**UB (pseudonym)**

**Software Requirements Specification**

**and**

**Project Proposal**

**Since June 2015**

**Last Edit 08.03**

**Version 1.3.1**

**Engine**

**강지현**

본 문서에 가칭 UB 프로그램 개발에 관하여 명세한다.

본 문서의 일부 또는 전체에 대해 무단 복제를 금한다.

2015년 여름방학(7월-8월) 중 본 프로젝트를 1차 완성하는 것을 목표로 한다.

INDEX

[1. Introduction 5](#_Toc426782720)

[1.1 Purpose 5](#_Toc426782721)

[1.2 Scope 5](#_Toc426782722)

[언어 변화의 기록장 5](#_Toc426782723)

[민주적 언어 결정 5](#_Toc426782724)

[빠른 언어변화 대응 5](#_Toc426782725)

[1.3 Glossary 5](#_Toc426782726)

[1.4 References 5](#_Toc426782727)

[2. Overall description 6](#_Toc426782728)

[2.x Product Perspective 6](#_Toc426782729)

[2.x System Environment 6](#_Toc426782730)

[2.x Rules and Constraints 6](#_Toc426782731)

[2.x.x Naming Convention 6](#_Toc426782732)

[MYSQL (Database) 6](#_Toc426782733)

[PHP 6](#_Toc426782734)

[HTML 6](#_Toc426782735)

[CSS 6](#_Toc426782736)

[2.x.x Collaboration Rules 6](#_Toc426782737)

[2.x Architectural Design 7](#_Toc426782738)

[2.x.x MVC Framework 7](#_Toc426782739)

[2.x.x Application Detached from System 7](#_Toc426782740)

[2.x URL Routing Outline 7](#_Toc426782741)

[2.x.x URL Contract 7](#_Toc426782742)

[2.x Database Design 7](#_Toc426782743)

[2.x.x Table Diagram 7](#_Toc426782744)

[2.x.x Table Description 8](#_Toc426782745)

[member 8](#_Toc426782746)

[term 8](#_Toc426782747)

[vote 8](#_Toc426782748)

[comment 9](#_Toc426782749)

[message 9](#_Toc426782750)

[3. System Features 9](#_Toc426782751)

[3.1 Object and Flow Diagram 9](#_Toc426782752)

[3.2 System Components Overview 10](#_Toc426782753)

[3.2.1 Gateway and Configuration 10](#_Toc426782754)

[Gateway (root/index.php) 10](#_Toc426782755)

[Config.xml 10](#_Toc426782756)

[3.2.2 Core 10](#_Toc426782757)

[Core.php 10](#_Toc426782758)

[Router.php 11](#_Toc426782759)

[Controller.php 11](#_Toc426782760)

[Model.php 11](#_Toc426782761)

[Database.php 12](#_Toc426782762)

[Data.php (ABORT) 12](#_Toc426782763)

[3.2.3 Controller 12](#_Toc426782764)

[Root.php 12](#_Toc426782765)

[Term.php 12](#_Toc426782766)

[About.php 13](#_Toc426782767)

[Rules.php 13](#_Toc426782768)

[Guide.php 13](#_Toc426782769)

[Notice.php 13](#_Toc426782770)

[Archive.php 14](#_Toc426782771)

[Login.php 14](#_Toc426782772)

[3.2.4 Model 14](#_Toc426782773)

[Term\_md.php 14](#_Toc426782774)

[Comment\_md.php 15](#_Toc426782775)

[Member\_md.php 15](#_Toc426782776)

[Vote\_md.php 16](#_Toc426782777)

[SNSBrowser\_md.php 16](#_Toc426782778)

[Notice\_md.php 16](#_Toc426782779)

[Archive\_md.php 16](#_Toc426782780)

[3.2.5 View 17](#_Toc426782781)

[3.3 Functional Requirements 17](#_Toc426782782)

[3.3.1 Account 17](#_Toc426782783)

[Account Registration 17](#_Toc426782784)

[Account Login 17](#_Toc426782785)

[Verification of Signed-Up Status (of an Email) 17](#_Toc426782786)

[Edit Account Information 18](#_Toc426782787)

[Find Account by Resetting Password 18](#_Toc426782788)

[Delete Account 18](#_Toc426782789)

[3.3.x Term 18](#_Toc426782790)

[Search Term by Query String 18](#_Toc426782791)

[View Definition of a Particular Entry 18](#_Toc426782792)

[Add Term 18](#_Toc426782793)

[Show Related Terms 19](#_Toc426782794)

[Show Recently Added Terms 19](#_Toc426782795)

