Dealistic

Design Specification



|  |  |
| --- | --- |
| Student Number | Name |
| 2014312794 | 김준현 (Junhyun Kim) |
|  |  |
|  |  |
|  |  |
|  |  |

Contents

[Preface 5](#_Toc9191657)

[Objectives 5](#_Toc9191658)

[Readership 5](#_Toc9191659)

[Document Structure 5](#_Toc9191660)

[Introduction 5](#_Toc9191661)

[Objectives 5](#_Toc9191662)

[Applied Diagram 5](#_Toc9191663)

[UML 5](#_Toc9191664)

[Package Diagram 5](#_Toc9191665)

[Deployment Diagram 5](#_Toc9191666)

[Class Diagram 5](#_Toc9191667)

[State Diagram 5](#_Toc9191668)

[Sequence Diagram 5](#_Toc9191669)

[ER Diagram 5](#_Toc9191670)

[Applied Tool 5](#_Toc9191671)

[Draw.io 5](#_Toc9191672)

[PowerPoint 5](#_Toc9191673)

[Project Scope 5](#_Toc9191674)

[System Architecture – Overall 6](#_Toc9191675)

[Objectives 6](#_Toc9191676)

[System Organization 6](#_Toc9191677)

[Package Diagram 6](#_Toc9191678)

[Deployment Diagram 6](#_Toc9191679)

[SA – Frontend 6](#_Toc9191680)

[Objectives 6](#_Toc9191681)

[Subcomponents 6](#_Toc9191682)

[SA – Backend 7](#_Toc9191683)

[Objectives 7](#_Toc9191684)

[Subcomponents 7](#_Toc9191685)

[Class Diagram 7](#_Toc9191686)

[Sequence Diagram 7](#_Toc9191687)

[State Diagram 7](#_Toc9191688)

[Protocol Design 8](#_Toc9191689)

[REST Protocol 8](#_Toc9191690)

[JSON 8](#_Toc9191691)

[Details 8](#_Toc9191692)

[Database Design 8](#_Toc9191693)

[Testing Plan 9](#_Toc9191694)

[Objective 9](#_Toc9191695)

[Testing Policy 9](#_Toc9191696)

[Development Testing 9](#_Toc9191697)

[Release Testing 9](#_Toc9191698)

[User Testing 9](#_Toc9191699)

[Testing Case 9](#_Toc9191700)

[Development Environment 9](#_Toc9191701)

[Frontend Environment 9](#_Toc9191702)

[Backend Environment 9](#_Toc9191703)

[Develop Plan 9](#_Toc9191704)

[Index 9](#_Toc9191705)

# Preface

## Objectives

## Readership

## Document Structure

# Introduction

## Objectives

## Applied Diagram

### **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.

### **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.

### **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.

### **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.

### **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.

### **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.

.

### **ER Diagram**

## Applied Tool

### Draw.io

### PowerPoint

## Project Scope

# System Architecture – Overall

## Objectives

## System Organization

## Package Diagram

## Deployment Diagram

# SA – Frontend

## Objectives

## Subcomponents

* Ranking
* Item Detail
* Recommendation
* Mypage
* Search

Each subcomponent should specify these diagrams:

* Class Diagram
* Sequence Diagram
* State Diagram

# SA – Backend

## Objectives

## Subcomponents

* Review Analyzer
* Review Crawler
* Database Handler(Updater)

## Class Diagram

**(WIP – if you have additional classes, notice me)**

* User [id(PK), name, authority, pw\_hash, bookmarks(FKs), recommend\_categories(FKs)]
* Authority [id(PK), name]
* Item [id(PK), name, company, links, specs, reviews(FKs), review\_scores(FKs), keywords(FKs)]
* RecommendCategory [id(PK), name, description, keywords(FKs)]
* Bookmark [user\_id(FK), item\_id(FK), add\_date]
* Review [id(PK), title, author, reference(FK), content, score, importance, keywords(FKs)]
* Reference [id(PK), name, link]

## Sequence Diagram

## State Diagram

# Protocol Design

## REST Protocol

common protocol design for communicate between client-server

## JSON

JS standard, widely used

## Details

* User
* Item
* Keyword
* RecommendCategory
* Bookmark
* Review

# Database Design

ER Diagram: Same as described at class diagram.

# Testing Plan

## Objective

## Testing Policy

### Development Testing

### Release Testing

### User Testing

### Testing Case

# Development Environment

## Frontend Environment

## Backend Environment

# Develop Plan

# Index