* **소프트웨어공학 최종 보고서 -**

텍스트, 표지판이(가) 표시된 사진

자동 생성된 설명

|  |  |  |  |
| --- | --- | --- | --- |
| **과목명** | 소프트웨어공학 | | |
| **학과** | 컴퓨터공학과 | | |
| **담당교수** | 장성진 교수님 | | |
| **분반** | 002분반 | | |
| **팀 번호** | 4조 | | |
| **학번 및 성명** | 팀장 | 20163241 | 조영문 |
| 팀원 | 20163265 | 유하영 |
| 20153182 | 유한규 |
| 20193019 | 이상미 |

[**최종 보고서 페이지**](#최종보고서) **<Ctrl과 함께 누르면 이동됩니다>**

**프로젝트 계획서 목차**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1.** | **프로젝트 개요** | | |  |
| 1.1 | | 프로젝트 목적 | |  |
| 1.2 | | 프로젝트 정의 및 시장분석 | |  |
| 1.2.1 | | | 서점 어플 |  |
| 1.2.2 | | | 시장 분석 |  |
| 1.3 | | 프로젝트 산출물 | |  |
|  | |  | |  |
| **2.** | **프로젝트 주요 일정** | | |  |
|  | |  | |  |
| **3.** | **프로젝트 조직** | | |  |
| 3.1 | | 프로젝트 팀 조직 | |  |
| 3.1.1 | | | 범위 및 역할 |  |
| 3.2 | | 개발 방법론 및 도구 선정 | |  |
|  | |  | |  |
| **4.** | **기술적 프로세스(프로젝트 개발 계획)** | | |  |
| 4.1 | | 개발 생명주기 모델 | | |
| 4.1.1 | | | 프로젝트 생명 주기 |  |
| 4.1.2 | | | 생명주기 |  |
| 4.2 | | 개발 방법론 및 도구 선정 | | |
| 4.2.1 | | | 개발 방법론 | |
| 4.2.2 | | | 객체지향 방법론 절차 | |
|  | | |  |  |
| **5.** | **개발 규모 산정** | | |  |
| 5.1 | | WBS | |  |
| 5.2 | | 상세 일정 | |  |
| 5.3 | | 프로젝트 규모 산정 – 간이 기능 점수법 | |  |
|  | |  | |  |
| **6.** | **마일 스톤 및 일정 계획** | | | |

1. **프로젝트 개요**

* 프로젝트명 : 메뉴 추천 프로그램
* 프로젝트 기간 : 21.09.17 ~ 21.12.10
* 개발 대상 : 메뉴 추천 어플리케이션
* 개발 목적 : JAVA를 이용하여 음식 메뉴를 추천하는 기능의 프로그램을 개발하여 프로젝트에 대한 경험을 쌓고 사용자들이 만족할 수 있도록 한다.
* 구현 환경 : Windows 10, Oracle, JDK 1.8
* 개발 언어 : JAVA
* 개발 도구 : Eclipse
* 팀 구성 : 조영문, 유하영, 유한규, 이상미
  1. 프로젝트 목적

본 프로젝트의 목적은 프로그램 사용자가 보다 편리하게 음식 메뉴를 선정할 수 있게 하는 것에 목적을 두고 있다.   
 사용자는 음식 추천 기능을 통해 쉽고 빠르게 음식 메뉴를 선정할 수 있고 선택한 메뉴를 기준으로 주변 음식점 정보를 확인할 수 있도록 한다. 음식점은 매장 정보를 입력하여 사용자의 편의를 돕고 홍보를 통한 매출 증가에 목적을 둔다.

* 1. 프로젝트 정의 및 시장분석
     1. 메뉴 추천 App

사용자가 어플을 통해 음식 메뉴를 추천 받을 수 있는 App이다. 랜덤으로 메뉴를 추천해주는 기능, 추천된 메뉴로 근처 식당을 추천해 주는 기능 등 앱마다 다양한 기능을 가지고 있다.

* + 1. 시장 분석

21.09.20 기준, 구글 스토어에서 메뉴 추천 어플 조사 결과, <점심 뭐 먹지?>, <뭐 먹을까?>, <그거 말고>, <오늘 뭐 먹지?> 등 여러 어플이 시중에 나와있다.  
  
- 점심 뭐 먹지? : 랜덤 메뉴 추천, 추천된 메뉴로 근처 식당 추천. 어플 오작동 많음. 별점 2.4/5  
- 뭐 먹을까? : 메뉴 랜덤 고르기, 여러 메뉴 중 다중 선택 후 랜덤 추천, 메뉴 월드컵 기능, 질문을 통한 추천. 별점 4.8/5  
- 그거 말고 : 단순 랜덤 추천. 별점 4.6/5  
- 오늘 뭐 먹지? : 내 주변 식당 랜덤 뽑기 기능, 랜덤 메뉴 추천 기능, 선택된 메뉴의 식당 정보 보기, 리스트 추가를 통해 나만의 랜덤 리스트 제작 기능. 별점 4.9/5  
  
 어플 기능에 따라 이용자들의 만족도는 천차만별이다. 만족도가 낮은 어플들은 대부분 메뉴를 단순히 랜덤으로 추천해 주는 기능만을 가지고 있고 어플의 오작동이 많이 있었다. 반면, 만족도가 높은 어플은 다양한 기능을 통해 이용자들의 만족도를 높였다. 우리는 단순 메뉴 랜덤 추천만이 아니라 차별성 있는 기능들을 두어 이용자들의 만족도를 높일 수 있도록 할 것이다.

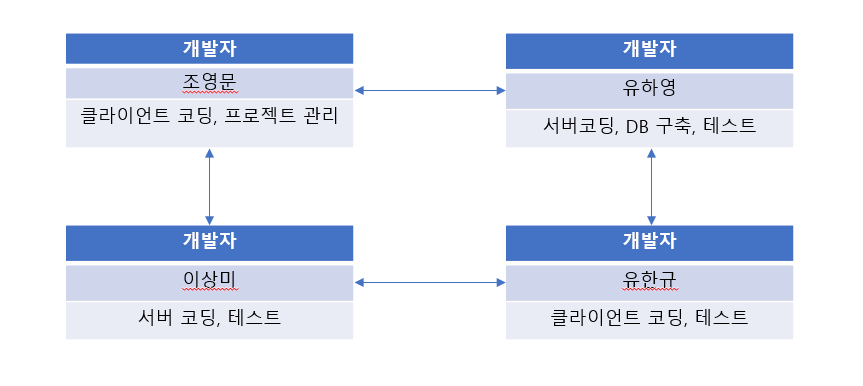
* 1. 프로젝트 산출물  
     - 프로젝트 제안서  
     - 요구사항 명세서  
     - 프로젝트 계획서  
     - 설계 사양서  
     - 소스코드  
     - 테스트 결과 보고서  
     - 프로젝트 완료 보고서

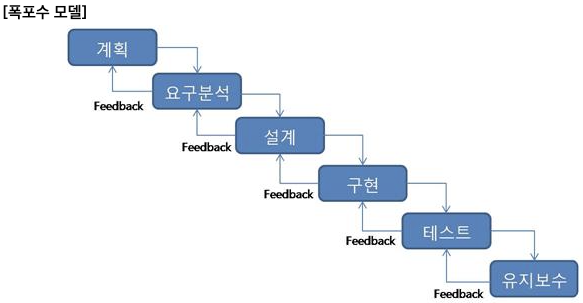
1. **프로젝트 주요 일정**

|  |  |  |
| --- | --- | --- |
| **주요 개발 활동** | **수행 일정** | **주요 산출물** |
| **프로젝트 제안** | 21.09.17 | 프로젝트 제안서 |
| **요구사항 분석** | 21.09.17 ~ 21.09.24 | 요구사항 명세서 |
| **프로젝트 계획** | 21.09.24 ~ 21.09.30 | 프로젝트 계획서 |
| **설계** | 21.10.01 ~ 21.10.15 | 설계 사양서 |
| **구현/테스트** | 21.10.15 ~ 21.11.31 | 소스/단위 테스트, 통합 테스트 결과 보고서, 소스코드 |
| **테스트** | 21.11.31 ~ 21.12.08 | 시스템 테스트 결과 보고서 |
| **프로젝트 종료** | 21.12.08 ~ 21.12.10 | 프로젝트 완료 보고서 |

1. **프로젝트 조직**
   1. 프로젝트 팀 조직

민주적 팀 구성

* + 1. 범위 및 역할  
       

1. **기술적 프로세스(프로젝트 개발 계획)**
   1. 개발 생명주기 모델
      1. 프로젝트 생명주기  
         피드백이 있는 폭포수 모델 (Waterfall Model)  
          
      2. 생명주기

|  |  |  |
| --- | --- | --- |
| **개발 단계** | **관련 요소** | **내용** |
| 분석 | 입력 산출물 | • 프로젝트 제안서 |
| 수행 태스크 | • 소프트웨어에 요구되는 기능, 성능, 인터페이스 등 사용자의  요구사항을 구체적으로 이해하고 문서화 한다.  • 프로젝트 제안서를 바탕으로 요구사항 명세서를 작성  - 요구사항 분석  - Use Case 작성  - UI프로토타입 작성  • 무엇을 어떻게 구현되어야 하는지 명확하게 작성한다.  • 생명주기 및 개발 방법론 결정  • 고객의 인터뷰 실시  • ERD작성 |
| 출력 산출물 | • 요구사항 명세서 |
| 시작 기준 | • 고객의 요구사항이 수집되고 기록이 되어 있어야 한다.  • 프로젝트 관계자는 고객의 요구사항을 파악하고 이해한다.  • 프로젝트 정의서가 승인되어야 하고 제안서가 문서화 되어야 한다. |
| 완료 기준 | • 유스케이스 별로 기능이 구체적으로 기술되어야 한다.  • 요구사항 명세서가 작성된다. |
| 계획 | 입력 산출물 | • 요구사항 명세서 |
| 수행 태스크 | • 고객의 요구사항 분석서 작성  • 추진 방안을 제시하고 시스템 개발 비용 및 소요기간, 인력등의 개발 계획을 설립한다.  • 데이터 및 컴퓨터 자원 관리 계획 수립  • 위험 관리 계획 수립  • 상세 수준의 WBS작성  • 프로젝트 계획서 작성 및 검토  • 테스트 계획서 작성 및 검토 |
| 출력 산출물 | • 프로젝트 계획서 |
| 시작 기준 | • 고객으로부터 요구사항이 전달된다.  • 요구사항 정의서가 문서화되고 승인되어야 한다.  • 요구사항 수집 및 정의서 작성 활동을 지원하기 위한 도구가 제공되고 이용 가능해야 한다.  • 요구사항 명세서를 기초로 무엇을 만들어야 하는지 안다. |
| 완료 기준 | • 고객의 요구사항이 변동되지 않고 수집 및 기록되어야 한다.  • 프로젝트 계획서가 작성되고 고객 및 이해 관계자에게 전달된다. |
| 설계 | 입력 산출물 | • 요구사항 명세서  • 프로젝트 계획서 |
| 수행 태스크 | • 아키텍처 정의서 작성 및 DB 설계  • 모든 시스템의 구조 결정(아래 내용을 포함)  - 프로그램의 데이터 구조  - 소프트웨어 구조  - 인터페이스 구조  - 알고리즘  • 클래스 다이어그램 작성  • 설계 명세서 작성 및 검토 |
| 출력 산출물 | • 설계 명세서 |
| 시작 기준 | • 요구사항 명세서가 구체적으로 기술되어 있고 프로젝트 관리자들이 요구사항 명세서를 이해하고 있다.  • 프로젝트 팀의 개발자 및 팀 구성원이 모두 결정되어야 한다.  • 프로젝트의 기간과 일정이 구체적으로 명확하게 결정되어야 한다.  • 프로젝트 개발 도구, 개발 언어, 관리 도구 등이 결정 되어야 한다. |
| 완료 기준 | • 설계 명세서가 작성된다.  • 프로젝트의 위험 요소가 정확하고 구체적으로 파악되고 이에 따른 계획이 수립되어야 한다. |
| 구현 | 입력 산출물 | • 설계 명세서 |
| 수행 태스크 | • 설계 명세서를 바탕으로 시스템의 프로그램 개발  • 개발 코딩 단위 테스트  • 설계 단계에서 논리적으로 결정한 문제 해결 방법을 특정  프로그래밍 언어를 사용하여 실제 프로그램을 작성한다.  • 프로그래밍 언어 선택, 기법, 스타일, 순서 등을 결정.  • 설계 단계에서 정해진 프로그램 인터페이스 및 알고리즘, 데이터 구조를 프로그램으로 실체화 |
| 출력 산출물 | • 소스코드 및 프로그램  • 테스트 보고서 |
| 시작 기준 | • 설계서가 작성된다.  • 프로젝트 팀 내 개발자는 설계서를 이해한다. |
| 완료 기준 | • 구현 후 단위 테스트를 마친다. |
| 테스트 | 입력 산출물 | • 소스코드 |
| 수행 태스크 | • 시스템 테스트 인수 테스트  • 프로그램이 입력에 따라 요구되는 결과대로 작동하는지, 내부적 이상 여부 및 오류 발견(테스트 시나리오 작성)  • 테스트 케이스 설계  • 테스트 결과 보고서 작성 및 검토  • 프로그램 오류 발견 |
| 출력 산출물 | • 테스트 결과 보고서 |
| 시작 기준 | • 통합 테스트를 마친다.  • 단위 테스트를 마친다.  • 테스트 계획을 세운다. |
| 완료 기준 | • 시스템/인수 테스트를 마친다.  • 테스트 결과를 문서화 한다. |
| 유지보수 | 입력 산출물 | • 테스트 보고서 |
| 수행 태스크 | • 프로젝트 변경 사항 파악  • 프로젝트 검토  • 개발된 소프트웨어의 변경 사항을 수정  • 수정 유지보수, 적응 유지보수, 기능 추가 유지보수  • 프로젝트 완료 보고서 작성 및 검토 |
| 출력 산출물 | • 프로젝트 완료 보고서 |
| 시작 기준 | • 모든 테스트를 실패 없이 마친다.  • 테스트 단계를 마치고 테스트 보고서가 문서화되어 있다. |
| 완료 기준 | • 완료 보고서가 작성된다.  • 소프트웨어의 변경 사항을 지속적으로 수정이 가능한 상태 |

* 1. 개발 방법론 및 도구 선정
     1. 개발 방법론 **객체지향 방법론**  
         객체지향 방법인 UML을 사용한다.  
         - 프로젝트 개발 언어가 객체지향 언어인 JAVA 이기에 메소드와 속성이 결합한 객체를 표현하기에 적합하다.

|  |  |
| --- | --- |
| **개발 단계** | **상세 내용** |
| 객체지향 분석 | 프로그램 사용자(User)와 프로그램의 기능을 UML 언어의 Use Case와 Sequence Diagram을 이용하여 시스템 요구사항을 분석하고, ER 다이어그램과 테이블 목록 및 기술을 하여 데이터 베이스 요구사항을 분석한다. |
| 객체지향 설계 | 시스템 설계 및 객체 설계를 분리하여 설계한다.  • 시스템 설계-DB 서버를 성능 및 최적 방안을 모색해 설계한다.  • 객체 설계-UML 언어의 Class Diagram을 이용 class를 정의하고 class들 간의 관계 및 알고리즘 그리고 자료구조를 설계한다. 또한 Sequence Diagram을 이용하여 메신저 프로그램의 흐름을 설계한다.  • DB 설계-ER 다이어그램을 이용하여 테이블을 상세 기술한다. |
| 객체지향 프로그래밍 | • 객체지향 설계를 바탕으로 Java 프로그램 언어를 이용하여 데이터 구조 부분과 관련 메소드를 포함하는 여러 객체들로 구현한다. |