[Edit Term 19](#_Toc426782796)

[Delete Term 19](#_Toc426782797)

[Author Information Expiration 19](#_Toc426782798)

[3.3.x Vote (For Term) 19](#_Toc426782799)

[Like Term 19](#_Toc426782800)

[Dislike Term 19](#_Toc426782801)

[3.3.x Vote (For Comment) 19](#_Toc426782802)

[Like Comment 19](#_Toc426782803)

[Dislike Comment 19](#_Toc426782804)

[3.4 Other Requirements 19](#_Toc426782805)

# 1. Introduction

## 1.1 Purpose

본 문서는 UB (가칭)의 개략적, 그리고 상세한 서술을 위해 작성되었다. 왜 프로젝트가 기획되었는지, 어떤 방식으로, 어떤 기능을 갖출 것인지 설명한다.

## 1.2 Scope

2015년 상반기 현재 국내에 있는 웹 서비스 중, 한국어의 사용의, 더 자세히 말하여, 신조어와 단어의 변화에 대한 경과를 기록하고 토의하는 Platform이 없어, 이와 같은 기능을 할 수 있는 웹 서비스를 개발하기로 하였다.

UB는 신조어와 파생어의 단어를 사용자로부터 제공받는 웹 어플리케이션이다. 사용자는 자신의 의지에 따라, 단어의 의미를 제공할 수 있고 또한 타인이 제시한 의미를 열람할 수 있다. 같은 단어의 의미는 점수 알고리즘에 의거, 점수의 오름차순으로 보여진다.

#### 언어 변화의 기록장

새로이 생성되는 표현 혹은 단어와 그 의미를 기록함에 여러 의미가 있다. 첫째, 언어의 변화 양상을 추적할 수 있다. 언어는 끊임없이 변화한다. 본 서비스는 변화하는 언어에 대한 지식을 언어 사용자 스스로 제공케 하여, 이용자 전체로 하여금 빠르고 효율적인 방법으로 변화 요소를 습득, 인지하게 한다. 또한 이 기록은 훗날 말뭉치로서 언어와 문화연구 등 다방면에 기여할 것이다.

#### 민주적 언어 결정

UB는 특정 권위체에 의해서가 아닌 이용자 불특정 다수로부터, 투표에 의해 의미를 형성하는 방식을 채택한다. 이는 민주적인 방식의 언어 형성을 돕는다. 다시 말해, 표현과 단어들은 공동의 합의에 의해 선택된다.

#### 빠른 언어변화 대응

현재 World Wide Web 과 무선인터넷의 보급이 과거 어느 시점보다 넓게 돼 있다. 디지털 소통 채널은 개인의 언어 사용빈도를 증가시켰고, 또한 채널의 특수성-구두가 아닌 Text를 통한 소통은 다양한 형태의 언어 Variation을 야기하고 있다. 변화가 전보다 매우 빠르다. 이에 대응하여, 전의 “사전”을 이용하는 것에 효용이 있지 못함을 인지할 수 있다. UB는 훌륭한 대안이 될 수 있다.

새로운 사전

잊혀질 권리

## 1.3 Glossary

본 프로젝트 명세에 등장하는 용어와 그 설명을 밝힌다.

|  |  |
| --- | --- |
| **Word** | **Description** |
| localhost | 도메인 혹은 root가 되는 주소이다. |
| root | 웹 서버 최상위 폴더 |
| UB | 본 프로젝트의 가명, 웹 사이트인 동시에 웹 프로그램을 지칭하는 용어이다. |
| singleton | Singleton Form. 객체를 단 한 개만 생성하는 형태의 프로그래밍 기법이다. |
|  |  |

## 1.4 References

추후 서술

# 2. Overall description

## 2.x Product Perspective

## 2.x System Environment

php

mysql

html5

css3

javascript

jquery

ajax

후에 상세서술

## 2.x Rules and Constraints

### 2.x.x Naming Convention

파일은 단일객체만을 가지고 있거나, 아무런 객체도 가지고 있지 않은 것으로 나뉜다. 객체를 가지고 있는 경우, 객체의 이름과 파일 명은 모두 첫 글자를 대문자로 시작한다. Controller의 경우 객체명을 위와 같은 규칙에 의거해 그대로 쓰고, 모델의 경우 말미에 \_md를 붙인다. View프로그램의 경우 첫 글자가 소문자이다.

