18: 연관 상품 Update 46](#_Toc23624809)

[Table 19: 리뷰 추가 요청 47](#_Toc23624810)

[Table 20: 상세정보 확인 47](#_Toc23624811)

## 10.2. Figures

[Figure 1: 온라인쇼핑 거래액 전년동월대비 증감률 10](#_Toc23624812)

[Figure 2: 온라인쇼핑 거래액 동향 11](#_Toc23624813)

[Figure 3: 1회 평균 쇼핑 시간 13](#_Toc23624814)

[Figure 4: Review Revolution - 모아 보기 기능 14](#_Toc23624815)

[Figure 5: Review Revolution – 상품 정보 화면 15](#_Toc23624816)

[Figure 6: Customized Search 예시 19](#_Toc23624817)

[Figure 7: item detail page 20](#_Toc23624818)

[Figure 8: review comparison page 21](#_Toc23624819)

[Figure 9: review analysis를 통한 키워드 추출 에시 22](#_Toc23624820)

## 10.3. Diagrams

[Diagram 1: Simple login process 18](#_Toc23624821)

[Diagram 2: Simple sign up process 19](#_Toc23624822)

[Diagram 3: Frontend Architecture 25](#_Toc23624823)

[Diagram 4: Backend Architecture 26](#_Toc23624824)

[Diagram 5: Review Analysis System Architecture 27](#_Toc23624825)

[Diagram 6: Recommendation System Architecture 28](#_Toc23624826)

[Diagram 7: Overall context diagram 41](#_Toc23624827)

[Diagram 8: Overall process diagram 42](#_Toc23624828)

[Diagram 9: Use case diagram 43](#_Toc23624829)

[Diagram 10: DFD of Review Analysis System 48](#_Toc23624830)

[Diagram 11: DFD of Similar Product Finder System 48](#_Toc23624831)

[Diagram 12: Sequence Diagram of Review Submit 49](#_Toc23624832)

# 11. References

KaemingkDiana. (2019). “20 online review stats to know in 2019”. qualtrics: https://www.qualtrics.com/blog/online-review-stats/에서 검색됨

LucaMichael. (2016). “Reviews, Reputation, and Revenue: The Case of Yelp.com”. Harvard Business School: https://www.hbs.edu/faculty/Pages/item.aspx?num=41233에서 검색됨

김준현최지혜, 허준범, 모하메드유도영,. (2019). “requirement.” github: (https://github.com/skkuse/2019spring\_41class\_team5/blob/master/docs/requirement.docx)에서 검색됨

미디어DMC. (2017). “2018 인터넷 쇼핑 행태와 쇼퍼 그룹 및 쇼핑몰 분석 보고서\_요약본.” digieco: https://www.digieco.co.kr/KTData/Board/FILE/PDF/2018%20%EC%9D%B8%ED%84%B0%EB%84%B7%20%EC%87%BC%ED%95%91%20%ED%96%89%ED%83%9C%EC%99%80%20%EC%87%BC%ED%8D%BC%20%EA%B7%B8%EB%A3%B9%20%EB%B0%8F%20%EC%87%BC%ED%95%91%EB%AA%B0%20%EB%B6%84%EC%84%9D%20%EB%B3%B4%EA%B에서 검색됨

통계청. (2019). “2019년 8월 온라인쇼핑 동향.” kostat: http://kostat.go.kr/portal/korea/kor\_nw/3/index.board?bmode=download&bSeq=&aSeq=377803&ord=2에서 검색됨