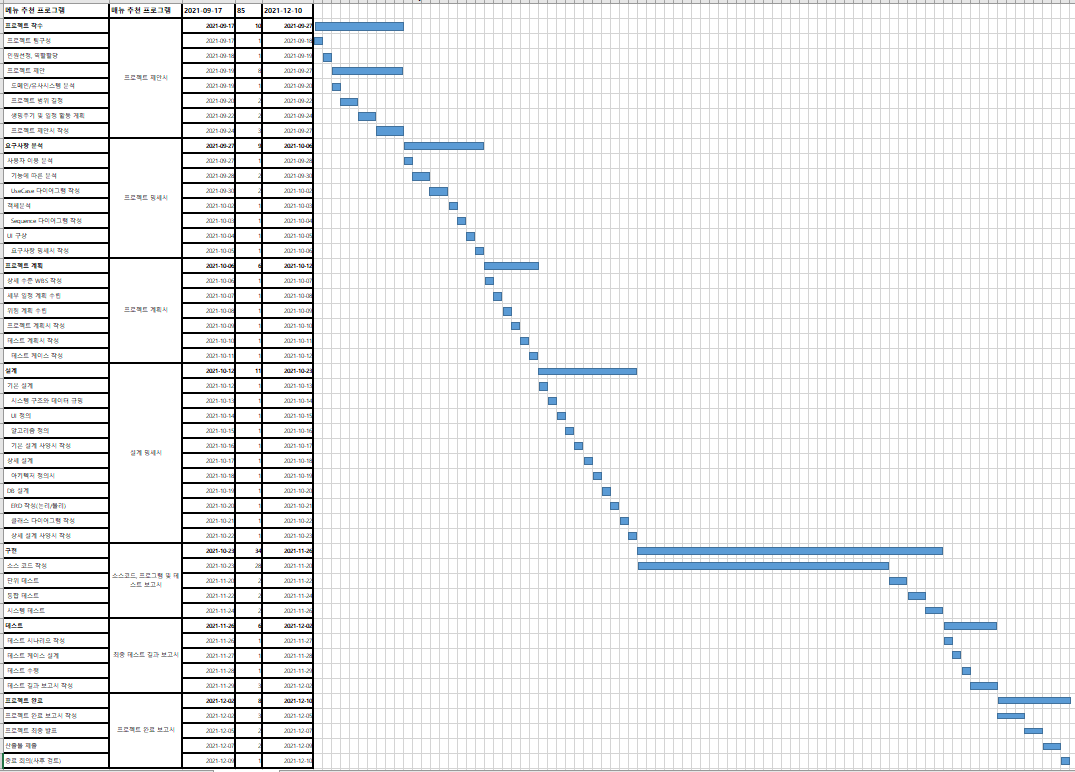
* + 1. 객체지향 방법론 절차  
        계획-설계-개발-테스트-검토(피드백) 순으로 반복적 진행

1. **개발 규모 산정**
   1. WBS



* 1. 상세일정

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **메뉴 추천 프로그램** | **매뉴 추천 프로그램** | **2021-09-17** | **85** | **2021-12-10** |
| **프로젝트 착수** | 프로젝트 제안서 | **2021-09-17** | **10** | **2021-09-27** |
| 프로젝트 팀구성 | 2021-09-17 | 1 | 2021-09-18 |
| 인원선정, 역할할당 | 2021-09-18 | 1 | 2021-09-19 |
| 프로젝트 제안 | 2021-09-19 | 8 | 2021-09-27 |
| 도메인/유사시스템 분석 | 2021-09-19 | 1 | 2021-09-20 |
| 프로젝트 범위 결정 | 2021-09-20 | 2 | 2021-09-22 |
| 생명주기 및 일정 활동 게획 | 2021-09-22 | 2 | 2021-09-24 |
| 프로젝트 제안서 작성 | 2021-09-24 | 3 | 2021-09-27 |
| **요구사항 분석** | 프로젝트 명세서 | **2021-09-27** | **9** | **2021-10-6** |
| 사용자 이용 분석 | 2021-09-27 | 1 | 2021-9-28 |
| 기능에 따른 분석 | 2021-09-28 | 2 | 2021-9-30 |
| UseCase 다이어그램 작성 | 2021-09-30 | 2 | 2021-10-2 |
| 객체분석 | 2021-10-2 | 1 | 2021-10-3 |
| Sequence 다이어그램 작성 | 2021-10-3 | 1 | 2021-10-4 |
| UI 구상 | 2021-10-4 | 1 | 2021-10-5 |
| 요구사항 명세서 작성 | 2021-10-5 | 1 | 2021-10-6 |
| **프로젝트 계획** | 프로젝트 계획서 | **2021-10-6** | **6** | **2021-10-12** |
| 상세 수준 WBS 작성 | 2021-10-6 | 1 | 2021-10-7 |
| 세부 일정 계획 수립 | 2021-10-7 | 1 | 2021-10-8 |
| 위험 계획 수립 | 2021-10-8 | 1 | 2021-10-9 |
| 프로젝트 계획서 작성 | 2021-10-9 | 1 | 2021-10-10 |
| 테스트 계획서 작성 | 2021-10-10 | 1 | 2021-10-11 |
| 테스트 케이스 작성 | 2021-10-11 | 1 | 2021-10-12 |
| **설계** | 설계 명세서 | **2021-10-12** | **11** | **2021-10-23** |
| 기본 설계 | 2021-10-12 | 1 | 2021-10-13 |
| 시스템 구조와 데이터 규명 | 2021-10-13 | 1 | 2021-10-14 |
| UI 정의 | 2021-10-14 | 1 | 2021-10-15 |
| 알고리즘 정의 | 2021-10-15 | 1 | 2021-10-16 |
| 기본 설계 사양서 작성 | 2021-10-16 | 1 | 2021-10-17 |
| 상세 설계 | 2021-10-17 | 1 | 2021-10-18 |
| 아키텍처 정의서 | 2021-10-18 | 1 | 2021-10-19 |
| DB 설계 | 2021-10-19 | 1 | 2021-10-20 |
| ERD 작성(논리/물리) | 2021-10-20 | 1 | 2021-10-21 |
| 클래스 다이어그램 작성 | 2021-10-21 | 1 | 2021-10-22 |
| 상세 설계 사양서 작성 | 2021-10-22 | 1 | 2021-10-23 |
| **구현** | 소스코드, 프로그램 및 테스트 보고서 | **2021-10-23** | **34** | **2021-11-26** |
| 소스 코드 작성 | 2021-10-23 | 28 | 2021-11-20 |
| 단위 데스트 | 2021-11-20 | 2 | 2021-11-22 |
| 통합 테스트 | 2021-11-22 | 2 | 2021-11-24 |
| 시스템 테스트 | 2021-11-24 | 2 | 2021-11-26 |
| **테스트** | 최종 테스트 결과 보고서 | **2021-11-26** | **6** | **2021-12-2** |
| 테스트 시나리오 작성 | 2021-11-26 | 1 | 2021-11-27 |
| 테스트 케이스 설계 | 2021-11-27 | 1 | 2021-11-28 |
| 테스트 수행 | 2021-11-28 | 1 | 2021-11-29 |
| 테스트 결과 보고서 작성 | 2021-11-29 | 3 | 2021-12-2 |
| **프로젝트 완료** | 프로젝트 완료 보고서 | **2021-12-2** | **8** | **2021-12-10** |
| 프로젝트 완료 보고서 작성 | 2021-12-2 | 3 | 2021-12-5 |
| 프로젝트 최종 발표 | 2021-12-5 | 2 | 2021-12-7 |
| 산출물 제출 | 2021-12-7 | 2 | 2021-12-9 |
| 종료 회의(사후 검토) | 2021-12-9 | 1 | 2021-12-10 |



* 1. 프로젝트 규모 산정 – 간이 기능 점수법

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 유형 | 내부 논리 파일 | 외부 연계 파일 | 외부 입력 | 외부 출력 | 외부 조회 |
| 가중치 | 7.5 | 5.4 | 4.0 | 5.2 | 3.9 |

|  |  |  |  |
| --- | --- | --- | --- |
| 기능 | 기능 유형 | 기능 명 | 기능 유형 |
| 사용자 관리 | 사용자 정보 관리 | 사용자 기본 정보 | 내부 논리 파일 |
| 추천 음식 관리 | 추천 음식 정보 관리 | 추천 음식 기본 정보 | 내부 논리 파일 |
| 게시판 관리 | 게시판 정보 관리 | 게시판 기본 정보 | 내부 논리 파일 |
| 음식점 관리 | 음식점 정보 관리 | 음식점 기본 정보 | 외부 연계 파일 |

**데이터 기능 점수**

데이터 기능 점수 = (ILF 개수 \* 7.5)+(EIF 개수 \* 5.4)

= (3\*7.5)+(1\*5.4)

= 22.5 + 5.4 = 27.9

**트랜잭션 기능 점수**

|  |  |  |  |
| --- | --- | --- | --- |
| 기능 명 | 기능 유형 | 기능 명 | 기능 유형 |
| 사용자 관리 | 사용자 정보 관리 | 사용자 등록 | EI |
| 사용자 정보 수정 | EI |
| 사용자 정보 조회 | EQ |
| 사용자 기본 정보 테이블 | 내부 논리 파일 |
| 비밀번호 관리 | 비밀번호 조회 | EQ |
| 비밀번호 변경 | EI |
| 추천 음식 관리 | 추천 음식 정보 관리 | 음식 추천 | EQ |
| 음식 등록 | EI |
| 음식 삭제 | EI |
| 추천 음식 정보 테이블 | 내부 논리 파일 |
| 게시판 관리 | 게시판 정보 관리 | 게시판 작성 | EI |
| 게시판 수정 | EI |
| 게시판 삭제 | EI |
| 게시판 조회 | EQ |
| 게시판 기본 정보 테이블 | 내부 논리 파일 |
| 음식점 관리 | 음식점 정보 관리 | 음식점 정보 조회 | EQ |
| 음식점 정보 테이블 | 외부 연계 파일 |

트랜잭션 기능 점수 = (EI개수 \* 4.0) + (EO개수 \* 5.2) + (EQ개수 \* 3.9)

= (8 \* 4.0) + (0 \* 5.2) + (5 \* 3.9)

= 32 + 0 + 19.5

= 51.5

**미조정 기능 점수 계산 = (데이터 기능 점수 + 트랜잭션 기능 점수)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 기능 | | 기능 수 | 평균 복잡도 | 기능 점수 | |
| 데이터 기능 점수 | ILF | 3 | 7.5 | 22.5 | 27.9 |
| EIF | 1 | 5.4 | 5.4 |
| 트랜잭션 기능 점수 | EI | 8 | 4.0 | 32 | 51.5 |
| EO | 0 | 5.2 | 0 |
| EQ | 5 | 3.9 | 19.5 |
| 계 | | | | | 79.4 |

**보정 전 개발 원가**

보정 전 개발 원가 = 미조정 기능 점수 \* 기능 점수당 단가

테이블이(가) 표시된 사진

자동 생성된 설명79.4 \* 519203 = 41,224,718.2

**보정 후 개발 원가**

보정 후 개발 원가 = 보정 전 개발 원가 \* (규모 보정 계수 \* 애플리케이션 보정 계수 \* 언어 보정 계수 \* 품질/특성 보정 계수)

= 41,224,718.2 \* (0.65 \* 1.0 \* 1.2 \* 1.0) = 32,155,280.196

1. **마일 스톤 및 일정 계획**

|  |  |  |
| --- | --- | --- |
| **마일 스톤** | | **일정 계획** |
| 착수 단계 | 프로젝트 제안서 | 21.09.17 ~ 21.09.20 |
| 분석 단계 | 요구사항 명세서 | 21.09.20 ~ 21.09.24 |
| 계획 단계 | 프로젝트 계획서 | 21.09.24 ~ 21.10.01 |
| 설계 단계 | 설계 사양서 | 21.10.01 ~ 21.10.15 |
| 구현 단계 | 소스 코드 | 21.10.15 ~ 21.11.31 |
| 테스트 단계 | 테스트 결과 보고서 | 21.11.31 ~ 21.12.08 |

**요구사항 명세서 목차**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1.** | **개요** | | | |  |
| 1.1 | | 개발 시스템의 목표 | | | p.3 |
| 1.2 | | 현재 시스템 상태 분석 | | |  |
| 1.2.1 | | | 메뉴 추천 App | |  |
| 1.2.2 | | | 성능 비교 | |  |
|  | | |  | |  |
| **2.** | **기능 요구사항** | | | |  |
| 2.1 | | 시스템 요구사항 | | |  |
| 2.1.1 | | | 사용자 스토리 | | p.4 |
| 2.1.2 | | | 유스케이스 다이어그램 | | p.5 |
| 2.1.3 | | | 액터 목록 | | p.6 |
| 2.1.4 | | | 유스케이스 목록 | | p.6 |
| 2.1.5 | | | 유스케이스 시나리오 | | p.7 |
| 2.1.6 | | | 기능 목록 | | p.17 |
| 2.2 | | 유저 인터페이스 요구사항 | | |  |
| 2.2.1 | | | 메인 화면 | | p.18 |
| 2.2.2 | | | 화면 목록 | | p.18 |
| 2.2.3 | | | 주요 화면 기술 | | p.20 |
| 2.2.3.1 | | | | UC-U01 |  |
| 2.2.3.2 | | | | UC-U02 |  |
| 2.2.3.3 | | | | 게시판 메인 화면 |  |
| 2.2.3.4 | | | | 즐겨찾기 메인 화면 |  |
|  | | |  | |  |
| **3.** | **비기능 요구사항** | | | | p.22 |
| 3.1 | | 성능 요구사항 | | |  |
| 3.2 | | 신뢰성 요구사항 | | |  |
| 3.3 | | 보안요구사항 | | |  |

**1. 개요**

**1.1 개발 시스템의 목표**  
본 프로젝트의 목적은 프로그램 사용자가 보다 편리하게 음식 메뉴를 선정할 수 있게 하는 것에 목적을 두고 있다.   
사용자는 음식 추천 기능을 통해 쉽고 빠르게 음식 메뉴를 선정할 수 있고 선택한 메뉴를 기준으로 주변 음식점 정보를 확인할 수 있도록 한다. 음식점은 매장 정보를 입력하여 사용자의 편의를 돕고 홍보를 통한 매출 증가에 목적을 둔다.

**1.2 현재 시스템 상태 분석**

1.2.1 오늘 뭐 먹지?

사용자가 어플을 통해 음식 메뉴를 추천 받을 수 있는 App이다. 랜덤으로 메뉴를 추천해주는 기능, 추천된 메뉴로 근처 식당을 추천해 주는 기능 등 앱마다 다양한 기능을 가지고 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명[사진] 오늘 뭐 먹지? 화면

(1) 다양한 기능들을 상단에 보기 쉽게 구성

(2) 음식 메뉴들을 카테고리, 음식 별로 검색할 수 있게 나누어 분류

(3) 질문을 통한 음식을 추천해주는 기능

(4) 음식 월드컵을 통한 재미요소 기능

1.2.2 성능 비교

|  |  |  |
| --- | --- | --- |
| 주요기능 | 오늘 뭐 먹지? | 메뉴추천 프로그램 |
| 회원가입, 로그인 | x | ㅇ |
| 간결한 인터페이스 | o | ㅇ |
| 랜덤 음식 추천 | o | o |
| 게시판 기능 | x | o |
| 음식 검색 기능 | x | o |
| 음식점 추천 | x | o |
| 음식점 즐겨찾기 | x | ㅇ |

**2. 기능 요구사항**

‘메뉴 추천 프로그램’은 로그인과 회원가입 등 기본적인 기능을 제공하고 메뉴를 랜덤으로 추천해주고 검색할 수 있는 기능, 다른 회원들과 정보를 주고받을 수 있는 게시판 기능, 즐겨찾기를 이용한 음식점 정보 저장 등을 제공한다.

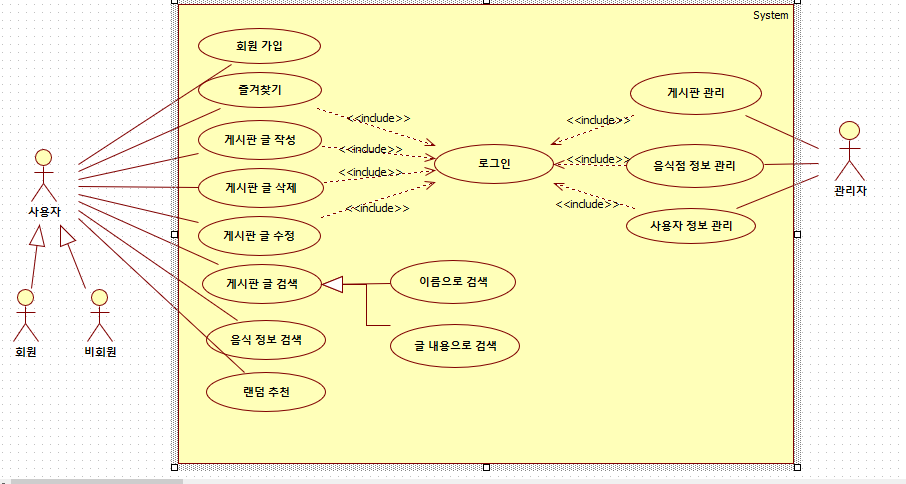
**2.1 시스템 요구사항**

프로그램 개발은 JAVA를 이용하고 개발 환경은 eclipse를 이용한다. 데이터베이스는 Oracle을 사용한다.