#### MYSQL (Database)

snake case with first letter lowercase.

singular names.

reference: https://launchbylunch.com/posts/2014/Feb/16/sql-naming-conventions/

ex) test\_id\_power, test2\_member

#### PHP

camelCase

ex) doSomething, variableName

#### HTML

snake case with first letter lower case.

ex) test\_id\_power, test2\_member

#### CSS

snake case with first letter lower case.

ex) test\_id\_power, test2\_member

### 2.x.x Collaboration Rules

협업 간 필요한 규칙을 명세한다.

## 2.x Architectural Design

### 2.x.x MVC Framework

UB는Model View Controller Framework에 입각해 설계한다. Model은 데이터 가공을, View는 화면 출력을, 그리고 Controller는 Model과 VIew를 포함한 전체 Logic을 관장한다.

### 2.x.x Application Detached from System

System과 Application을 분리해서 설계하고, System에 필수적인 기능만 구현한 후, Application이 이를 상속하여 실행될 수 있도록 한다. 이렇게 디자인하는 가장 큰 이유는 요소별 모듈화를 통해 이식을 최대한 효율적으로 하기 위함이다.

## 2.x URL Routing Outline

### 2.x.x URL Contract

URL은 domain/segment1/segment2/segment3,,.형식이다. UB는 요청 받은 URL을 토대로 Routing을 통해 적절한 Contents를 제공한다. URL은 다음과 같이 요소화한다.

segment[0]/segment[1] /segment[2]/,,.

= **domain / CONTROLLER / PARAMETER1 / PARAMETER2,,.**

즉, segment[1] 에 요청될 컨트롤러가 위치하고, 그 이후 Parameter들로 취급한다.

이와 같은 URL은 사용자 직관성을 우선적으로 고려하여 설정하였으며, MVC Framework 기반의 개발과도 맥락을 같이 한다.

## 2.x Database Design

### 2.x.x Table Diagram

*다음에 Database Table Structure를 첨부한다.*

### 2.x.x Table Description

본 프로그램의 Database 에 포함된 테이블 목록을 명시한다.

#### member

Member은 회원 정보를 가지고 있는 테이블이다.

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| *id (PK)* |  |  |
| *email* |  | client의 사이트 로그인 id로 사용 |
| *password* |  | 비밀번호 |
| *name* |  | 사이트에서 사용할 이름. (실명 사용하지 말라고 권고) |
| *intro* |  | 자기 소개 |
| *photo* |  | 잠정적 ABORT |
| *register\_date* |  | 회원가입일 |
| *last\_login* |  | 가장 최근 로그인 일시. |
| *submitted* |  | 제공한 definition의 수 |
| *rating* |  | 사용자의 평판. |
|  |  |  |

#### term

Term은 단어와 의미 정보를 가지고 있는 테이블이다.

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| *id (PK)* |  |  |
| *term* |  | 단어 |
| *pos* |  | 품사 |
| *lemma* |  | lemma |
| *def* |  | 정의 |
| *usage* |  | 용례 |
| *img* |  | definition의 이해를 돕는 이미지 파일 혹은 경로. |
| *hit* |  | 조회수 |
| *like* |  | 호감수 |
| *dislike* |  | 비호감수 |
| *pop* |  | 인기도 (고유 알고리즘 이용 계산) |
| *member\_id (FK)* |  | author. 작성자. 외래 키로 member의 id를 가짐. |
|  |  |  |

#### vote

vote는 투표 정보를 가지고 있다.

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| *id (PK)* |  |  |
| *like* |  | 좋아요 |
| *dislike* |  | 싫어요 |
| *term\_id (FK)* |  | term id |
| *member\_id (FK)* |  | 투표를 한 사용자 id |
|  |  |  |

#### comment

comment는 term에 달리는 댓글 정보를 가지고 있다.

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| *id (PK)* |  |  |
| *name* |  | 이름 |
| *content* |  | 내용 |
| *date* |  | 일시 |
| *like* |  | 좋아요 수 |
| *dislike* |  | 싫어요 수 |
| *pop* |  | 인기도 |
| *term\_id (FK)* |  | 댓글을 단 term id |
| *comment\_id (FK)* |  | 댓글에 댓글을 달았을 경우 그 댓글의 id |
| *member\_id (FK)* |  | 댓글 단 사람의 id |
|  |  |  |

#### message

Message은 사용자간 메시지 정보를 가지고 있는 테이블이다.

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| *추후 서술* |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 3. System Features

## 3.1 Object and Flow Diagram

*다음 장에 전체 Object Diagram과 Flow Chart를 첨부한다.*

## 3.2 System Components Overview

본 프로그램은 다양한 구성 요소로 이루어진다. 주요 성분으로 Gateway, Configuration, Core, Controller, Model과 View등이 있다. 이하 이에 대한 내용을 기술한다.

