[소프트웨어 공학 최종 보고서 양식]

|  |
| --- |
| **소프트웨어공학 최종 보고서** |
| **[프로젝트명: 과외서비스 ]** |

**2018. 12**

|  |  |
| --- | --- |
| **성 명** | **학 번** |
| 최상호 | 201464039 |
| 김성건 | 201495007 |
| 허준석 | 201464042 |
| 박진오 | 201464018 |

**목 차**

1. 프로젝트 계획서 ‧‧‧
2. 문제 설명서 ‧‧‧
3. 유스케이스 다이어그램 ‧‧‧
4. 유스케이스 식별자 목록 ‧‧‧
5. 유스케이스 시나리오 ‧‧‧
6. 클래스 다이어그램 ‧‧‧
7. 시스템 구조도 ‧‧‧
8. ER 다이어그램 ‧‧‧
9. 테이블 기술서 ‧‧‧
10. 시퀀스 다이어그램 ‧‧‧
11. 커뮤니케이션 다이어그램 ‧‧‧
12. 코드 리뷰 체크리스트 ‧‧‧
13. 소스 프로그램의 헤더 주석 ‧‧‧

[붙임]

1) 클래스 자바 코드 ‧‧‧

2) SQL DDL ‧‧‧

3) 헤더 주석을 포함한 소스 프로그램 ‧‧‧

|  |  |
| --- | --- |
| **프로젝트 계획서** | |
| **프로젝트명** | [과외 서비스] |
| **프로젝트 목적** [개요와 해결할 문제를 기술한다] | |
| 회원들에게 국, 영, 수 과목 뿐만 아니라 사,과탐 과목 , 예체능 과목 괴외를 선택할 수 있게 합니다.  과목을 가르쳐주는 선생님들에게 좀 더 간편하고 쉽게 접근할 수 있도록 도와주는 서비스를 제공 합니다.  학생들과 과외선생님들에게 연결을 쉽게 해줄 수 있도록 하는 서비스를 개발하려고 합니다.  전국 각지에 있는 수험생들에게 지속적인 발전과 효울적이고 능동적인 과외 시스템을 도입하여 자기주도학습, 맞춤과외에 있어 좀 더 좋은 품질의 과외를 소개 시켜 줌으로써 성적향상의 변화를 느껴볼 수 있도록 공부에 대한 의지를 갖게 해줍니다. | |
| **프로젝트 범위** [개발할 기능을 중심으로 기술한다] | |
| 회원가입 서비스  학생 – 등록,회원정보수정,탈퇴  교사 - 등록,회원정보수정,탈퇴  과목관리  조회 – 시간표 (학생 , 교사) , 지역별 최다별점 교사 , 성적 그래프  등록 – 학생(관심과목) 교사(등록과목)  검색 – 학생(교사, 관심과목강의) 교사(학생)  쪽지시험 서비스  1대1 채팅, 상담 서비스  후기, 별점 서비스 | |

|  |  |  |
| --- | --- | --- |
| **팀원 이름 / 학번** | **역할** | **구분** |
| 최상호/201464039 | 설계 / 프로그래밍 | 팀장 |
| 허준석/201464042 | 디자인 / 프로그래밍 | 팀원 |
| 박진오/201464018 | 요구 분석 / 프로그래밍 | 팀원 |
| 김성건/201495007 | 테스트 / 프로그래밍 | 팀원 |

|  |
| --- |
| **과외서비스 문제 설명서** |

과외서비스 시스템은 과외의 등록이나 신청을 온라인 상에서 간편하게 하기 위한 시스템이다.

과외관리 시스템 관리자는 학생이나, 선생님이 신청 또는 등록해 놓은 과외를 관리하고,

학생은 자신이 과외를 신청하거나, 선생님이 올려놓은 과외를 통해 과외를 신청할 수 있고, 선생님은 자신이 할 과외를 등록하거나, 학생이 신청해 놓은 과외를 통해 과외를 할 수 있다.

과외서비스를 이용하는 사용자는 일반고객(비회원), 회원(선생, 학생), 관리자로 나뉜다.

비회원은 과외서비스 기능을 이용하기 위해서 회원으로 가입할 수 있다.

회원은 다시 학생 회원과 선생 회원으로 나뉜다.

학생 회원은 가입 시에 자신의 인적사항을 적고 학벌, 경제여력, 관심과목 등을 선택적으로 적는다.

선생 회원은 가입 시에 자신의 인적사항과 학벌, 계좌 정보를 필수적으로 적는다.

과외를 등록하는 것은 선생 회원만 할 수 있고, 과외를 신청하는 것은 학생 회원만 가능하다.

관리자는 과외 관리(등록, 수정, 삭제)나 회원 관리(조회, 삭제) 등과 같은 활동을 수행한다.

과외에 대한 조회 및 검색은 과외서비스에 가입한 회원뿐만 아니라 가입하지 않은 비회원 및 관리자도 수행할 수 있다.