2.1.1 사용자 스토리

|  |
| --- |
| 사용자 요구사항 |
| ㆍ프로그램의 기능은 회원과 비회원의 기능이 나뉘어있다. |
| ㆍ회원과 비회원 모두 음식 추천 기능을 사용할 수 있다. |
| ㆍ음식 추천은 처음부터 랜덤으로 받을 수도 있고 카테고리별로 받을 수도 있다. |
| ㆍ게시판 기능은 로그인 한 회원만 사용할 수 있다. |
| ㆍ게시판은 글 작성, 수정, 삭제 기능이 있다. |
| ㆍ관리자는 회원 관리 및 게시판에 작성된 글을 삭제할 수 있다. |
| ㆍ로그인 한 사용자는 게시판에 작성된 글을 검색할 수 있다. |
| ㆍ게시판에 작성된 글을 검색할 땐 작성자와 글 내용으로 나누어서 검색할 수 있다. |
| ㆍ로그인 한 회원은 즐겨찾기 기능을 사용할 수 있다. |
| ㆍ즐겨찾기 기능은 추천받은 음식의 음식점을 즐겨찾기 목록에 추가하고 목록에서 삭제할수 있다. |
| ㆍ즐겨찾기 정보는 DB에 자동 보관ㆍ관리된다. |
| ㆍDB는 회원의 즐겨찾기 된 목록을 불러온다. |

2.1.2 유스케이스 다이어그램



2.1.3 액터 목록

|  |  |  |
| --- | --- | --- |
| 액터 | 액터 기능 | 액터 설명 |
| 사용자 | ㆍ로그인 | ㆍ프로그램을 이용하는 사람 |
| ㆍ즐겨찾기 이용 |
| ㆍ게시판 이용 |
| ㆍ게시판 검색  (내용별, 이름별) |
| ㆍ음식 추천 |
| ㆍ음식점 정보 확인 |
| 관리자 | ㆍ회원 관리 | ㆍ시스템과 시스템 이용자를 관리하는 사람 |
| ㆍ게시판 관리 |
| ㆍ음식점 정보 관리 |

2.1.4 유스케이스 목록

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 식별자 | 행위자 | 설명 | 식별자 | 행위자 | 설명 |
| UC-U01 | 사용자 | 음식 랜덤 추천 | UC-U10 | 사용자 | 게시판 검색 |
| UC-U02 | 사용자 | 음식 종류 검색 | UC-U11 | 사용자 | 이름으로 검색 |
| UC-U03 | 사용자 | 게시판 글 작성 | UC-U12 | 사용자 | 글 내용으로 검색 |
| UC-U04 | 사용자 | 게시판 글 수정 | UC-U13 | 사용자 | 회원가입 |
| UC-U05 | 사용자 | 게시판 글 삭제 | UC-A01 | 관리자 | 사용자 관리 |
| UC-U06 | 사용자 | 즐겨찾기 추가 | UC-A02 | 관리자 | 음식점 정보 관리 |
| UC-U07 | 사용자 | 즐겨찾기 삭제 | UC-A03 | 관리자 | 게시판 관리 |
| UC-U08 | 사용자 | 음식점 정보 출력 |  |  |  |
| UC-U09 | 사용자 | 로그인 |  |  |  |

2.1.5 유스케이스 시나리오

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자/명** | UC-U01 음식 추천 유스케이스 | |
| **개요** | 현대인들의 다소 심각한 선택장애로 인해 특정 다수의 인원들이 고작 점심메뉴를 가지고 꽤 심각한 스트레스를 받고 있습니다. 고로 음식을 추천하는 프로그램을 작성하여 그 고민을 조금이나마 해결할 수 있도록 도움을 드릴려고 합니다. | |
| **주 행위자** | 사용자 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 음식의 종류를 먼저 선정하고 랜덤추천버튼을 클릭한다. | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 2. 원하는 음식 종류를 선택한다.  3. 음식종류를 선택후 랜덤 버튼을 클릭한다.  7. 관련 음식점 중 원하는 음식점을 클릭하여 음식점 정보를 확인한다. | | 1. 음식 종류 리스트를 출력한다.   4. 선택된 음식종류 중 하나의 메뉴를 선정하여 출력한다.  5. 출력된 메뉴와 관련된 음식점도 같이 출력한다.  6. 다시하기 버튼도 함께 출력한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| [1. 다시하기 버튼을 누른 경우]  3. 원하는 음식 종류를 선택한다.  7. 관련 음식점 중 원하는 음식점을 클릭하여 음식점 정보를 확인한다. | | 2. 음식종류가 선택된 시점의 페이지로 돌아간다.  4. 선택된 음식종류 중 하나의 메뉴를 선정하여 출력한다.  5. 출력된 메뉴와 관련된 음식점도 같이 출력한다.  6. 다시하기 버튼도 함께 출력한다. |

|  |  |
| --- | --- |
| **예외상황 시나리오** | |
| **행위자** | **시스템** |
| **[E1. 음식종류를 선택하지 않고 버튼을 누른 경우]**  5. 관련 음식점 중 원하는 음식점을 클릭하여 음식점 정보를 확인한다. | 2. 음식 종류가 선택되지 않았으므로 모든 음식 종류를 기반으로 랜덤으로 추천한다.  3. 출력된 메뉴와 관련된 음식점도 같이 출력한다.  4. 다시하기 버튼도 함께 출력한다. |
| **사후조건** | 해당사항 없음 |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자/명** | UC-U02 음식종류 검색 유스케이스 | |
| **개요** | 사용자가 원하는 음식이 있지만, 그 이름이 생각나지 않을 때 이용할 수 있다. | |
| **주 행위자** | 사용자 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 메인 UI가 열려있는 상태에서 음식 검색 탭으로 이동해야 한다. | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 검색창에 원하는 종류의 음식 이름의 분류를 선택한다. (파스타, 국밥 등)  3. 원하는 음식을 찾았을 때 그 음식을 클릭한다. | | 2. 해당 음식 종류에 해당하는 음식을 출력한다.  4. UC-A01(음식종류추천)의 기본 흐름 5를 수행한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| 해당사항 없음 | | 해당사항 없음 |
| **예외상황 시나리오** | | |
| **행위자** | | **시스템** |
| **[E1. 음식 분류를 선택하지 않았을 경우]**  3. 다시 검색하고자 하는 경우 기본 흐름1로 돌아가서 검색한다. | | 2. 분류를 선택해달라는 문구를 출력한다. |
| **사후조건** | | 해당사항 없음 |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자/명** | UC-U03 게시판 작성 유스케이스 | |
| **개요** | 음식점에 대한 의견이나 특정 음식에 대한 의견 공유, 평가 등을 남기기 위해 게시판에 글을 작성한다. | |
| **주 행위자** | 사용자 | |
| **부 행위자** | 관리자 | |
| **사전 조건** | 사용자는 ID로 로그인하고, 메인 UI가 열려있는 상태에서 게시판 탭으로 이동해야 한다. | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 게시판 글 작성 버튼을 클릭한다.   3. 등록하고자 하는 게시물의 제목과 내용을 작성하고 작성 버튼을 클릭한다. | | 2. 작성하고자 하는 글의 태그를 선택할 수 있게 출력한다. (후기 또는 질문)  4. 등록된 게시물을 포함한 게시판을 갱신하여 출력한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| **[A1. 태그를 등록하지 않고 글을 저장하려는 경우]**  3. 등록하고자 하는 게시물의 태그를 선택하고 저장 버튼을 클릭한다. | | 2. 작성하고자 하는 글의 태그를 선택할 수 있게 출력한다.  4. 기본 흐름 4를 수행한다. |
| **예외상황 시나리오** | | |
| **행위자** | | **시스템** |
| **[E1. 글 작성 취소 버튼을 클릭하는 경우]**  3. 예 버튼을 클릭한다. | | 2. 글 작성을 취소할 것인지 예/아니요 로 물어보는 창을 클릭한다.  4. 게시판으로 돌아가 게시판을 갱신하여 출력한다. |
| **사후조건** | | 해당사항 없음 |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자/명** | UC-U04 게시판 글 수정 | |
| **개요** | 사용자는 게시판에 작성한 글을 수정할 수 있어야 한다. 사용자 본인이 작성한 게시물에는 수정 버튼이 있으며 본인의 게시물이 아닌 경우에는 수정 버튼이 없다. 게시물의 수정 버튼을 통해 게시물 작성 화면으로 진입하고 작성 화면에는 등록했던 게시물의 제목과 내용 정보가 기입 되어있다. | |
| **주 행위자** | 사용자 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 사용자는 ID로 로그인하고, 메인 UI가 열려있는 상태에서 게시판 탭으로 이동 후 본인이 작성한 게시물로 진입해 수정 기능을 실행한다. | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 2. 수정 버튼을 클릭한다.  5. 게시물의 제목, 내용을 수정하고 수정 기능을 실행한다. | | 1. 게시물의 작성자와 현재 사용자의 정보를 확인하여 일치하면 수정 버튼을 출력한다.  3. 게시물 작성 화면을 출력한다.  4. 게시물의 제목, 내용 입력란에 작성했던 게시물의 내용을 입력시킨다.  6. 게시물의 제목, 내용을 갱신하여 출력한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| 해당사항 없음 | | 해당사항 없음 |

|  |  |
| --- | --- |
| **예외상황 시나리오** | |
| **행위자** | **시스템** |
| **[E1. 게시물의 제목을 입력하지 않았을 경우]**  5. 게시물의 제목을 비워두고 수정 기능을 실행한다.  7. 기본 흐름 5를 수행한다.  **[E2. 게시물의 내용을 입력하지 않았을 경우]**  5. 게시물의 내용을 비워두고 수정 기능을 실행한다.  7. 기본 흐름 5를 수행한다. | 6. 게시물의 제목을 입력하지 않았다는 메시지를 출력한다.  8. 기본 흐름 6을 수행한다.  6. 게시물의 제목을 입력하지 않았다는 메시지를 출력한다.  8. 기본 흐름 6을 수행한다. |
| **사후조건** | 해당사항 없음 |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자/명** | UC-U05 게시판 삭제 | |
| **개요** | 자신이 게시판에 올린 글을 삭제할 수 있습니다. | |
| **주 행위자** | 사용자 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 사용자는 ID로 로그인하고, 메인 UI가 열려있는 상태에서 게시판 탭으로 이동해야 한다. 자신이 작성한 글이 있어야 한다. | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 2. 자신이 쓴 글을 확인 후 자신이 쓴 글을 클릭한다.  4. 오른쪽 하단의 삭제 버튼을 클릭한다. | | 1. 게시판 화면을 출력한다.  3. 자신이 쓴 글을 출력한다.  5. 삭제 완료 문구를 출력한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| 3. 자신의 계정 설정에 들어가서 내가 쓴 글을 확인 후 삭제하고 싶은 글을 클릭한다.  5. 오른쪽 하단의 삭제 버튼을 클릭한다. | | 2. 접속 확인 후 기본화면(메뉴추천)을 출력한다.  4. 자신이 쓴 글을 출력한다.  6. 삭제 완료 문구를 출력한다. |
| **예외상황 시나리오** | | |
| **행위자** | | **시스템** |
| 해당사항 없음 | | 해당사항 없음 |
| **사후조건** | | 삭제된 게시글은 복구 불가능하다. |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자/명** | UC-U06 즐겨찾기 추가 | |
| **개요** | 즐겨찾기 기능은 추천받은 음식에 대한 음식점 정보를 사용자의 즐겨찾기 카테고리 내에 저장할 수 있다. 또는 즐겨찾기 카테고리 내에서 추가 버튼을 통해 음식점 정보를 직접 입력하여 저장할 수 있다. 저장된 음식점 이름 정보는 사용자의 즐겨찾기 카테고리 내에서 유일해야 한다. 즐겨찾기 추가 시 즐겨찾기 추가 시 음식점 이름 정보의 중복 여부를 확인해야 한다. | |
| **주 행위자** | 사용자 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 사용자는 사용자 ID로 로그인하고 음식추천 기능을 통해 음식을 추천받는다. 이후 음식점 찾기 기능을 실행하여 음식점을 선택하여 즐겨찾기 추가 기능을 실행한다. | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 음식점 정보 화면의 즐겨찾기 추가 기능을 실행한다. | | 2. 사용자의 즐겨찾기 카테고리에 음식점 정보를 저장한다.  3. 저장된 즐겨찾기 목록을 갱신하여 출력한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| 해당사항 없음 | | 해당사항 없음 |
| **예외상황 시나리오** | | |
| **행위자** | | **시스템** |
| **[E1. 기본 흐름 1에서 즐겨찾기 카테고리 내에 해당 음식점이 이미 존재하는 경우]**   1. 이미 추가된 음식점 정보 화면의 즐겨찾기 추가 기능을 실행한다.   **[E2. 대안 흐름 7에서 즐겨찾기 카테고리 내에 해당 음식점이 이미 존재하는 경우]**  7. 이미 추가된 음식점 정보 화면의 즐겨찾기 추가 기능을 실행한다. | | 2. 이미 동일한 음식점이 존재한다는 메시지를 출력한다.  8. 이미 동일한 음식점이 존재한다는 메시지를 출력한다. |
| **사후조건** | | 해당사항 없음 |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자/명** | UC-U07 즐겨찾기 삭제 | |
| **개요** | 즐겨찾기 카테고리 내에 저장한 음식점 정보를 삭제한다. | |
| **주 행위자** | 고객 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 고객은 본인의 ID로 로그인하고, 즐겨찾기 항목으로 진입한다.  즐겨찾기에 추가된 항목이 있어야 한다. | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 2. 삭제하고 싶은 음식점 정보의 즐겨찾기 삭제 버튼(혹은 아이콘)을 클릭한다. | | 1. 해당 고객의 즐겨찾기 목록을 모두 출력한다.  3. 수정된 즐겨찾기 목록을 출력한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| **[A1. 한 번에 여러 항목을 삭제하려고 하는 경우]**  2. 여러 항목 선택 버튼을 클릭한다.  3. 삭제하고 싶은 음식점 정보를 모두 클릭하고 삭제 버튼을 클릭한다. | | 4. 기본흐름 3을 수행한다. |
| **예외상황 시나리오** | | |
| **행위자** | | **시스템** |
| 해당사항 없음 | | 해당사항 없음 |
| **사후조건** | | 해당사항 없음 |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자/명** | UC-U08 음식점 정보 출력 | |
| **개요** | 고객이 추천받은 메뉴의 음식점 정보를 출력한다. | |
| **주 행위자** | 고객 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 고객은 음식 추천 기능을 통해 음식을 추천받는다. | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 추천받은 음식 메뉴에서 음식점 정보 출력 버튼을 클릭한다.  3. 마음에 드는 음식점을 클릭한다. | | 2. 해당하는 음식점 정보 리스트를 출력한다.  4. 해당 음식점의 정보를 출력한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| **[A1. 다른 음식점의 정보를 확인하려는 경우]**  5. 뒤로 가기 버튼을 클릭한다.  7. 기본 흐름 3을 수행한다. | | 6. 기본 흐름 2를 수행한다.  8. 기본 흐름 4를 수행한다. |
| **예외상황 시나리오** | | |
| **행위자** | | **시스템** |
| **[E1. 해당 음식점의 정보가 존재하지 않을 경우]**  5. 앞선 음식점 정보 리스트를 다시 보고자 하는 경우 확인 버튼이나 뒤로 가기 버튼을 클릭한다. | | 4. 해당 음식점의 정보가 존재하지 않음을 출력한다.  6. 기본 흐름 2를 수행한다. |
| **사후조건** | | 해당사항 없음 |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자/명** | UC-U10 게시판 검색 | |
| **개요** | 모든 사용자는 게시판에 글을 검색할 수 있다. | |
| **주 행위자** | 고객 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 사용자는 ID로 로그인하고, 메인 UI가 열려있는 상태에서 게시판 탭으로 이동해야 한다.  이름으로 검색, 게시판 내용으로 검색을 선택한다. | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 검색창 왼쪽에 작성자 아이디, 글 내용, 음식점 이름으로 분류를 선택한다.  2. 게시판 맨 위의 검색창에 자신이 검색하고자 하는 단어나 문장을 작성 후 검색 버튼을 누른다. | | 3. 선택한 분류에 관련하여 검색한 단어나 문장이 포함된 게시글을 최신순으로 출력한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| 해당사항 없음 | | 해당사항 없음 |
| **예외상황 시나리오** | | |
| **행위자** | | **시스템** |
| **[E1. 검색한 내용의 글이 없는 경우]**  3. 게시판 맨 위의 검색창에 자신이 검색하고자 하는 단어나 문장을 작성 후 검색 버튼을 누른다. | | 4. 검색한 내용과 관련 있는 글이 발견되지 않으면 관련된 글이 없이 목록을 출력한다. |
| **사후조건** | | 해당사항 없음 |