### 3.2.1 Gateway and Configuration

#### Gateway (root/index.php)

localhost 가 호출되면 가장 먼저 root 폴더의 index.php 를 실행한다. 즉, root/index.php 가 UB의 Gateway 기능을 한다. 또한, 새로운 URL 로의 이동이 있을 시 (예: Anchor을 통한), 다시 가장먼저 Gateway를 실행한 후 요청에 맞게 프로그램을 실행한다. 즉, Script를 이용한 페이지 요소 갱신 제외, 모든 링크 이동은 반드시 Gateway를 요청한다.

|  |  |
| --- | --- |
| ***Procedure*** |  |
| *Path Setting* | 시스템 전반에 필요한 Path를 설정한다. |
| *Loading Files* | debug.php, define.php, core.php 로드한다. |
| *Initiate Core* |  |
|  |  |

#### Config.xml

Config.xml은 UB 실행에 관한 여러 Configuration 정보를 갖고 있는 XML형식의 파일이다.

Config.xml이 가지고 있는 XML Element에 관한 명세는 다음과 같다.

|  |  |
| --- | --- |
| ***Components*** |  |
| */Info* | 시스템의 전반적 정보를 담는다.  하위 속성: name, version, root. |
| */Database* | Database 객체가 필요한 정보들로 데이타베이스와 관련한 정보를 가지고 있다.  하위 속성: name, host, port, user, password |
| */Router* | Router 객체가 필요한 정보들.  하위 속성: url, controller, method |
|  |  |
|  |  |

### 3.2.2 Core

#### Core.php

Core는 UB의 최상위 Managing Object 이다. Core는 Config객체와 Router객체를 갖고, Router객체의 route()를 실행하는 기능을 한다.

|  |  |
| --- | --- |
| ***Field*** |  |
| *$instance []* |  |
| *$config* |  |
|  |  |
| ***Method*** |  |
| *\_\_construct($file\_config)* |  |
| *spl\_autoload\_register(function($class){})* |  |
| *getInstance()* |  |
| *getControllerParam()* |  |
| *main()* |  |
|  |  |

#### Router.php

URL 정보를 파싱하는 중추적인 역할을 하는 객체이다. 해당 기능을 하는 route()를 포함한다.

|  |  |
| --- | --- |
| ***Field*** |  |
| *$\_routes* |  |
|  |  |
|  |  |
| ***Method*** |  |
| *\_\_construct($config\_Router)* |  |
| *toRoutes($url, $ctrl, $method)* |  |
| *match($\_routes, $url, $method)* |  |
| *route()* |  |
|  |  |
|  |  |

#### Controller.php

MVC의 C에 해당하는 요소로, Model과 View를 포함하고 전체적인 로직을 관장하는 객체이다. 모든 Controller는 main 함수를 기본함수로 가진다. 사용자 정의 Controller 객체들은 모두 이 Controller 객체를 상속한다.

|  |  |
| --- | --- |
| ***Field*** |  |
| *$view* |  |
| *$model []* |  |
| *$data []* |  |
| *$\_CONSTANT []* |  |
|  |  |
| ***Method*** |  |
| *\_\_construct()* |  |
| *main()* |  |
| *\_\_set($name, $value)* |  |
| *\_\_get($name)* |  |
| *redirect($url, $permanent)* |  |
| *render($file\_template)* |  |

#### Model.php

MVC의 M에 해당하는 요소로, 데이터 입출력을 관장한다. 모든 사용자 정의 Model 객체들은 이 객체를 상속한다.

한 개의 모델은 한 개의 DB테이블과 교신하는 것을 원칙으로 한다.

|  |  |
| --- | --- |
| ***Field*** |  |
| *$db* |  |
|  |  |
| ***Method*** |  |
| *getDatabase()* |  |

#### Database.php

Model이 소유하고 있는 Database객체로, DB와 접속을 관장하는 객체이다.

|  |  |
| --- | --- |
| ***Field*** |  |
| *$name* |  |
| *$host* |  |
| *$port* |  |
| *$user* |  |
| *$password* |  |
|  |  |
| ***Method*** |  |
| *\_\_construct($config\_Database)* |  |
| *getConnection()* |  |
| *\_\_clone()* |  |

#### Data.php (ABORT)

Model이 소유하고 있는 Data객체로, DB에서 가져온 데이터를 가지고 있다.

Data.php는 다음의 요소를 포함한다.