과외 조회는 특정 카테고리의 선택 과정을 통해 이루어지며, 선택한 정렬방법대로 보여준다.

과외 조회 시 선택할 수 있는 카테고리에는 지역, 학벌, 요금, 평점, 별점 등이 있다.

선생 회원이 과외를 등록할 때는 학생 회원과 접촉할 방법을 선택할 수 있다.

선생과 회원이 서로의 시간대를 맞추기 위해서 스케줄을 조회, 수정 할 수 있게 한다.

회원가입시 기재한 연락처를 공개할 수도 있지만, 연락처를 공개하고 싶지 않다면 사이트 내의 메시지 기능으로만 학생 회원과 접촉할 수도 있다.

선생이 쪽지시험 서비스를 이용하여 온라인으로 학생에게 서비스를 이용할 수 있게 할 수 있다.

학생이 쪽지시험의 성적을 바탕으로 성적그래프를 조회를 할 수 있다.

학생이 과외를 수강완료를 했을 때 과외 선생에 대해서 후기를 남기거나 별점을 줄 수 있다.

학생 회원이 과외를 신청할 때는 선생 회원이 제공하는 연락방법을 통해서만 신청할 수 있다.

과외를 결재한 학생에게만 선생의 전화번호와 메세지기능을 활성화 시킴으로써 어느 정도의 보안(장난전화, 거짓말에 대한)을 챙기도록 관리자가 관리한다.

관리자는 카테고리의 분류 체계를 관리하거나 양식에 맞지 않게 등록된 과외를 수정하거나 삭제한다. 사이트 규칙을 위반하는 회원에게는 경고 메시지를 주거나 탈퇴 시킨다.

과외 신청 시 결제 수단은 무통장입금 , 온라인 입금 , 신용카드 결제, PAYPAL ,포인트로 구분한다.

온라인 입금은 뱅킹 시스템을 통해 입금 여부를 확인한 후 결제 처리하며, 신용카드 결제는 신용카드 인증회사로의 결제승인 서비스를 통해 즉시 처리가 가능하도록 한다.