2.1.6 기능 목록

|  |  |
| --- | --- |
| 기능 | 설명 |
| 회원가입, 로그인 | ㆍ회원과 비회원으로 나눠서 사용할 수 있다.  ㆍ회원으로 이용하고자 하는 사용자는 회원가입 후 로그인하여  프로그램을 이용할 수 있고, 비회원으로 이용하고자 하는 사용자는 비회원이용 버튼으로 프로그램을 이용할 수 있다. |
| 랜덤 음식 추천 | ㆍ랜덤으로 음식을 추천해준다.  ㆍ원하는 종류의 음식을 추천받고자 하는 경우엔 음식 카테고리를 선택하여 원하는 항목으로 들어가 그 음식의 종류를 추천해준다.  ㆍ그리고 그 음식의 종류에 대한 음식점을 나타내어 주변의 음식점에 대해 알 수 있게 정보를 화면에 띄워준다. |
| 음식 검색 | ㆍ음식을 검색하는 기능이다.  ㆍ예를 들어 어떤 음식(파스타, 국밥 등)을 먹고 싶은데 자세한 이름이 기억이 나지 않을 때 그 음식(파스타, 국밥 등)으로 검색하면 그 음식에 대한 자세한 분류(봉골레 파스타, 크림파스타, 순대국밥, 콩나물국밥 등)를 사용자에게 나타낸다.  ㆍ그리고 음식 추천의 기능에서 음식점을 나타내는 기능을 나타내어 주변의 음식점에 대한 정보를 화면에 띄워준다. |
| 게시판 기능 | ㆍ회원으로 시작한 사용자만 쓸 수 있는 기능이다.  ㆍ게시판은 글 작성, 수정, 삭제 기능이 있다.  ㆍ사용자는 게시판에 작성된 글 중 글 작성자 또는 글 내용으로 검색할 수 있다. |
| 즐겨찾기 기능 | ㆍ회원으로 시작한 사용자만 쓸 수 있는 기능이다.  ㆍ음식 추천을 받은 후 음식점에 대한 정보를 즐겨찾기에 추가할 수 있다.  ㆍDB에 저장되는 기능이므로 다시 로그인하여도 즐겨찾기에 추가한 음식점들에 대한 정보는 저장된다. |

**2.2 유저 인터페이스 요구사항**

2.2.1 메인 화면

|  |  |  |
| --- | --- | --- |
| 화면 | 텍스트, 화이트보드이(가) 표시된 사진  자동 생성된 설명 | 테이블이(가) 표시된 사진  자동 생성된 설명 |
| 설명 | ㆍ프로그램을 실행하면 로그인 화면이 나타난다.  ㆍ비회원 버튼을 클릭하여 비회원으로 제한된 시스템을 사용할 수 있다.  ㆍ회원은 아이디와 비밀번호를 입력하고 로그인 버튼을 눌러 회원으로 사용할 수 있다.  ㆍ회원 가입 버튼을 눌러 회원 가입을 할 수 있다.  ㆍ회원, 비회원 로그인 후 기본 메인 화면은 음식 랜덤 추천 화면이다.  ㆍ상단의 메뉴바의 버튼을 클릭해 해당 기능을 사용할 수 있다.  ㆍ비회원은 게시판, 즐겨찾기 클릭 시 기능이 제한된다는 메시지가 출력된다. | |

2.2.2 화면 목록

|  |  |  |  |
| --- | --- | --- | --- |
| 유스케이스 식별자 | 유스케이스 명 | 화면 명 | 설명 |
| UC-U01 | 메뉴 랜덤 추천 | UI\_random | 메뉴를 랜덤으로 추천받는 기능을 사용할 수 있다. |
| UC-U02 | 음식 종류 검색 | UI\_searchfood | 음식 종류를 검색한다. |
| UC-U03 | 게시판 글 작성 | UI\_board\_write | 게시판에 글을 작성한다. |
| UC-U04 | 게시판 글 수정 | UI\_board\_revise | 게시판에 작성한 글을 수정한다. |
| UC-U05 | 게시판 글 삭제 | UI\_board\_delete | 게시판에 작성한 글을 삭제한다. |
| UC-U06 | 즐겨찾기 추가 | UI\_mine\_add | 즐겨찾기 목록에 항목을 추가한다. |
| UC-U07 | 즐겨찾기 삭제 | UI\_mine\_delete | 즐겨찾기 목록에 있는 항목을 삭제한다. |
| UC-U08 | 음식점 정보 출력 | UI\_restaurant\_info | 음식점 정보를 출력한다. |
| UC-U09 | 로그인 | UI\_login | 아이디와 비밀번호를 입력하여 접속한다. |
| UC-U10 | 게시판 검색 | UI\_board\_search | 게시판에 작성된 글을 검색한다. |
| UC-U11 | 이름으로 검색 | UI\_board\_search\_name | 게시판에 작성된 글을 이름으로 검색하여 목록을 보여준다. |
| UC-U12 | 글 내용으로 검색 | UI\_board\_search\_contents | 게시판에 작성된 글의 내용 일부를 검색하여 목록을 보여준다. |
| UC-U13 | 회원가입 | UI\_sign\_up | 사용자 정보(아이디, 비밀번호, 이름)를 등록한다. |
| UC-A01 | 사용자 관리 | UI\_manage\_user | 관리자가 사용자 정보를 관리한다. |
| UC-A02 | 음식점 정보 관리 | UI\_manage\_restaurant | 관리자가 음식점 정보를 관리한다. |
| UC-A03 | 게시판 관리 | UI\_manage\_board | 관리자가 게시판 글을 관리한다. |
| - | - | UI\_board\_main | 게시판의 메인 화면이다. 작성된 글 목록을 확인한다. |
| - | - | UI\_mine\_main | 즐겨찾기의 메인 화면으로 즐겨찾기 한 목록을 확인한다. |

2.2.3 주요 화면 기술

2.2.3.1 UC-U01

|  |  |
| --- | --- |
| 유스케이스 명 | 메뉴 랜덤 추천 |
| 화면 | 테이블이(가) 표시된 사진  자동 생성된 설명 |
| 요구사항 | ㆍ음식 종류를 선택한다.  ㆍ음식 분류를 선택한다.  ㆍ주사위 버튼을 클릭하여 선택한 종류와 분류 내에서 음식을 랜덤으로 추천받는다.  ㆍ음식 종류나 분류를 선택하지 않을 시에는 전체 중에서 랜덤으로 추천받는다. |

2.2.3.2 UC-U02

|  |  |
| --- | --- |
| 유스케이스 명 | 음식 종류 검색 |
| 화면 | 테이블이(가) 표시된 사진  자동 생성된 설명 |
| 요구사항 | ㆍ대분류, 소분류를 선택한다.  ㆍ검색 버튼을 눌러 해당 분류의 음식 목록을 확인한다. |

2.2.3.3 게시판 메인

|  |  |
| --- | --- |
| 유스케이스 명 | 게시판 메인화면 |
| 화면 | 테이블이(가) 표시된 사진  자동 생성된 설명 |
| 요구사항 | ㆍ게시판에 작성된 글 목록을 확인할 수 있다.  ㆍ게시판에 작성된 글을 클릭하여 내용을 확인한다.  ㆍ게시판에 작성된 글을 검색할 수 있다.  ㆍ이름별로 검색할지 내용별로 검색할지 선택 후 검색한다.  ㆍ게시글 작성 버튼을 눌러 게시판에 글을 작성할 수 있다. |

2.2.3.4 즐겨찾기 메인

|  |  |
| --- | --- |
| 유스케이스 명 | 즐겨찾기 메인화면 |
| 화면 | 테이블이(가) 표시된 사진  자동 생성된 설명 |
| 요구사항 | ㆍ즐겨찾기 한 목록을 확인한다.  ㆍ목록의 항목을 클릭하면 항목에 대한 정보가 나타난다.  ㆍ항목을 추가, 삭제가 가능하다. |

**3. 비기능 요구사항**

3.1 성능 요구사항

|  |  |  |
| --- | --- | --- |
| 항목 | 요구사항 | 비고 |
| 반응시간 | 시스템은 사용자의 요구에 3초 이내에 처리하여 결과를 보여야 한다. |  |
| 가용시간 | 시스템은 24시간 지속적으로 서비스를 제공해야 한다. |  |

3.2 신뢰성 요구사항

|  |  |  |
| --- | --- | --- |
| 항목 | 요구사항 | 비고 |
| 데이터 신뢰성 | 사용자가 이용하는 게시판이나 추천 기능 시스템은 항상 정확한 정보를 유지 및 관리해야 한다. |  |
| 데이터 정확성 | 사용자가 입력한 글의 내용은 입력 내용 그대로 작성되어야 한다. |  |
| 데이터 무결성 | 음식 정보나 음식점 정보 등 시스템 이용과 관련된 모든 정보는 항상 중복되지 않도록 한다. |  |

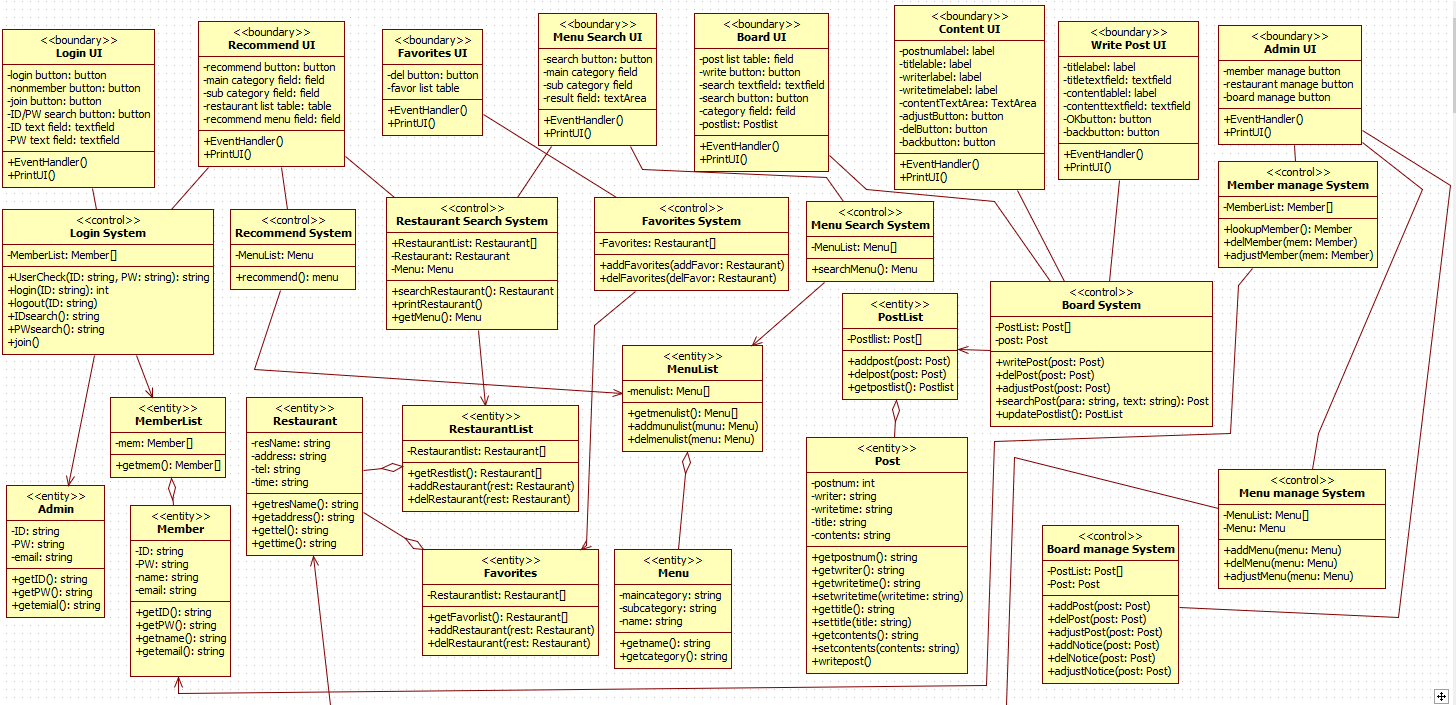
3.3 보안 요구사항

|  |  |  |
| --- | --- | --- |
| 항목 | 요구사항 | 비고 |
| 로그인 비밀번호 | 로그인 비밀번호는 6자리 이상 10자리 이하 영문 및 숫자로한다. |  |
| DB 관리 | DB의 사용자 추가/삭제/수정은 인증된 관리자만이 수행할 수 있다. |  |

**설계서 목차**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1.** | | **클래스 설계** | | | |  |
| 1.1 | | | | | 클래스 다이어그램 | p.3 |
| 1.2 | | | | | 클래스 명세서 |  |
| **2.** | | **순차 다이어그램** | | | |  |
|  | | |  | | |  |
| **3.** | **데이터베이스** | | | | |  |
| 3.1 | | | | ER 다이어그램 | |  |
| 3.2 | | | | 테이블 설계 | |  |
|  | | | | |  |  |
|  | | | | |  |  |

1. 클래스 설계

1.1 클래스 다이어그램

1.2 클래스 명세서

<메뉴 추천 클래스 명세서>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | Recommend UI | | | |
| 클래스 개요 | 메뉴를 랜덤으로 추천 받고자 할 때 보이는 화면 클래스 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| EventHandler() | void | 클릭 시 이벤트를 발생시킴 |
| PrintUI() | void | 화면을 출력한다. |
| Attributes | Attribute | Type | 설명 |
| main catagory | JTextField | 음식 대분류 필드 |
| sub catagory | JTextField | 음식 소분류 필드 |
| recommend button | JButton | 음식 검색 버튼 |
| recommend menu field | JTextField | 검색 결과 표시 필드 |
| restaurant list table | JTextField | 음식점 리스트 필드 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | Recommand System | | | |
| 클래스 개요 | 음식을 추천받고 추천 받은 음식과 관련된 식당을 추천해주는 클래스 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| recommend() | void | 메뉴를 랜덤추천한다. |
| Attributes | Attribute | Type | 설명 |
| MenuList | menu[] | 메뉴 리스트 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | Restaurant Search System | | | |
| 클래스 개요 | 음식을 추천받고 추천 받은 음식과 관련된 식당을 추천해주는 클래스 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| searchRestaurant() | void | 음식점 리스트를 찾는다 |
| printRestaurant() | void | 음식점 리스트를 출력한다. |
| getMenu() | void | 선택된 음식을 가져온다. |
| Attributes | Attribute | Type | 설명 |
| Menu | menu[] | 메뉴 리스트 |
| RestaurantList | restaurant[] | 음식점 리스트 |
| Rastaurant | restaurant[] | 음식점 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | MenuList | | | |
| 클래스 개요 | 메뉴의 정보를 가지고 있는 리스트 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| getMenuList() | void | 메뉴 리스트를 불러온다. |
| addMenuList(menu) | void | 메뉴 리스트를 추가한다. |
| delMenuList(menu) | void | 메뉴 리스트를 삭제한다. |
| Attributes | Attribute | Type | 설명 |
| MenuList | Menu[] | 메뉴 리스트 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | RestaurantList | | | |
| 클래스 개요 | 음식점 정보를 가져오는 클래스 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| getRestlist() | Restaurant[] | 음식점 리스트를 불러온다. |
| addRestaurant() | void | 음식점 정보를 추가한다. |
| delRestaurant() | void | 음식점 정보를 삭제한다. |
| Attributes | Attribute | Type | 설명 |
| Restaurantlist() | Restaurant[] | 음식점 리스트 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | Restaurant | | | |
| 클래스 개요 | 사용자가 선택한 음식점의 정보를 가지고 있는 클래스 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| getresname() | string | 음식점 이름을 가져온다. |
| getaddress() | string | 음식점 주소를 가져온다. |
| gettel() | string | 음식점 전화번호를 가져온다. |
| gettime() | string | 음식점 운영시간을 가져온다. |
| Attributes | Attribute | Type | 설명 |
| resName | string | 음식점 이름 |
| address | string | 음식점 주소 |
| tel | string | 음식점 전화번호 |
| time | string | 음식점 운영시간 |