함수

set():

get():

### 3.2.3 Controller

#### Root.php

Domain이 호출되면 기본값으로 출력되는 화면을 구성하는 컨트롤러 객체다.

|  |  |
| --- | --- |
| ***URL*** | localhost/ |
|  |  |
| ***Field*** |  |
| *$model []* |  |
| *$view* |  |
| ***Method*** |  |
|  |  |
|  |  |

#### Term.php

URL: localhost/term/

검색 요청을 관장하는 컨트롤러로, 사용자가 요청하는 Term을 추출하여 보여주는 기능을 한다.

|  |  |
| --- | --- |
| ***URL*** | localhost/term/\* |
|  |  |
| ***Field*** |  |
| *$model []* |  |
| *$view* |  |
| ***Method*** |  |
|  |  |
|  |  |

#### About.php

URL: localhost/about/

UB에 관한 개략적 정보를 출력하는 것을 관장하는 컨트롤러이다.

|  |  |
| --- | --- |
| ***URL*** | localhost/about/\* |
|  |  |
| ***Field*** |  |
| *$model []* |  |
| *$view* |  |
| ***Method*** |  |
|  |  |
|  |  |

#### Rules.php

URL: localhost/rules/

UB의 이용 규칙 전반을 출력하는 것을 관장하는 컨트롤러이다.

|  |  |
| --- | --- |
| ***URL*** | localhost/rules/\* |
|  |  |
| ***Field*** |  |
| *$model []* |  |
| *$view* |  |
| ***Method*** |  |
|  |  |
|  |  |

#### Guide.php

URL: localhost/guide/

UB를 어떻게 이용해야 하는지 소개하는 것을 관장하는 컨트롤러이다.

|  |  |
| --- | --- |
| ***URL*** | localhost/guide/\* |
|  |  |
| ***Field*** |  |
| *$model []* |  |
| *$view* |  |
| ***Method*** |  |
|  |  |
|  |  |

#### Notice.php

URL: localhost/notice/

최근 공지사항을 출력하는 것을 관장하는 컨트롤러이다.

|  |  |
| --- | --- |
| ***URL*** | localhost/notice/\* |
|  |  |
| ***Field*** |  |
| *$model []* |  |
| *$view* |  |
| ***Method*** |  |
|  |  |
|  |  |

#### Archive.php

URL: localhost/archive/

계층 별, 또는 다른 군 별 단어 검색 분포와, 최근 신조어 동향 등 큰 단위의 Data를 수집하고, 분석하고, 출력하는 것을 관장하는 컨트롤러이다.

|  |  |
| --- | --- |
| ***URL*** | localhost/archive/\* |
|  |  |
| ***Field*** |  |
| *$model []* |  |
| *$view* |  |
| ***Method*** |  |
|  |  |
|  |  |

#### Login.php

URL: localhost/login/

사용자가 UB에 로그인요청 시, 이를 관장하는 컨트롤러이다.

|  |  |
| --- | --- |
| ***URL*** | localhost/login/\* |
|  |  |
| ***Field*** |  |
| *$model []* |  |
| *$view* |  |
| ***Method*** |  |
|  |  |
|  |  |

#### Register.php

URL: localhost/register/

사용자가 UB에 회원가입을 요청 시, 이를 관장하는 컨트롤러이다.

|  |  |
| --- | --- |
| ***URL*** | localhost/login/\* |
|  |  |
| ***Field*** |  |
| *$model []* |  |
| *$view* |  |
| ***Method*** |  |
|  |  |
|  |  |

#### Mypage.php

URL: localhost/mypage/

사용자가 UB에 로그인 혹은 회원가입을 요청 시, 이를 관장하는 컨트롤러이다.

|  |  |
| --- | --- |
| ***URL*** | localhost/login/\* |
|  |  |
| ***Field*** |  |
| *$model []* |  |
| *$view* |  |
| ***Method*** |  |
|  |  |
|  |  |

### 3.2.4 Model

#### Term\_md.php

단어 검색 요청 시, UB의 DB로 부터 해당 요청을 처리하는 모델이다.

|  |  |
| --- | --- |
| ***Field*** |  |
| *$data []* |  |
|  |  |
| ***Method*** |  |
| *getTerm($term)* | 해당 string과 연관한 term을 가져온다. |
| *getTermExact($id)* | 해당 id의 한 개의 term을 가져온다. |
| *getRecentTerm($num)* | 1일 전까지 등록되고 현재까지 삭제되지 않은 term을 $num 개만큼 가져온다. |
| *getRecentHotTerm($num)* | 1일전까지 등록되고 현재까지 삭제되지 않은 term중 pop이 어느 이상인 것들을 $num 개 만큼 가져온다. |
| *getRelatedTerm($term)* | 해당 term과 연관한 term을 가져온다. |
| *addTerm()* | term을 추가한다. |
| *getOtherDef($term)* | 정확히 같은 term의 다른 entry를 가져온다. |
| *getMyTerm()* | 내가 올린 term을 가져온다. |
| *delTerm($id)* | 해당 id의 term을 삭제한다. |
| *editTerm($id)* | 해당 id의 term을 수정한다. |

#### Comment\_md.php

단어 검색 요청 시, UB의 DB로 부터 해당 요청을 처리하는 모델이다.

|  |  |
| --- | --- |
| ***Field*** |  |
| *$data []* |  |
|  |  |
| ***Method*** |  |
| *getComment($term\_id)* | 해당 term에 달린 댓글을 가져온다. |
| *getMyComment($member\_id)* | 내가 올린 댓글을 가져온다. |
| *getCommented($member\_id)* | 내가 올린 글 혹은 댓글에 누군가가 쓴 댓글들을 가져온다. |
| *addComment()* | 댓글을 입력한다. |
| *delComment()* | 댓글을 삭제한다. |
| *editComment()* | 댓글을 수정한다. |

#### Member\_md.php

회원정보를 관장하는 모델이다.

|  |  |
| --- | --- |
| ***Field*** |  |
| *$data []* |  |
|  |  |
| ***Method*** |  |
| *addMember()* | 새로운 회원이 폼을 작성하여 가입을 희망할 때 db에 새로운 회원정보를 입력한다 |
| *editInfo()* | 회원이 스스로 본인의 자기소개를 수정한다. |
| *editName($id, $name)* | 회원이 스스로 웹 사이트상 본인의 이름을 수정한다. |
| *editPassword($id, $password)* | 회원이 스스로 비밀번호를 수정한다. |
| *isMember($id, $password)* | 요청받은 아이디(이메일)와 패스워드에 해당하는 회원이 있는지 확인한다. |
| *signedUp($id)* | 해당 이메일로 가입한 바가 있는지 확인. 추후에는 메일 서버를 두어, 즉시 응답을 주지 않고, 해당 이메일로 가입이 되어 있는 경우, 그 이메일로 메일을 보내겠다고 응답. |
| *delMember()* | 회원 탈퇴 희망 시, db에서 삭제하지 않고, 외부에 보이지 않도록 변경한다. |
| *resetPassword($email)* | 메일서버에서 해당 $email로 reset된 password를 전송한다. |

#### Vote\_md.php

좋아요/싫어요 기능을 관장하는 모델이다.

|  |  |
| --- | --- |
| ***Field*** |  |
| *$data []* |  |
|  |  |
| ***Method*** |  |
| *likeTerm($term\_id)* | 해당 term의 like을 1올린다. |
| *dislikeTerm($term\_id)* | 해당 term의 dislike을 1올린다. |
| *likeComment($comment\_id)* | 해당 comment의 like을 1올린다. |
| *dislikeComment($comment\_id)* | 해당 comment의 dislike을 1올린다. |
| *calcPopTerm(term\_id)* | term의 hit, like, dislike를 이용하여 pop을 구하고 갱신한다. |
| *calcPopComment($comment\_id)* | comment의 hit, like, dislike를 이용하여 pop을 구하고 갱신한다. |

#### SNSBrowser\_md.php

특정 커뮤니티, 포털 사이트들로부터 Term에 관한 용례를 가져오는 기능을 관장하는 모델이다.

추후 설명

#### Notice\_md.php

공지사항에 있는 자료를 처리하는 모델이다.

|  |  |
| --- | --- |
| ***Field*** |  |
| *$data []* |  |
|  |  |
| ***Method*** |  |
| *getNotice()* |  |
| *getNoticeExact()* |  |
| *addNotice ()* |  |
| *getOtherDef()* |  |
| *addTerm()* |  |
|  |  |

#### Archive\_md.php

추후 서술

|  |  |
| --- | --- |
| ***Field*** |  |
| *$data []* |  |
|  |  |
| ***Method*** |  |
|  |  |
|  |  |

### 3.2.5 View

추후 서술

## 3.3 Functional Requirements

### 3.3.1 Account

사용자 계정과 관련한 기능의 명세.

#### Account Registration

사용자가 사이트에 회원가입하는 기능.

회원가입 시, 사용자로부터 {email, password, name, intro (optional)} 를 POST로 입력받아 db에 등록한다.

photo필드는 일단 사용하지 않는다.

Register 컨트롤러에서 실행한다.

Member\_md 모델의 addMember() 메소드를 이용하고, post방식으로 data를 전송받아 member 테이블에 입력.