<메뉴 검색 클래스 명세서>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | Menu Search UI | | | |
| 클래스 개요 | 원하는 음식의 종류를 검색해서 찾고자 할 때 보이는 화면 클래스 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| EventHandler() | void | 클릭 시 검색 이벤트를 발생시킴 |
| PrintUI() | void | 화면을 출력한다. |
| Attributes | Attribute | Type | 설명 |
| maincategoryField | JTextField | 음식 대분류 필드 |
| subcategoryField | JTextField | 음식 소분류 필드 |
| searchButton | JButton | 음식 검색 버튼 |
| resultField | JTextField | 검색 결과 표시 필드 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | Menu Search System | | | |
| 클래스 개요 | 음식을 검색한 후 음식 목록을 보여주는 클래스 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| searchMenu() | void | 메뉴를 검색한다. |
| Attributes | Attribute | Type | 설명 |
| MenuList | menu[] | 메뉴 리스트 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | MenuList | | | |
| 클래스 개요 | 메뉴들의 정보를 가지고 있는 리스트 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| getMenuList() | void | 메뉴 리스트를 불러온다. |
| addMenuList(menu) | void | 메뉴 리스트를 추가한다. |
| delMenuList(menu) | void | 메뉴 리스트를 삭제한다. |
| Attributes | Attribute | Type | 설명 |
| MenuList | Menu[] | 메뉴 리스트 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | Menu | | | |
| 클래스 개요 | 메뉴의 정보를 가지고 있는 리스트 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| getname() | string | 메뉴 이름을 불러온다. |
| getcategory() | string | 메뉴 분류를 불러온다. |
| Attributes | Attribute | Type | 설명 |
| maincategory | string | 음식 대분류 |
| subcategory | string | 음식 소분류 |
| name | string | 음식 이름 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | Restaurant Search System | | | |
| 클래스 개요 | 메뉴의 정보를 가지고 음식점을 추천해주는 클래스 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| searchRestaurant() | void | 음식점들을 검색한다. |
| printRestaurant() | void | 음식점들의 리스트를 출력한다. |
| getMenu() | void | 사용자가 선택한 메뉴 |
| Attributes | Attribute | Type | 설명 |
| RestaurantList | Restaurant[] | 음식점 리스트 |
| Restaurant | Restaurant | 사용자가 선택한 음식점 정보 |
| Menu | Menu | 사용자가 선택한 메뉴 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | RestaurantList | | | |
| 클래스 개요 | 음식점 정보를 가져오는 클래스 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| getRestlist() | Restaurant[] | 음식점 리스트를 불러온다. |
| addRestaurant() | void | 음식점 정보를 추가한다. |
| delRestaurant() | void | 음식점 정보를 삭제한다. |
| Attributes | Attribute | Type | 설명 |
| Restaurantlist() | Restaurant[] | 음식점 리스트 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | Restaurant | | | |
| 클래스 개요 | 사용자가 선택한 음식점의 정보를 가지고 있는 클래스 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| getresname() | string | 음식점 이름을 가져온다. |
| getaddress() | string | 음식점 주소를 가져온다. |
| gettel() | string | 음식점 전화번호를 가져온다. |
| gettime() | string | 음식점 운영시간을 가져온다. |
| Attributes | Attribute | Type | 설명 |
| resName | string | 음식점 이름 |
| address | string | 음식점 주소 |
| tel | string | 음식점 전화번호 |
| time | string | 음식점 운영시간 |

<게시판 클래스 명세서>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **클래스명** | | PostList | | |
| **클래스 개요** | | 게시물의 집합인 게시물 목록 클래스 | | |
| **Stereotype** | | entity | | |
| **Public Interface** | **Operations** | **Opertation(Parameter)** | **Return** | **설명** |
| addpost(post) | void | 게시물 목록에 게시물을 추가한다. |
| delpost(post) | void | 게시물 목록에서 게시물을 삭제한다. |
| getpostlist() | Postlist | 게시물 목록 리스트 반환 |
| **Attributes** | **Attribute** | **Type** | **설명** |
| postlist | Post[] | 게시물들의 배열. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **클래스명** | | Content UI | | |
| **클래스 개요** | | 게시물 내용 표시를 위한 화면 클래스 | | |
| **Stereotype** | | Boundary | | |
| **Public Interface** | **Operations** | **Opertation(Parameter)** | **Return** | **설명** |
| EventHandler() | void | 클릭을 통한 이벤트 발생 |
| PrintUI() | void | 화면 구성 내용 출력 |
| **Attributes** | **Attribute** | **Type** | **설명** |
| postnumlabel | label | 게시물 번호 라벨 |
| titlelabel | label | 게시물 제목 라벨 |
| writerlabel | label | 작성자 라벨 |
| writetimelabel | label | 작성 일시 라벨 |
| contentTextArea | TextArea | 내용 텍스트 영역 |
| adjustButton | button | 수정버튼 |
| delButton | button | 삭제버튼 |
| backButton | button | 돌아가기 버튼 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **클래스명** | | Write Post UI | | |
| **클래스 개요** | | 게시물 작성 및 수정을 위한 화면 클래스 | | |
| **Stereotype** | | Boundary | | |
| **Public Interface** | **Operations** | **Opertation(Parameter)** | **Return** | **설명** |
| EventHandler() | void | 클릭을 통한 이벤트 발생 |
| PrintUI() | void | 화면 구성 내용 출력 |
| **Attributes** | **Attribute** | **Type** | **설명** |
| titlelabel | label | 제목 라벨 |
| titletextfield | textfield | 제목 텍스트 입력 필드 |
| contentlabel | label | 내용 라벨 |
| contenttextfield | textfield | 내용 텍스트 입력 필드 |
| OKbutton | button | 확인 버튼 |
| backbutton | button | 취소 버튼 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **클래스명** | | PostList | | |
| **클래스 개요** | | 게시물의 집합인 게시물 목록 클래스 | | |
| **Stereotype** | | entity | | |
| **Public Interface** | **Operations** | **Opertation(Parameter)** | **Return** | **설명** |
| addpost(post) | void | 게시물 목록에 게시물을 추가한다. |
| delpost(post) | void | 게시물 목록에서 게시물을 삭제한다. |
| getpostlist() | Postlist | 게시물 목록 리스트 반환 |
| **Attributes** | **Attribute** | **Type** | **설명** |
| postlist | Post[] | 게시물들의 배열 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **클래스명** | | Content UI | | |
| **클래스 개요** | | 게시물 내용 표시를 위한 화면 클래스 | | |
| **Stereotype** | | Boundary | | |
| **Public Interface** | **Operations** | **Opertation(Parameter)** | **Return** | **설명** |
| EventHandler() | void | 클릭을 통한 이벤트 발생 |
| PrintUI() | void | 화면 구성 내용 출력 |
| **Attributes** | **Attribute** | **Type** | **설명** |
| postnumlabel | label | 게시물 번호 라벨 |
| titlelabel | label | 게시물 제목 라벨 |
| writerlabel | label | 작성자 라벨 |
| writetimelabel | label | 작성 일시 라벨 |
| contentTextArea | TextArea | 내용 텍스트 영역 |
| adjustButton | button | 수정버튼 |
| delButton | button | 삭제버튼 |
| backButton | button | 돌아가기 버튼 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **클래스명** | | Write Post UI | | |
| **클래스 개요** | | 게시물 작성 및 수정을 위한 화면 클래스 | | |
| **Stereotype** | | Boundary | | |
| **Public Interface** | **Operations** | **Opertation(Parameter)** | **Return** | **설명** |
| EventHandler() | void | 클릭을 통한 이벤트 발생 |
| PrintUI() | void | 화면 구성 내용 출력 |
| **Attributes** | **Attribute** | **Type** | **설명** |
| titlelabel | label | 제목 라벨 |
| titletextfield | textfield | 제목 텍스트 입력 필드 |
| contentlabel | label | 내용 라벨 |
| contenttextfield | textfield | 내용 텍스트 입력 필드 |
| OKbutton | button | 확인 버튼 |
| backbutton | button | 취소 버튼 |

<로그인 클래스 명세서>

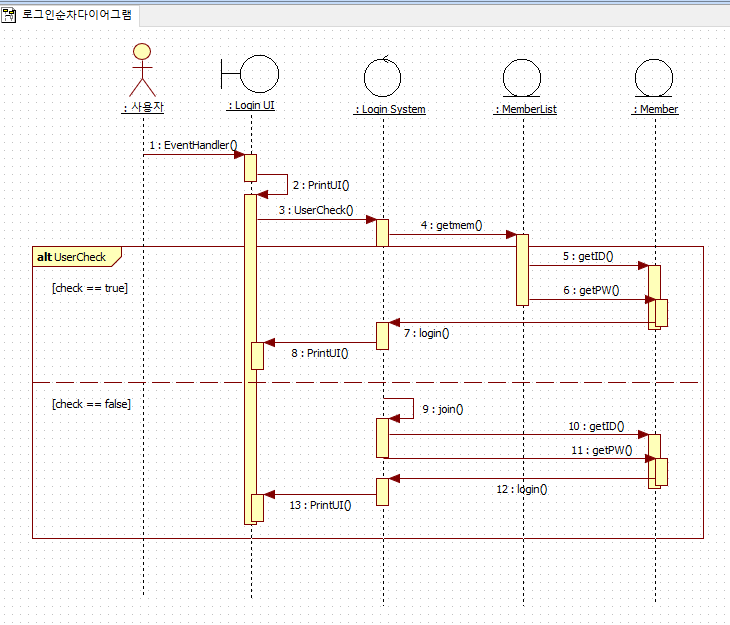
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | Login UI | | | |
| 클래스 개요 | 프로그램 시작 시 보이는 로그인 화면 클래스 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| EventHandler() | void | 클릭 시 이벤트를 발생시킴 |
| PrintUI() | void | 화면을 출력 |
| Attributes | Attribute | Type | 설명 |
| loginbutton | button | 로그인 버튼 |
| nonmemberbutton | button | 비회원 로그인 버튼 |
| joinbutton | button | 회원가입 버튼 |
| ID/PWsearchbutton | button | 아이디/패스워드 찾기 버튼 |
| IDtextfield | textfield | 아이디 입력 텍스트 필드 |
| PWtextfield | textfield | 패스워드 입력 텍스트 필드 |

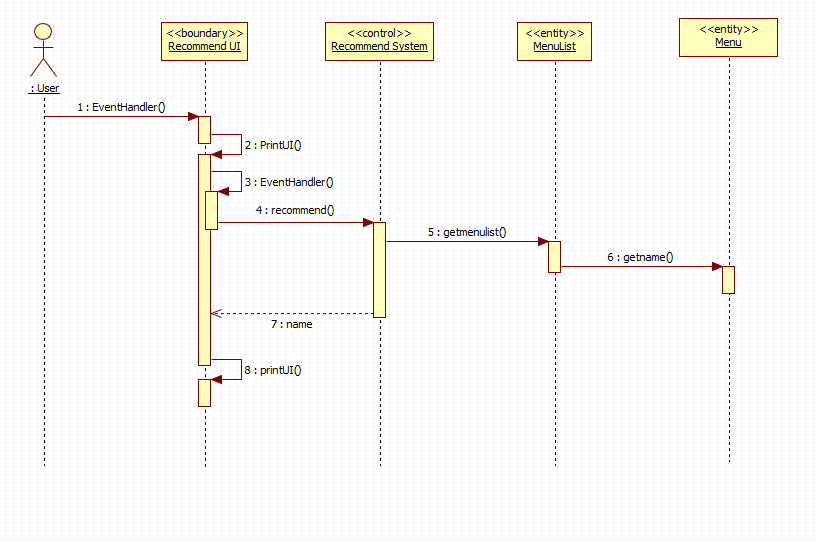
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | Login System | | | |
| 클래스 개요 | 로그인 관련 기능들을 위한 클래스 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| UserCheck(ID, PW) | string | ID, PW를 확인해서 올바른 사용자인지 확인 |
| login(ID) | int | 사용자 계정으로 로그인 |
| logout(ID) | void | 사용자 로그아웃 |
| IDsearch() | string | ID를 검색 |
| PWsearch() | string | PW를 검색 |
| join() | string | 회원가입 진행 |
| Attributes | Attribute | Type | 설명 |
| MemberList | Member[] | 사용자 리스트(목록) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | MemberList | | | |
| 클래스 개요 | 사용자의 정보 목록인 리스트 클래스 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| getmem() | Member[] | 사용자 리스트를 불러온다. |
| Attributes | Attribute | Type | 설명 |
| mem | Member[] | 사용자 리스트 |

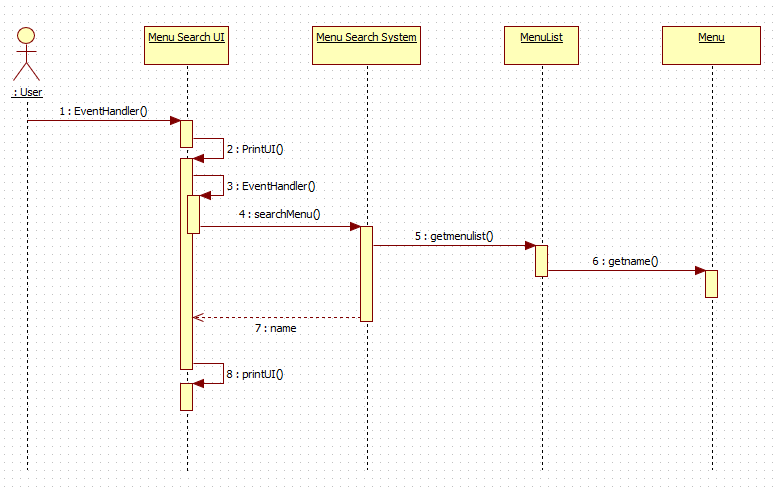
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 클래스명 | Admin | | | |
| 클래스 개요 | 메뉴의 정보를 가지고 있는 리스트 | | | |
| Public  Interface | Operations | Operation | Return | 설명 |
| getID() | string | 관리자 ID 정보를 반환 |
| getPW() | string | 관리자 PW 정보를 반환 |
| getemail() | string | 관리자 이메일 정보를 반환 |
| Attributes | Attribute | Type | 설명 |
| ID | string | 관리자 아이디 |
| PW | string | 관지라 패스워드 |
| email | string | 관리자 이메일 |

2. 시퀸스

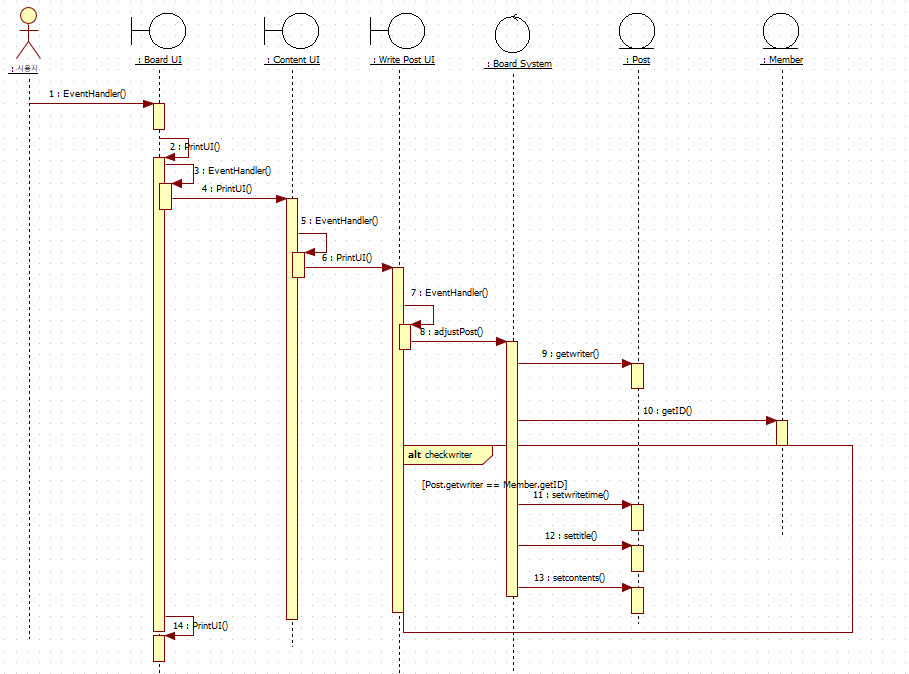
<로그인순차다이어그램>

<랜덤추천순차다이어그램>

<메뉴 검색 다이어그램>

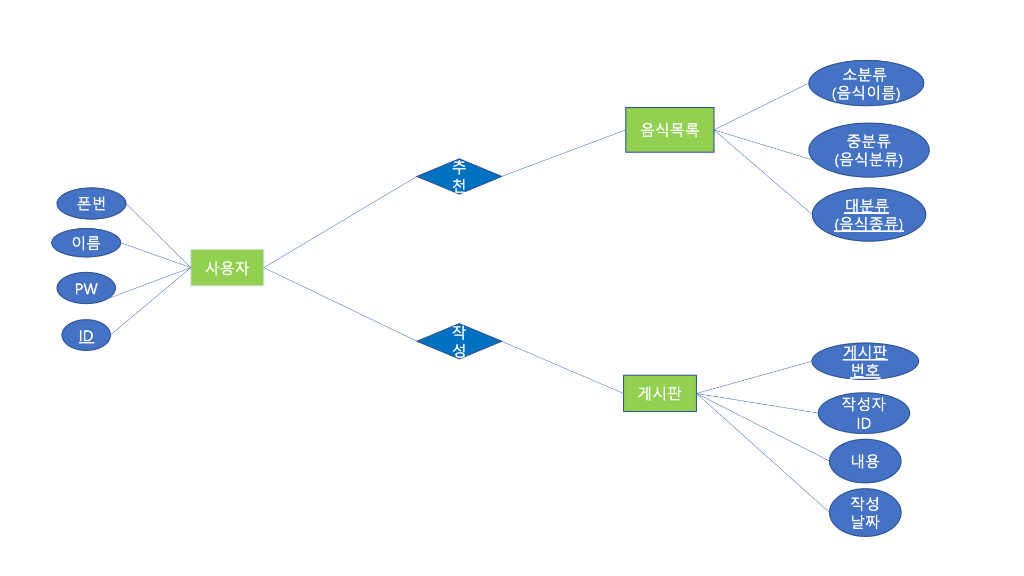


<게시판순차다이어그램>



3. 데이터베이스

3.1 ER 다이어그램



3.2 테이블 설계

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 테이블명 | | 사용자 | | | |
| 테이블 | 속성 | | 필드명 | 데이터타입 | 비고 |
| 사용자 ID | | userid | Varchar(12) | PK |
| 비밀번호 | | userpw | Varchar(12) |  |
| 사용자 이름 | | username | Varchar(8) |  |
| 전화번호 | | userphone | Varchar(12) |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 테이블명 | | 음식 | | | |
| 테이블 | 속성 | | 필드명 | 데이터타입 | 비고 |
| 소분류(음식이름) | | foodname | Varchar(12) | PK |
| 중분류(음식종류) | | subcatagory | Varchar(6) |  |
| 대분류(음식종류) | | maincatagory | Varchar(6) |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 테이블명 | | 게시판 | | | |
| 테이블 | 속성 | | 필드명 | 데이터타입 | 비고 |
| 게시판번호 | | boardnumber | number | PK |
| 게시판작성일자 | | boarddate | timestamp |  |
| 리뷰내용 | | contents | Nvarchar |  |
| 작성자아이디 | | userid | Varchar(12) | FK |

**프로젝트 최종 보고서 목차**

|  |  |  |  |
| --- | --- | --- | --- |
| **1.** | **프로젝트 개요** | |  |
| 1.1 | | 프로젝트 주제 |  |
| 1.2 | | 프로젝트 목적 및 소개 |  |
|  | |  |  |
| **2.** | **소프트웨어 설계 및 구현** | |  |
|  | |  |  |
| **3.** | **테스팅** | |  |
| 3.1 | | 블랙박스 |  |
| 3.2 | | 화이트박스 |  |
|  | |  |  |
| **4.** | **소감문** | |  |

1. **프로젝트 개요**

1.1 프로젝트 주제

- 프로젝트명 : 메뉴 추천 프로그램

- 프로젝트 기간 : 21.09.17 ~ 21.12.10

- 개발 목적 : JAVA를 이용하여 음식 메뉴를 추천하는 기능의 프로그램을 개발하여

프로젝트에 대한 경험을 쌓고 사용자들이 만족할 수 있도록 한다.

- 구현 환경 : Windows 10, Oracle, JDK 1.8

- 개발 언어 : JAVA

- 개발 도구 : Eclipse

- 팀 구성 : 조영문, 유하영, 유한규, 이상미

* 1. 프로젝트 목적 및 소개

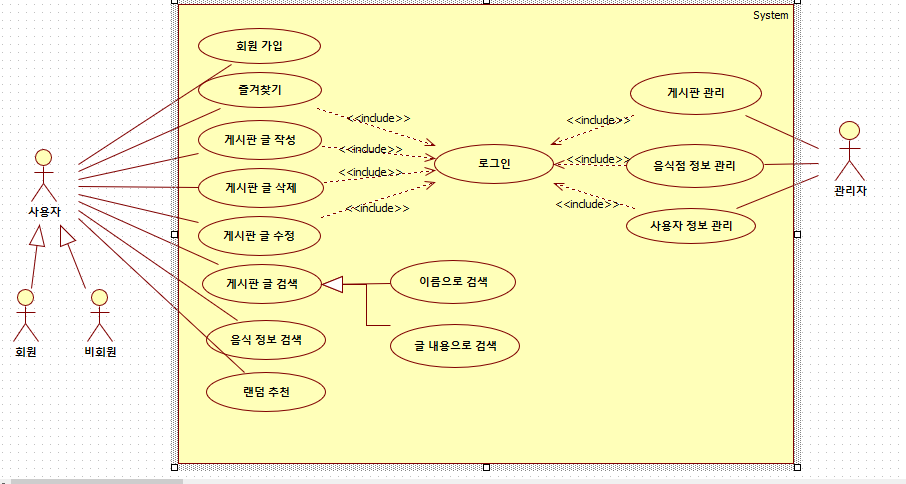
본 프로젝트의 목적은 프로그램 사용자가 보다 편리하게 음식 메뉴를 선정할 수 있게 하는 것에 목적을 두고 있다.

사용자는 음식 추천 기능을 통해 쉽고 빠르게 음식 메뉴를 선정할 수 있고 선택한 메뉴를 기준으로 주변 음식점 정보를 확인할 수 있도록 한다. 음식점은 매장 정보를 입력하여 사용자의 편의를 돕고 홍보를 통한 매출 증가에 목적을 두고 있다.

1. **소프트웨어 설계 및 구현**

2.1 요구명세서 및 설계서

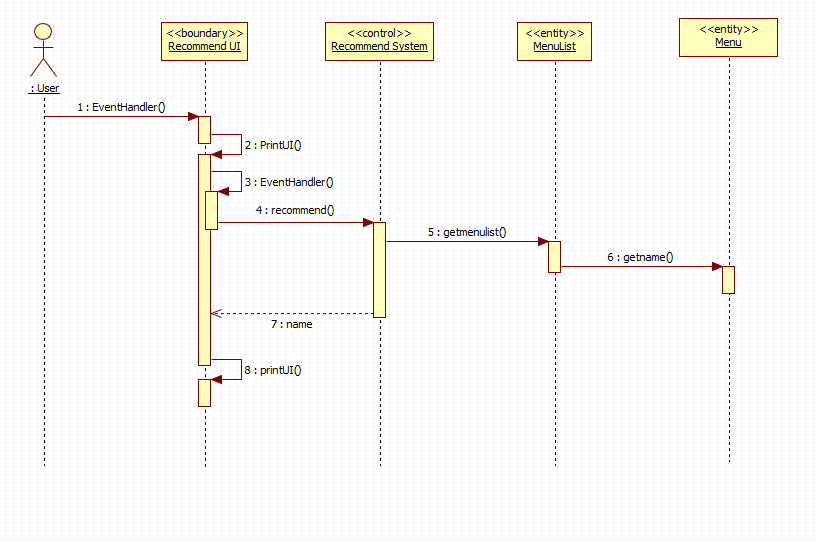
2.1.1 유스케이스 다이어그램



2.1.2 유스케이스 시나리오 및 순차 다이어그램

1.1) 음식 추천 시나리오

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자/명** | UC-U01 음식 추천 유스케이스 | |
| **개요** | 현대인들의 다소 심각한 선택장애로 인해 특정 다수의 인원들이 고작 점심메뉴를 가지고 꽤 심각한 스트레스를 받고 있습니다. 고로 음식을 추천하는 프로그램을 작성하여 그 고민을 조금이나마 해결할 수 있도록 도움을 주려고 합니다. | |
| **주 행위자** | 사용자 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 음식의 종류를 먼저 선정하고 랜덤추천 버튼을 클릭한다. | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 2. 원하는 음식 종류를 선택한다.  3. 음식종류를 선택 후 랜덤 버튼을 클릭한다.  7. 관련 음식점 중 원하는 음식점을 클릭하여 음식점 정보를 확인한다. | | 1. 음식 종류 리스트를 출력한다.   4. 선택된 음식종류 중 하나의 메뉴를 선정하여 출력한다.  5. 출력된 메뉴와 관련된 음식점도 같이 출력한다.  6. 다시하기 버튼도 함께 출력한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| [1. 다시하기 버튼을 누른 경우]  3. 기본 흐름 3부터 실행한다. | | 2. 음식종류가 선택된 시점의 페이지로 돌아간다. |
| **예외상황 시나리오** | | |
| **행위자** | | **시스템** |
| [1. 음식 종류를 선택하지 않고 랜덤 버튼을 누른 경우] | | 2. 음식 종류가 선택되지 않았으므로 모든 음식 종류를 기반으로 랜덤으로 추천한다.  3. 기본 흐름5부터 실행한다. |
| **사후조건** | | 해당사항 없음 |

 1.2) 음식 추천 순차 다이어그램

1.3) 음식 추천 화면



일식 대분류를 선택하고 튀김 소분류를 선택 후 추천을 클릭해서 [덴푸라]가 추천된 화면

대분류와 소분류를 선택하지 않고 랜덤추천을 클릭해도 음식이 추천된다.

|  |
| --- |
| 음식 추천 시스템 코드 |
| **public** String recommend(String main, String mid) **throws** SQLException{  String sql=**null**;  String sql1="SELECT \* FROM (SELECT \* FROM food ORDER BY dbms\_random.value)"  + "WHERE rownum=1";  String sql2="SELECT \* FROM (SELECT \* FROM food ORDER BY dbms\_random.value)"  + "WHERE maincategory=? and rownum=1";  String sql3="SELECT \* FROM (SELECT \* FROM food ORDER BY dbms\_random.value)"  + "WHERE maincategory=? and middlecategory=? and rownum=1";  **if**(main.equals("") && mid.equals("")) {  sql=sql1;  }  **else** **if**(mid.equals("") && !main.equals("")) {  sql=sql2;  }  **else** sql=sql3;    String val=**null**;  **try** {  PreparedStatement pstmt = con.prepareStatement(sql,ResultSet.***TYPE\_SCROLL\_INSENSITIVE***, ResultSet.***CONCUR\_UPDATABLE***);  **if**(sql.equals(sql1)) {    }  **else** **if**(sql.equals(sql2)) {  pstmt.setString(1, main);  }  **else** {  pstmt.setString(1, main);  pstmt.setString(2, mid);  }    ResultSet rs = pstmt.executeQuery();  **if**(rs.next()) val=rs.getString(1);  pstmt.close();  }  **catch**(SQLException e) {  e.printStackTrace();  }  **finally** {  con.close();  }  **return** val;  } |

|  |
| --- |
| 메뉴 리스트를 불러오는 DB 코드 |
| **public** MenuList() **throws** SQLException {  String sql="SELECT \* FROM food";    **try** {  PreparedStatement pstmt = con.prepareStatement(sql,ResultSet.***TYPE\_SCROLL\_INSENSITIVE***, ResultSet.***CONCUR\_UPDATABLE***);    ResultSet rs = pstmt.executeQuery();    rs.last();  **int** rnum=rs.getRow();  rs.beforeFirst();    menulist=**new** Menu[rnum];  **int** i=0;  **while**(rs.next()) {  menulist[i]=**new** Menu(rs.getString(1), rs.getString(2),rs.getString(3));  i++;  }  pstmt.close();  rs.close();  }  **catch**(SQLException e) {  e.printStackTrace();  }  **finally** {  con.close();  }  }  **public** Menu[] getmenulist() {  **return** **this**.menulist;  } |

2.1) 음식 검색 시나리오

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자/명** | UC-U02 음식종류 검색 유스케이스 | |
| **개요** | 사용자가 원하는 음식이 있지만, 그 이름이 생각나지 않을 때 이용할 수 있다. | |
| **주 행위자** | 사용자 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 메인 UI가 열려있는 상태에서 음식 검색 탭으로 이동해야 한다. | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 검색창에 원하는 종류의 음식 이름의 분류를 선택한다. (파스타, 국밥 등)  3. 원하는 음식을 찾았을 때 그 음식을 클릭한다. | | 2. 해당 음식 종류에 해당하는 음식을 출력한다.  4. UC-A01(음식종류추천)의 기본 흐름 5를 수행한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| 해당사항 없음 | | 해당사항 없음 |
| **예외상황 시나리오** | | |
| **행위자** | | **시스템** |
| **[E1. 음식 분류를 선택하지 않았을 경우]**  3. 다시 검색하고자 하는 경우 기본 흐름1로 돌아가서 검색한다. | | 2. 분류를 선택해달라는 문구를 출력한다. |
| **사후조건** | | 해당사항 없음 |

2.2) 음식 검색 순차 다이어그램

텍스트, 지도, 실내이(가) 표시된 사진

자동 생성된 설명

2.3) 음식 검색 순차 다이어그램

테이블이(가) 표시된 사진

자동 생성된 설명

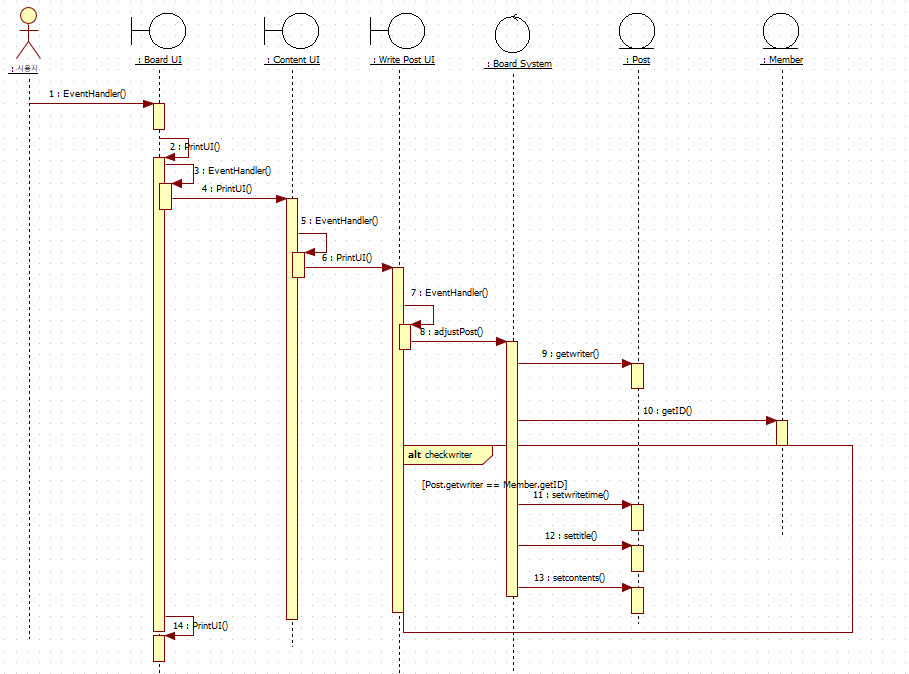
원하는 메뉴가 있지만 이름이 생각나지 않을 때 이용할 수 있는 기능이다.