“Verification of Signed-Up Status (of an Email)” 기능과 연동한다.

#### Account Login

사용자가 사이트에 로그인하는 기능.

입력받은email과 password가 db에 존재하는 메일/비밀번호 쌍이면 참 값을 반환하여 로그인을 가능케 한다.

만약 일치 계정정보가없을 경우 (동일한 ip로 한 email당) 3번까지 재도전 기회를 준다. 추후 3번 이후, 10분 쉬라고 prompt를 띄우는 기능을 구현한다.

로그인 성공 시, db에 last login 일시를 갱신한다.

Login 컨트롤러에서 실행한다.

Member\_md모델의 isMember($email, $password) 를 이용한다.

#### Verification of Signed-Up Status (of an Email)

사용자의 회원가입 시, 이미 가입된 이메일이 있는지 확인하는 기능이 필요하다.

이메일을 입력받고, db에 존재하면 있다고 출력한다.

보류(이 때, client의 정보보호를 위해 즉시 해당 이메일의 가입여부를 알려주지 않고, 가입되어 있으면 해당메일로 계정정보를 전송하고, “전송했다”고 이를 화면에 출력한다.)

Register 컨트롤러에서 실행한다.

Member\_md 모델의 signedUp($email) 이용한다.

#### Edit Account Information

계정의 이메일 주소는 우선은 변경할 수 없도록 규정한다. 다만, 보여지는 name과 intro는 변경할 수 있게 한다. 이를 위해 폼을 만들어 POST방식으로 데이터를 받는다.

Member\_md의 editInfo()를 이용한다.

#### Password Reset

계정의 비밀번호를 잃어버린 경우, 해당 메일 주소로 임시 생성한 비밀번호를 보내준다.

Member\_md의 resetPassword($email)를 이용한다.

#### Delete Account

회원탈퇴 희망 시, 탈퇴한다.

탈퇴 시, db 삭제가 아니라, 사이트에서 더 이상 보이지 않게 한다.

Member\_md의 delAccount()를 이용한다.

### 3.3.x Term

Term 과 관련한 기능의 명세.

#### Search Term by Query String

String 을 입력 받아 db로 부터 term을 검색한다. 더 연관성이 높은 entry가 위로 오게 출력한다. 따라서, 출력 전 어떤 sorting 이 요구된다.

추후에, 사용자 정보를 제한적으로 수집하여, 어떤 연령대 혹은 어떤 성별에서 어떤 Query를 날리는지 데이터 통계화 하는 기능 구현을 계획한다.

추후에, 검색어 자동완성 기능을 계획한다. (showRelatedTerms() 기능 연동)

Term\_md의 getTerm($term)을 이용한다.

#### View Definition of a Particular Entry

하나의 entry를 열람하는 기능이다.

Term\_md의 getTermExact($id)를 이용한다. 해당 단어를 보여주고, hit (조회 수) 을 1추가 하는 기능을 한다.

#### Add Term

단어를 추가하는 기능이다.

사용자로부터 {term, pos(optional), lemma(optional), definition, usage, image(optional)}를 POST로 입력 받아 db에 등록한다.

실제 db입력은 member 테이블의 id도 외래키로 참조하여 작성자를 함께 넣는다.

Show Related Terms와 연동하여, 사용자가 term을 입력할 때, ajax로 “혹시 이 term이 아닙니까?” 로 리스트를 출력하여, term입력을 돕는다.

pos와 lemma는 optional로 받고, 도움말을 제공한다. 추후 자동 추천기능도 구현한다.

Term 컨트롤러에서 실행.

Term\_md의 addTerm()을 이용.

#### Show Related Terms

연관 단어를 보여주는 기능으로, def가 아니라 연관 단어들의 term만 가져온다.

어떤 Term String을 인자로 갖고, 해당 String과 비슷하거나 연관이 있는 다른 Term들을 반환한다.

Term 컨트롤러에서 실행.

Term\_md의 getRelatedTerms($term)을 을 이용.

#### Show Recently Added Terms

최근 등록된 단어를 보여준다.

등록된 지 24시간이 지났고, 어느정도 pop을 획득한 term을 출력한다.

Term 컨트롤러에서 실행.

Term\_md의 getRecentHotTerm() 이용.

#### Edit Term

작성자 본인은 스스로 게재한 Term과 Definition을 수정할 수 있다.

수정 form은 입력 form과 틀은 동일하나 각 input에 기 입력한 데이타가 먼저 뿌려질 수 있도록 한다.

수정 후 글에 언제 수정을 하였다고 표시한다.

화면 출력은 ajax로 처리하여, 페이지 갱신없이 구현한다.

Term 컨트롤러에서 실행.

Term\_md의 editTerm($id) 이용.

#### Delete Term

작성자 본인에 한해, term을 삭제할 수 있게 한다.

화면 출력은 ajax로 처리하여, 페이지 갱신없이 구현한다.

Term 컨트롤러에서 실행.

Term\_md의 delTerm($id)를 이용.

#### Like Term

사용자들로부터 Upvote를 입력받는 기능이다.

like을 1올리고, pop을갱신한다.

화면 출력은 ajax로 처리하여, 페이지 갱신없이 구현한다.

Term 컨트롤러에서 실행.

Vote\_md의 likeTerm($term\_id)에서 like를 1올리고, calcPopTerm($term\_id)을 이용하여 pop을 갱신한다.

#### Dislike Term

사용자들로부터 Upvote를 입력받는 기능이다.

dislike을 1올리고, pop을갱신한다.

화면 출력은 ajax로 처리하여, 페이지 갱신없이 구현한다.

Term 컨트롤러에서 실행.

Vote\_md의 likeTerm($term\_id)에서 dislike를 1올리고, calcPopTerm($term\_id)을 이용하여 pop을 갱신한다.

#### Show Term I registered

본인이 등록한 term entry를 반환한다.

반환시, term만 보일지, def 일부도 보일지는추후 고려한다.

Mypage 컨트롤러에서 실행한다.

Term\_md의 getMyTerm($member\_id)를 이용한다.

#### Expiration of Author Information of a Term Entry

추후구현.

1년이 지나면 작성자 정보를 소멸시켜주는 기능이다.

사용자에게 이메일을 보내어 의사를 물어보고 6개월 간 답을 기다린다. 만약 답이 없을 시 잠정적으로 author info을 보이지 않게 하고, 이후라도 그러지 않기 원하면 author info를 남겨둔다.

### 3.3.x Comment

댓글과 관련한 기능을 명세한다.

#### Show Comment for an Entry

해당 entry에 달린 댓글을 가져온다.

term의 id를 외래키로 참조한다.

Comment컨트롤러에서 실행.

Comment\_md의 getComment($term\_id) 이용.

#### Add Comment

Comment를추가한다.

폼에서 {author, content} 를 POST로 입력 받아 db에 추가한다.

실제 db입력은 일시와 comment\_id를 외래키 (같은 table내의)로 참조하여 포함한다.

Comment 컨트롤러에서 실행.

Comment\_md의 addComment() 이용.

#### Edit Comment

작성자 본인에 한해Comment를 수정하는 기능.

수정 폼은 입력 폼과 같으나, input안에 기 게재한 내용이 표시될 수 있도록 한다.

수정 되면 수정되었음을 표시하고, 수정일시를 표시한다. 따라서 원 게제일과 수정일을 모두 표시 한다.

화면 출력은 ajax로 처리하여, 페이지 갱신없이 구현한다.

Comment 컨트롤러에서 실행.

Comment\_md의 editComment() 이용.

#### Delete Comment

작성자 본인에 한해 Comment를 삭제하는 기능.

삭제 시 prompt를 띄우고, 삭제하면 댓글 리스트에서 회색 음영처리를 하고 “해당 댓글은 삭제되었습니다”로 표현한다.

화면 출력은 ajax로 처리하여, 페이지 갱신없이 구현한다.

Comment 컨트롤러에서 실행.

Comment\_md의 delComment($comment\_id) 이용.

#### Show Comment I posted

본인이 작성한 comment 목록을 가져온다.

Mypage 컨트롤러에서 실행.

Comment\_md 의 getMyComment() 이용.

#### Like Comment

사용자들로부터 Upvote를 입력받는 기능이다.

like을 1올리고, pop을갱신한다.

화면 출력은 ajax로 처리하여, 페이지 갱신없이 구현한다.

Term 컨트롤러에서 실행.

Vote\_md의 likeComment($comment\_id)에서 like를 1올리고, calcPopComment($comment\_id)을 이용하여 pop을 갱신한다.

#### Dislike Comment

사용자들로부터 Upvote를 입력받는 기능이다.

dislike을 1올리고, pop을갱신한다.

화면 출력은 ajax로 처리하여, 페이지 갱신없이 구현한다.

Vote\_md 컨트롤러에서 실행.

Vote\_md의 likeComment($comment\_id)에서 dislike를 1올리고, calcPopComment($comment\_id)을 이용하여 pop을 갱신한다.

### 3.3.x Archive

Archive의기능과 관련한 명세

#### Show Trend

추후 서술

## 3.4 Other Requirements