중식에서 만두를 검색하면 목록을 불러와 사용자에게 만두와 관련된 메뉴들을 보여준다.

|  |
| --- |
| 음식 검색 시스템 코드 |
| **public** Menu[] SearchMenu(String a, String b) **throws** SQLException {  Menu[] menus=**null**;  **int** mnum=0;  **int** snum=0;    **if**(a.equals("")&&b.equals("")) {  menus=Menulist;  }  **else** **if**(!a.equals("")&&b.equals("")) {  **for**(**int** i=0; i<Menulist.length;i++) {  **if**(Menulist[i].get\_maincategory().equals(a)) {  mnum++;  }  }  menus=**new** Menu[mnum];  **int** k=0;  **for**(**int** i=0; i<Menulist.length;i++) {  **if**(Menulist[i].get\_maincategory().equals(a)) {  menus[k]=Menulist[i];  k++;  }  }  }  **else** {  **for**(**int** i=0;i<Menulist.length; i++) {  **if**(Menulist[i].get\_maincategory().equals(a)&&Menulist[i].get\_subcategory().equals(b)) {  mnum++;  }  }  menus=**new** Menu[mnum];  **int** k=0;  **for**(**int** i=0; i<Menulist.length;i++) {  **if**(Menulist[i].get\_maincategory().equals(a)&&Menulist[i].get\_subcategory().equals(b)) {  menus[k]=Menulist[i];  k++;  }  }  }      **return** menus;  } |

3.1) 게시판 시나리오

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자/명** | UC-U03 게시판 작성 유스케이스 | |
| **개요** | 음식점에 대한 의견이나 특정 음식에 대한 의견 공유, 평가 등을 남기기 위해 게시판에 글을 작성한다. | |
| **주 행위자** | 사용자 | |
| **부 행위자** | 관리자 | |
| **사전 조건** | 사용자는 ID로 로그인하고, 메인 UI가 열려 있는 상태에서 게시판 탭으로 이동해야 한다. | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 게시판 글 작성 버튼을 클릭한다.   3. 등록하고자 하는 게시물의 제목과 내용을 작성하고 작성 버튼을 클릭한다. | | 2. 게시물 작성 UI로 이동한다.  4. 등록된 게시물을 포함한 게시판을 갱신하여 출력한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| **[A1. 제목 또는 내용을 입력하지 않고 작성하는 경우]**  3. 기본 흐름 4를 수행한다. | | 2. 제목 또는 내용이 입력되지 않았다는 안내 문구를 보여준다.  4. 기본 흐름 4를 수행한다. |
| **예외상황 시나리오** | | |
| **행위자** | | **시스템** |
| **[E1. 글 작성 취소 버튼을 클릭하는 경우]**  3. 예 버튼을 클릭한다. | | 2. 글 작성을 취소할 것인지 예/아니요 로 물어보는 창을 클릭한다.  4. 게시판으로 돌아가 게시판을 갱신하여 출력한다. |
| **사후조건** | | 해당사항 없음 |

 3.2) 게시판 순차 다이어그램

3.3) 게시판 화면

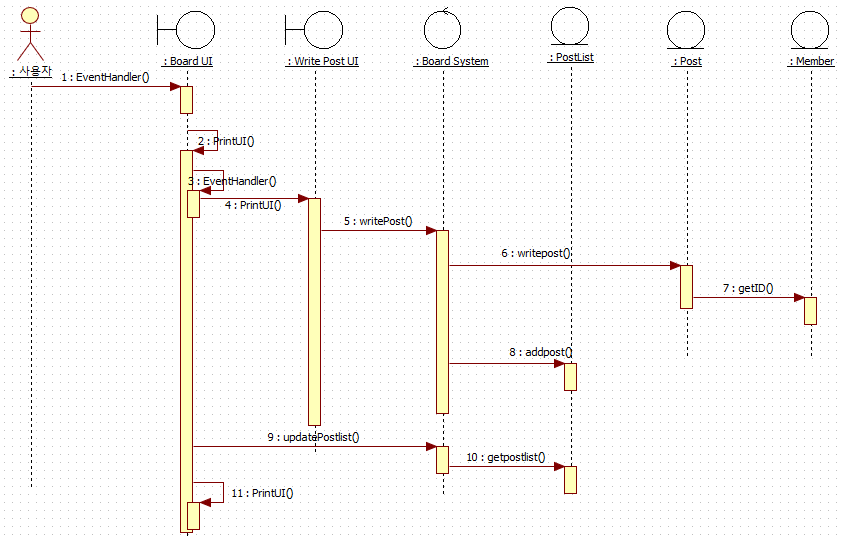
테이블이(가) 표시된 사진

자동 생성된 설명

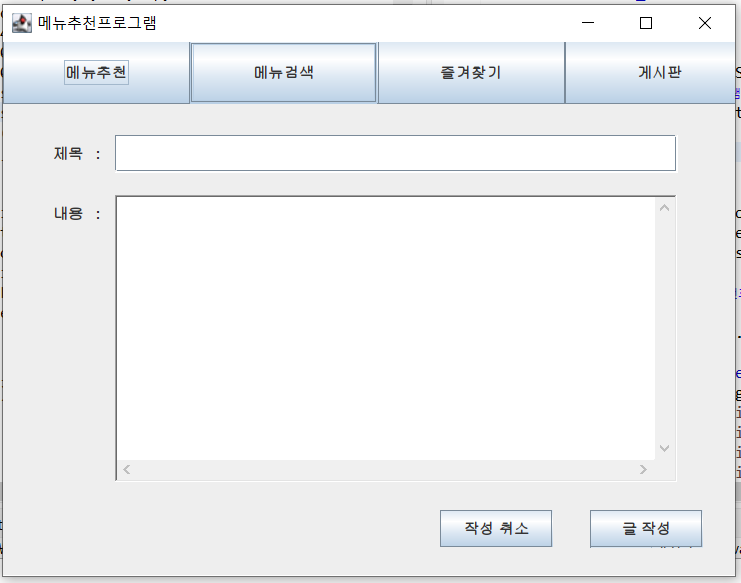
|  |
| --- |
| 게시물 목록 코드 |
| **public** Postlist() **throws** SQLException {  String query = "select \* from POST order by 작성일시 desc";    **try** {  PreparedStatement pstmt = con.prepareStatement(query,ResultSet.***TYPE\_SCROLL\_INSENSITIVE***, ResultSet.***CONCUR\_UPDATABLE***);    ResultSet rs = pstmt.executeQuery();    rs.last();  **int** rnum=rs.getRow();  rs.beforeFirst();    Postlist=**new** Post[rnum];  **int** i=0;    **while**(rs.next()) {  String num=rs.getString(1);  String title=rs.getString(2);  String contents=rs.getString(3);  String writer=rs.getString(4);  String writetime=rs.getString(5);    Postlist[i]=**new** Post(Integer.*parseInt*(num), title, contents, writer, writetime);  i++;  }  pstmt.close();  rs.close();  }  **catch** (SQLException e) {  e.printStackTrace();  }  **finally** {  con.close();  }  }  **public** Post[] get\_postlist() {  **return** **this**.Postlist;  }  **public** **static** **void** main(String[] args) **throws** SQLException {  Postlist PP=**new** Postlist();  } |

|  |
| --- |
| 게시물 목록 호출 코드 |
| String header[]= {"번호", "제목", "내용", "작성자","작성일시"};  String[][] conts;  conts=**new** String[pll.length][header.length];  **for**(**int** i=0; i<pll.length; i++) {  conts[i][0]=Integer.*toString*(pll[i].get\_postnum());  conts[i][1]=pll[i].get\_title();  conts[i][2]=pll[i].get\_contents();  conts[i][3]=pll[i].get\_writer();  conts[i][4]=pll[i].get\_writetime();  }  DefaultTableModel model=**new** DefaultTableModel(conts, header) {  **public** **boolean** isCellEditable(**int** i, **int** c){ **return** **false**; }  };  postlisttable=**new** JTable(model);  pane=**new** JScrollPane(postlisttable);  // 컬럼 사이즈 지정  postlisttable.getColumnModel().getColumn(0).setPreferredWidth(40);  postlisttable.getColumnModel().getColumn(1).setPreferredWidth(100);  postlisttable.getColumnModel().getColumn(2).setPreferredWidth(250);  postlisttable.getColumnModel().getColumn(3).setPreferredWidth(70);  postlisttable.getColumnModel().getColumn(4).setPreferredWidth(120); |

3.3) 게시물 작성 시나리오



3.4) 게시물 작성 화면



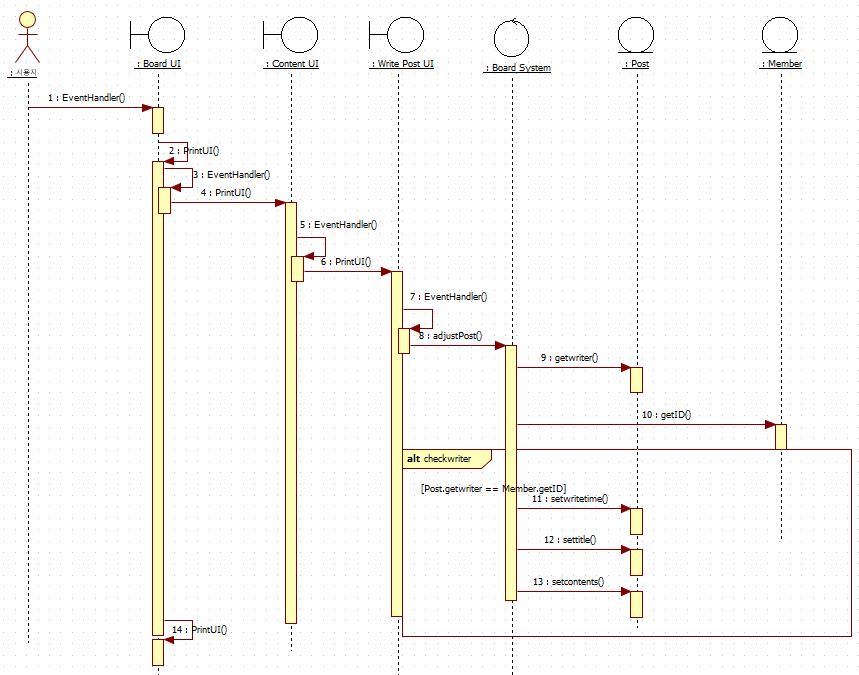
|  |
| --- |
| 게시물 작성 코드 |
| **public** **int** writePost(Post post) **throws** SQLException {  String sql="insert into POST values(a.nextval, ?, ?, ?, TO\_CHAR(sysdate, 'YYYYMMDDHH24MISS'))";  **try** {  PreparedStatement pstmt = con.prepareStatement(sql);  pstmt.setString(1, post.get\_title());  pstmt.setString(2, post.get\_contents());  pstmt.setString(3, post.get\_writer());    **int** num=pstmt.executeUpdate();    pstmt.close();  **return** num;  }  **catch**(SQLException e) {  e.printStackTrace();  **return** -1;  }  **finally** {  con.close();  }  } |

|  |
| --- |
| 게시물 작성 클릭 코드 |
| **if**(e.getSource()==OK\_btn) {  // 작성하기 버튼 클릭  **if**(OK\_btn.getText().equals("글 작성")) {  **if** ("".equals(title\_TF.getText())) {  JOptionPane.*showMessageDialog*(**null**, "제목을 입력해주세요.", "게시물 작성", JOptionPane.***INFORMATION\_MESSAGE***);  } **else** **if** ("".equals(contents\_TA.getText())) {  JOptionPane.*showMessageDialog*(**null**, "내용을 입력해주세요.", "게시물 작성", JOptionPane.***INFORMATION\_MESSAGE***);  } **else** {  BoardSystem BS=**new** BoardSystem(mm);  **try** {  Post post=**new** Post(0, title\_TF.getText(), contents\_TA.getText(), mm.getID(), **null**);  **int** num=BS.writePost(post);  **if**(num>0) {  JOptionPane.*showMessageDialog*(**null**, "글이 작성되었습니다.", "게시물 작성", JOptionPane.***INFORMATION\_MESSAGE***);  BoardUI BUI=**new** BoardUI();  dispose();  }  } **catch** (SQLException e1) {  // **TODO** Auto-generated catch block  e1.printStackTrace();  }  }  }  } |

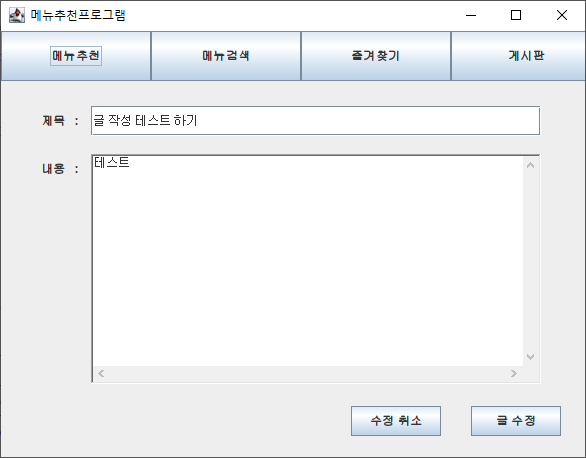
4.1) 게시물 수정 시나리오

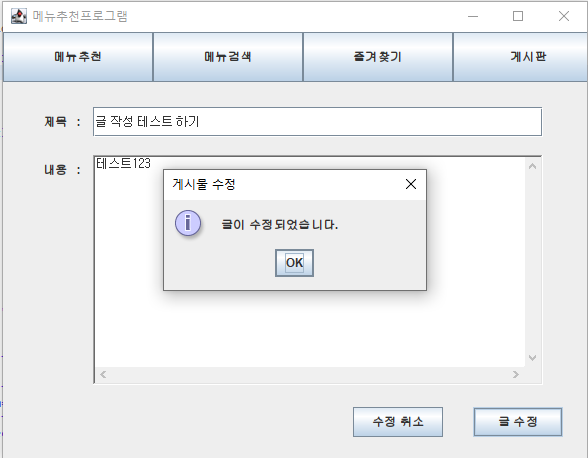
|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자/명** | UC-U04 게시판 글 수정 | |
| **개요** | 사용자는 게시판에 작성한 글을 수정할 수 있어야 한다. 사용자 본인이 작성한 게시물에는 수정 버튼이 있으며 본인의 게시물이 아닌 경우에는 수정 버튼이 없다. 게시물의 수정 버튼을 통해 게시물 작성 화면으로 진입하고 작성 화면에는 등록했던 게시물의 제목과 내용 정보가 기입 되어있다. | |
| **주 행위자** | 사용자 | |
| **부 행위자** | 관리자 | |
| **사전 조건** | 사용자는 ID로 로그인하고, 메인 UI가 열려있는 상태에서 게시판 탭으로 이동 후 본인이 작성한 게시물로 진입해 수정 기능을 실행한다. | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 2. 수정 버튼을 클릭한다.  5. 게시물의 제목, 내용을 수정하고 수정 기능을 실행한다. | | 1. 게시물의 작성자와 현재 사용자의 정보를 확인하여 일치하면 수정 버튼을 출력한다.  3. 게시물 작성 화면을 출력한다.  4. 게시물의 제목, 내용 입력란에 작성했던 게시물의 내용을 입력시킨다.  6. 게시물의 제목, 내용을 갱신하여 출력한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| 해당사항 없음 | | 해당사항 없음 |
| **예외상황 시나리오** | | |
| **행위자** | | **시스템** |
| **[E1. 게시물의 제목을 입력하지 않았을 경우]**  5. 게시물의 제목을 비워두고 수정 기능을 실행한다.  7. 기본 흐름 5를 수행한다.  **[E2. 게시물의 내용을 입력하지 않았을 경우]**  5. 게시물의 내용을 비워두고 수정 기능을 실행한다.  7. 기본 흐름 5를 수행한다. | | 6. 게시물의 제목을 입력하지 않았다는 메시지를 출력한다.  8. 기본 흐름 6을 수행한다.  6. 게시물의 제목을 입력하지 않았다는 메시지를 출력한다.  8. 기본 흐름 6을 수행한다. |
| **사후조건** | | 해당사항 없음 |

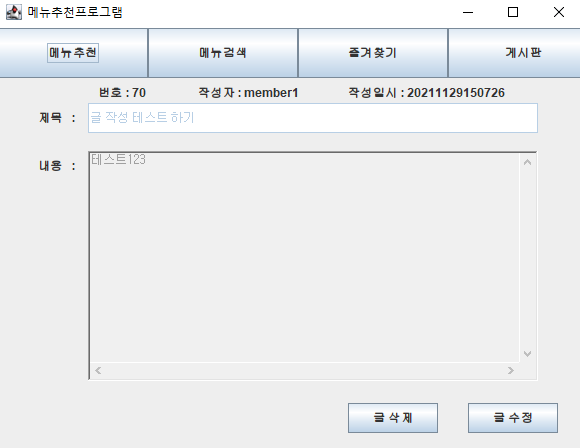
4.2) 게시물 수정 순차 다이어그램



4.3) 게시물 수정 화면



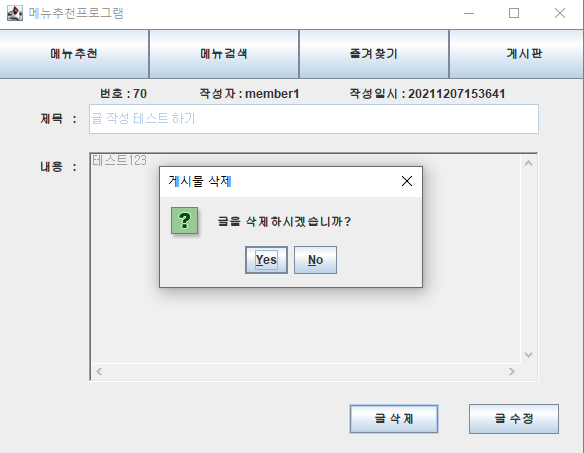


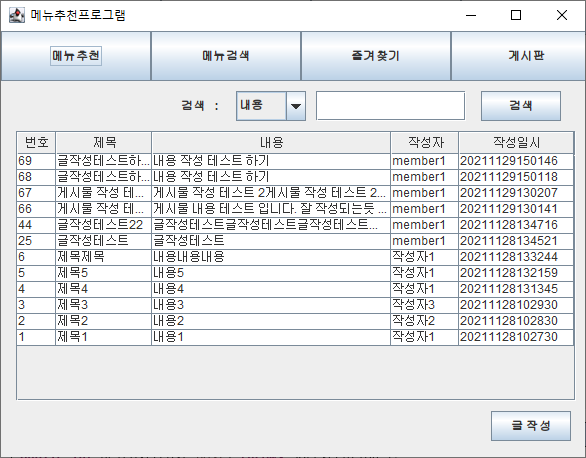


|  |
| --- |
| 게시물 수정 코드 |
| **public** **int** adjustPost(Post post) **throws** SQLException {  String sql="update POST set 제목=?, 내용=?, 작성일시=TO\_CHAR(sysdate, 'YYYYMMDDHH24MISS') where 게시물번호=?";  **try** {  PreparedStatement pstmt = con.prepareStatement(sql);  pstmt.setString(1, post.get\_title());  pstmt.setString(2, post.get\_contents());  pstmt.setInt(3, post.get\_postnum());    **int** num=pstmt.executeUpdate();    pstmt.close();  **return** num;  }  **catch**(SQLException e) {  e.printStackTrace();  **return** -1;  }  **finally** {  con.close();  }  } |

|  |
| --- |
| 게시물 수정 클릭 코드 |
| **if**(OK\_btn.getText().equals("글 수정")) {  **if** ("".equals(title\_TF.getText())) {  JOptionPane.*showMessageDialog*(**null**, "제목을 입력해주세요.", "게시물 수정", JOptionPane.***INFORMATION\_MESSAGE***);  } **else** **if** ("".equals(contents\_TA.getText())) {  JOptionPane.*showMessageDialog*(**null**, "내용을 입력해주세요.", "게시물 수정", JOptionPane.***INFORMATION\_MESSAGE***);  } **else** {  BoardSystem BS=**new** BoardSystem(mm);  pp.set\_title(title\_TF.getText());  pp.set\_contents(contents\_TA.getText());  **int** num;  **try** {  num = BS.adjustPost(pp);  **if**(num>0) {  JOptionPane.*showMessageDialog*(**null**, "글이 수정되었습니다.", "게시물 수정", JOptionPane.***INFORMATION\_MESSAGE***);  ContentUI CUI=**new** ContentUI(mm, pp);  dispose();  }  } **catch** (SQLException e1) {  // **TODO** Auto-generated catch block  e1.printStackTrace();  }  }  } |

5.1) 게시물 삭제 화면





|  |
| --- |
| 게시물 삭제 코드 |
| **public** **int** delPost(Post post) **throws** SQLException {  String sql="delete from POST where 게시물번호=?";  **try** {  PreparedStatement pstmt = con.prepareStatement(sql);  pstmt.setInt(1, post.get\_postnum());    **int** num=pstmt.executeUpdate();    pstmt.close();  **return** num;  }  **catch**(SQLException e) {  e.printStackTrace();  **return** -1;  }  **finally** {  con.close();  }  } |

|  |
| --- |
| 게시물 삭제 클릭 코드 |
| **if**(e.getSource()==del\_btn) {  // 사용자와 작성자가 같으면  **if**(mm.getID().equals(pp.get\_writer())) {  **int** result = JOptionPane.*showConfirmDialog*(**null**, "글을 삭제하시겠습니까?", "게시물 삭제", JOptionPane.***YES\_NO\_OPTION***);  **if** (result == JOptionPane.***CLOSED\_OPTION***) {  // 확인, 취소가 아닌 X로 닫은 경우 (계속 작성)  } **else** **if** (result == JOptionPane.***YES\_OPTION***) {  // 확인을 누른경우(삭제)  BoardSystem BS=**new** BoardSystem(mm);  **try** {  **int** num=BS.delPost(pp);  **if**(num>0) {  JOptionPane.*showMessageDialog*(**null**, "글이 삭제되었습니다.", "게시물 삭제", JOptionPane.***INFORMATION\_MESSAGE***);  BoardUI BUI=**new** BoardUI();  dispose();  }  } **catch** (SQLException e1) {  e1.printStackTrace();  }  } **else** { // 취소를 누른 경우  }  }  // 작성자와 사용자가 같지 않으면  **else** JOptionPane.*showMessageDialog*(**null**, "작성자가 아닙니다.", "error", JOptionPane.***INFORMATION\_MESSAGE***);  } |

5.2) 게시물 검색 화면

테이블이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| 게시물 검색 코드 |
| **public** Post[] searchPost(String str, String str2) **throws** SQLException {  String query = "select \* from POST where "+str+" like '"+str2+"' order by 작성일시 desc";  **try** {  Statement stmt = con.createStatement(ResultSet.***TYPE\_SCROLL\_INSENSITIVE***, ResultSet.***CONCUR\_UPDATABLE***);    ResultSet rs = stmt.executeQuery(query);    rs.last();  **int** rnum=rs.getRow();  rs.beforeFirst();    pp=**new** Post[rnum];  **int** i=0;    **while**(rs.next()) {  String num=rs.getString(1);  String title=rs.getString(2);  String contents=rs.getString(3);  String writer=rs.getString(4);  String writetime=rs.getString(5);    pp[i]=**new** Post(Integer.*parseInt*(num), title, contents, writer, writetime);  i++;  }  stmt.close();  rs.close();  }  **catch** (SQLException e) {  e.printStackTrace();  }  **finally** {  con.close();  }    **return** pp;  }  **public** Postlist updatePostlist() **throws** SQLException {  Postlist pl=**new** Postlist();    **return** pl;  } |

|  |
| --- |
| 게시물 검색 클릭 코드 |
| **if**(e.getSource()==search\_btn) {  //control 클래스 생성  BoardSystem BS=**new** BoardSystem(mm);    str=search\_CB.getSelectedItem().toString();  String str2="%"+search\_TF.getText()+"%";    Postlist p;  Post[] pll = **null**;  **try** {  pll = BS.searchPost(str, str2); //control 클래스 searchPost 호출  } **catch** (SQLException e2) {  // **TODO** Auto-generated catch block  e2.printStackTrace();  }    String header[]= {"번호", "제목", "내용", "작성자","작성일시"};  String[][] conts;  conts=**new** String[pll.length][header.length];    **for**(**int** i=0; i<pll.length; i++) {  conts[i][0]=Integer.*toString*(pll[i].get\_postnum());  conts[i][1]=pll[i].get\_title();  conts[i][2]=pll[i].get\_contents();  conts[i][3]=pll[i].get\_writer();  conts[i][4]=pll[i].get\_writetime();  }  DefaultTableModel model=**new** DefaultTableModel(conts, header) {  **public** **boolean** isCellEditable(**int** i, **int** c){ **return** **false**; }  };    postlisttable=**new** JTable(model);  // 컬럼 사이즈 지정  postlisttable.getColumnModel().getColumn(0).setPreferredWidth(40);  postlisttable.getColumnModel().getColumn(1).setPreferredWidth(100);  postlisttable.getColumnModel().getColumn(2).setPreferredWidth(250);  postlisttable.getColumnModel().getColumn(3).setPreferredWidth(70);  postlisttable.getColumnModel().getColumn(4).setPreferredWidth(120);  pane.setViewportView(postlisttable);  postlisttable.addMouseListener(**new** MouseAction());  } |

1. **테스팅**
   1. 블랙박스 테스팅

게시물 작성

* 게시물 작성을 작성할 경우 제목과 내용을 입력하고 둘 모두 입력하였다면 정상적으로 게시물이 작성된다.
* 제목을 입력하지 않은 경우 제목 입력 경고창을 보여준다.
* 내용을 입력하지 않은 경우 내용 입력 경고창을 보여준다.
* 제목, 내용 모두 입력하지 않은 경우 제목 입력 경고창을 보여준다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 테스트 케이스 | | 1 | 2 | 3 | 4 |
| 입력값 | 제목 입력 | T | T | F | F |
| 내용 입력 | T | F | T | F |
| 예상  출력값 | 작성된 게시물 출력 | T | F | F | F |
| 제목 입력 경고창 | F | F | T | T |
| 내용 입력 경고창 | F | T | F | F |

* 1. 화이트박스 테스팅

시작

**N**

g

**N**

d

**Y**

**Y**

h

f

e

c

a

b

음식 출력

종료

subcategory = 소분류

maincategory = 대분류

If (소분류 선택)

If (대분류 선택)

음식 추천

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 테스트 케이스 | | |
| 입력값 | 경로 | 출력값 |
| 1 | X | a-d-h | 전체 음식목록 중 출력 |
| 2 | 일식 | a-b-c-g-h | 일식 전체 목록 중 출력 |
| 3 | 일식, 튀김 | a-b-c-e-f-h | 일식-튀김의 목록 중 출력 |
| 4 | 한식, 밥 | a-b-c-e-f-h | 한식-밥의 목록 중 출력 |

1. **소감문**

조영문 - 일단, 우리 모든 팀원들이 부족한 팀장을 만나 고생한 거 같고, 또 다들 나름대로 열심히 해줘서 고맙습니다. 일단 전체적인 계획과 일정 및 테스크 분배, 음식랜덤추천기능 등을 맡아서 진행했으며, 모든 보고서와 피피티 등의 모든 문서 작업에 참여하며, 검토하고 수정하고, 제출하였다. 먼저, 우리 모든 주요기능은 4가지이며, 4가지를 각자 하나씩 맡아 진행하였습니다. 그 주요 기능 중 하나인 음식랜덤추천기능을 작성하였습니다. 또한 그것과 관련된 자바 GUI도 작성하였습니다. 또한, 계획서, 클래스 다이어그램, 순차다이어그램, 시나리오, 구현, 설계, 모든 부분에서 저의 맡은바를 다했습니다. 맡은 바라고 함은, 각자의 맡은 계획서, 다이어그램 등등을 다 진행했으며, 또 서로 모르는 것이나 도움이 필요한 부분은 나서서 도와주고 해결했습니다. 그리고 모든 보고서와 피피티의 검토를 맡았습니다. 모든 프로젝트에서 저의 손길이 닿지 않은 곳은 없습니다.

하지만, 그것보다 큰 저의 임무는 기간을 설정하여 모든 팀원들이 팀에 참여하여 자신의 일을 하고 또 다른 팀원의 테스크에도 조금씩은 참여하여 다른 테스크를 이해하고 팀위크를 만들어내 데드라인 맞추는 일, 그리고 팀워크의 중요성을 꺠워 다 같이 참여하는 그럼 팀프로젝트를 이끌어 냈다고 생각합니다. 소프트웨어공학 프로젝트 이외에도 다양한 프로젝트를 이번 학기에 팀장으로서 맡아서 진행했습니다. 그 중에서도 비교할 수 없게 우리 조의 팀원들이 가장 잘 따라줬으며, 가장 잘 진행됐다고 생각합니다. 결국 팀프로젝트의 목적과 의미는 다같이 하나의 목적을 가지고 앞으로 나아가는거라고 생각합니다. 그 부분에서는 결과가 어떻든 우리 조가 최고였다고 생각합니다. 그리고 저 또한 팀장으로 나쁘지 않았던 거 같습니다. 모든 팀원들이 정말 고생했지만, 그 중에서도 제일 고생한 유하영학생에서 많은 감사를 표합니다.

프로젝트 계획서 - 보고서 작성

요구사항 명세서 – 음식 랜덤 추천 기능 시나리오 작성 / 보고서 내용 작성

설계서 – 클래스 다이어그램 작성, 음식 추천 시퀀스 다이어그램 작성.

최종프로그램 작성 – 음식 추천 화면 담당 및 기능 담당, 전체적인 틀을 수정 및 검토, 프로젝트 전반에 걸친 검토 및 점검

유하영 – 프로젝트에서 핵심 인물로 전체적인 자바 GUI와 기능을 담당하고 취합

유한규 – 다양한 자료수집과 많은 양의 보고서 작성 및 게시판 작성 기능 담당

이상미 – 피피티 작업과 음식 찾기 기능 담당

유한규 – 우선 1년을 휴학하고 복학해서 아는 사람들이 없었는데 이번에 조원으로 참여하게 되었습니다. 첫 만남부터 친근하게 대해준 팀장 영문이에게 고맙다는 말을 하고 싶습니다. 처음보는 친구들이고 적응하기 어려울 수 있었는데 처음부터 분위기를 잘 만들어준 덕분에 잘 적응해서 프로젝트를 마무리할 수 있었던 것 같습니다.

처음 주제를 정하고 음식 추천이라는 단순한 프로젝트로 시작했었지만 이렇게 자세히 설계하면서 진행한게 이번이 처음이라 보고서 작성에 많은 어려움을 격었었는데 조원들과 힘을 합해서 하나하나 해결하다보니 벌써 프로젝트 종료가 눈 앞에 와있습니다.

주로 조원들이 보고서를 작성해주면 같이 작성하고 취합하는 역할을 하였고, 메인 프로젝트에서 메뉴 추천, 검색, 게시판 기능 중 게시판 기능을 하영이와 함께 하였습니다. 데이터베이스 수업을 듣지 않아 어려움을 겪는 저를 위해 많은 부분을 희생해주면서 고생해주었고, 저는 옆에서 코드를 배우며 같이 알아가고 UI부분을 담당해서 했었습니다.

다양한 과제들 중 조원당 1개 혹은 2개씩 제출하는 과제에서 서로 미루려고 하지 않고 새벽까지 다들 고심하여 의견을 나누고 제출하는데 문제가 생기지 않을만큼 진행했던게 가장 인상 깊었던 것 같습니다.

조원끼리의 합도 잘 맞고 다른 과제들과 겹치지 않게 미리미리 끝내자는 의견이 있었는데 그 의견을 따라서 뒤쳐지지 않고 다같이 고생해준 조원들에게 고마움을 느낍니다.

유하영

프로젝트 계획서 - 보고서 작성(기본 내용 및 기능 점수 등)

요구사항 명세서 – 게시판 수정, 삭제, 즐겨찾기 추가 시나리오 작성 / 보고서 전반적인 내용 작성

설계서 – 클래스 다이어그램 작성, 게시판 시퀀스 다이어그램 작성.

최종프로그램 작성 – 프로그램 GUI, DB 연동, 게시판 메인화면 담당, 각 클래스 내용 작성 및 수정, 테스트하여 프로그램 완성

느낀점 : SW공학 과목에서 배우는 내용을 실제로 적용해보려니 예상외로 어려웠습니다. 특히 시퀀스 다이어그램, 테스팅 부분에 대한 내용이 깊게 이해가 되지 않았습니다. 그러나 프로젝트를 진행하며 계획하고 설계하여 프로그램을 작성하니 평소보다 훨씬 수월하고 체계적이게 프로그램이 작성되는 것을 느꼈습니다. 앞으로도 이러한 습관을 들여 체계적인 개발을 하도록 노력할 것입니다.

이상미 - 다른 수업들의 팀플도 동시에 진행되는 것치고 프로젝트 일정이 매우 여유롭게 짜였던 것 같습니다. 미리미리 일정을 짜고 공지하여 프로젝트 진행해준 저희 팀장분과 팀원분들께 감사드립니다. 클래스 다이어그램이나 시나리오 작성 등의 분야에서 담당했던 기능이 각각 달랐기 때문에 정확히 어느 부분을 진행했다! 라고 하기엔 조금 난감한 감이 있는 것 같습니다. 당장 생각나는 걸로는 시나리오 작성에서는 즐겨찾기 삭제와 음식점정보에 관한 것을, 클래스 다이어그램에서는 관리자 부문을, 팀원 모두가 같이 생각해 뽑아낸 UI의 그림을, GUI 부문에서는 음식검색기능을 진행한 것 정도인 것 같습니다. 첫 발표때의 ppt와 마지막 발표의 ppt 또한 본인이 작성했고 하는 중인데, 팀원 모두 여쭤본 것들에 대해 피드백을 잘 해주고 괜찮다고 격려도 해주어 잘 진행됐다고 생각합니다.

중간중간 개인 사정으로 부득이하게 참여하지 못하는 일이 있었는데, 그럼에도 질책하지 않고 이해해주려 노력하며 같이 진행해주신 모든 팀원분들께 감사의 말 드리고 싶습니다. 아쉬운 점이 있다면 Java는 단 한학기만 배웠던 터라 아는 점이 잘 없어 짐을 조금 지워드린게 아니었을까 싶은 부분 같습니다. 나름 인터넷에 검색하고 과거 강의들을 돌아보며 노력했으나 DB 연동 부분에서 영 감을 잡지 못해 죄송하고 감사했습니다. 그런 느낌으로 개인적으로 아쉬운 점도 있었지만 팀적인 부분에서 보면 정말 좋은 팀이었다고 생각합니다. 마지막 남은 작업도 열심히 해서 만족스러운 발표 하실 수 있게 만들어드리고 싶습니다. 지금까지 프로젝트 진행 수고하셨습니다!