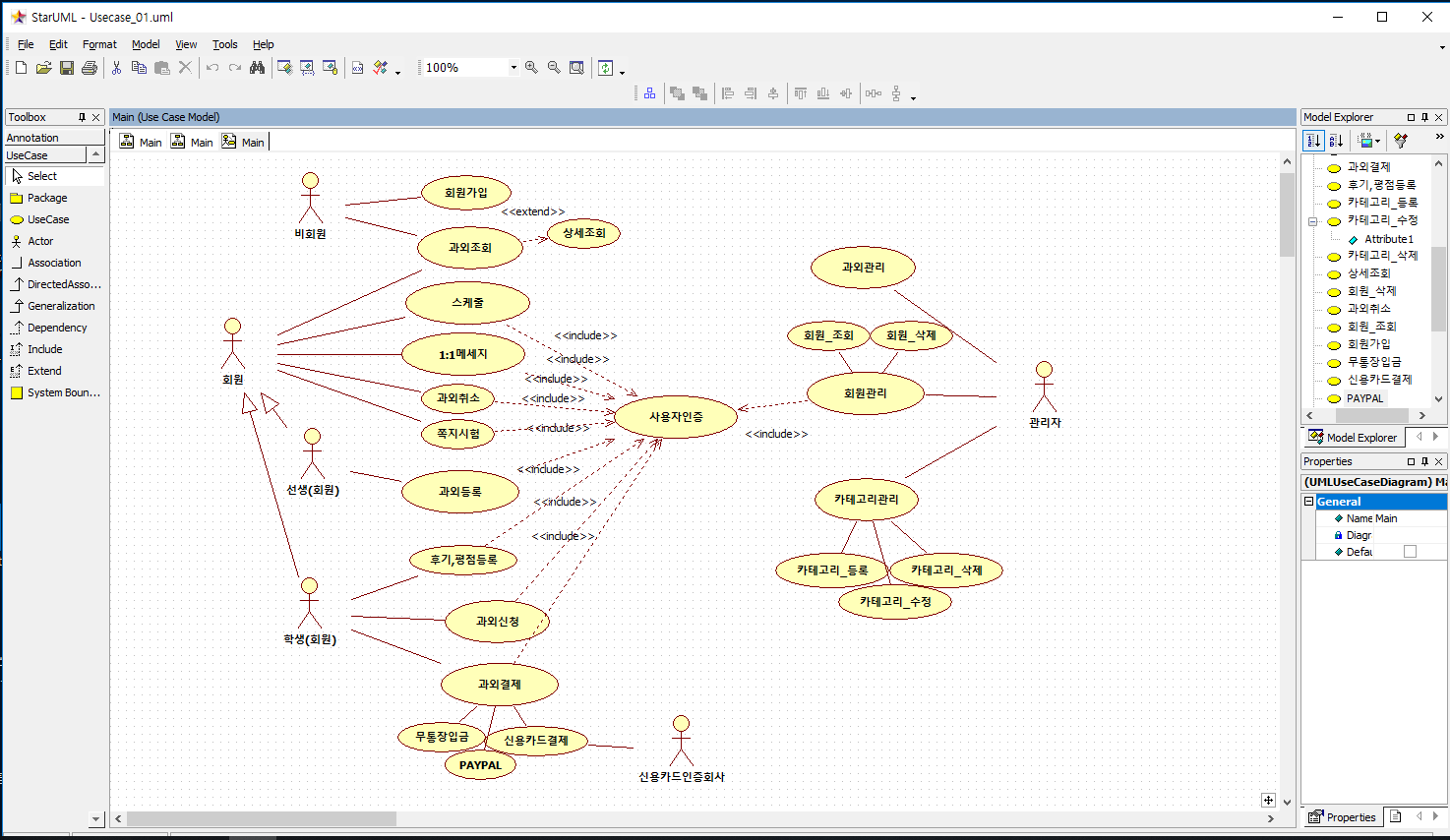
포인트로도 과외결제가 가능하도록 한다

결제를 하면 사이트 내에서만 이용할 수 있는 환전 가능한 포인트가 지급된다.(단, 포인트로 구매한 과외 일 경우 포인트가 지급되지 않는다.)

학생이 포인트를 충전하고 과외 서비스를 신청하면 포인트는 관리자 쪽에서 중개하게 된다.

선생 회원은 학생 회원과 직접 만난 뒤, 학생 회원이 최종 확인을 해야만 포인트가 지급된다.

|  |
| --- |
| **유스케이스 다이어그램** |



|  |
| --- |
| **유스케이스 식별자 목록** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 식별자 | 행위자 | 설명 | 식별자 | 행위자 | 설명 |
| UC\_N01 | 비회원 | 회원가입 | UC\_S01 | 학생 | 과외신청 |
| UC\_N02 | \* | 과외조회 | UC\_S02 | 학생 | 평점등록 |
| UC\_N03 | \* | 상세조회 | UC\_S03 | 학생 | 신용카드결제 |
| UC\_M01 | 회원 | 스케줄 | UC\_S04 | 학생 | 무통장입금 |
| UC\_M02 | 회원 | 1:1메세지 | UC\_S05 | 학생 | PAYPAL |
| UC\_M03 | 회원 | 쪽지시험 | UC\_A01 | 관리자,회원 | 사용자인증 |
| UC\_M04 | 회원 | 과외취소 | UC\_A02 | 관리자 | 카테고리\_동록 |
| UC\_T01 | 선생 | 과외등록 | UC\_A03 | 관리자 | 카테고리\_수정 |
|  |  |  | UC\_A04 | 관리자 | 카테고리\_삭제 |
|  |  |  | UC\_A05 | 관리자 | 회원\_조회 |
|  |  |  | UC\_A06 | 관리자 | 회원\_삭제 |

|  |
| --- |
| **유스케이스 시나리오** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC\_N01 회원가입 서비스 | |
| **개요** | 비회원인 사람에게 좀 더 다양한 서비스를 제공하기 위해서 회원 서비스를 통해 제공한다 | |
| **주 행위자** | 비회원 | |
| **부 행위자** | 해당상황 없음 | |
| **사전 조건** | 비회원인 경우 | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 회원가입 서비스를 요청한다.   3. 회원정보, 항목(아이디, 닉네임, 비밀번호,비밀번호 재입력, 연락처, 가입유형(선생,학생), 사는 곳)을 입력한다. | | 1. 회원 가입 서비스를 위한 페이지를 열어주고 입력항목을 보여준다.   4. 회원정보를 저장, 등록을 완료한다 |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| E1. 로그인한 상태로 회원가입을 요청했을 경우.  E3. 회원정보 입력항목 중 입력하지 않은 항목이 있을 경우  E3. 아이디가 중복일 경우  E3. 비밀번호와, 비밀번호 재입력이 다를 경우  E3. 아이디가 한글일 경우  E3. 아이디가 숫자로 시작할 경우  E3. 아이디가 6자리 미만일 경우  E3. 비밀번호가 6자리 미만일 경우  E3. 연락처 형식에 맞지 않게 입력했을 경우 | | 메인 화면으로 이동한다.  해당항목 입력요청 메시지를 보여준다.”  “아이디가 중복되었다는 메시지를 보여준다.”  “비밀번호 다름 메시지와 재입력 요청 메시지를 보여준다.  “영어, 숫자로만 입력요청 메시지를 보여준다.”  “영어 소문자로 입력요청 메시지를 보여준다.”  “최소 6자이상으로 입력요청 메시지를 보여준다.”  “최소 6자이상으로 입력요청 메시지를 보여준다.”  “연락처 형식에 맞게 입력요청 메시지를 보여준다.” |
| **사후 조건** | | **해당 사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC\_N02과외 조회 서비스 | |
| **개요** | 회원 , 비회원이 과외 시스템을 조회하기 위해 사용하는 유스 케이스이다. | |
| **주 행위자** | 회원 , 비회원 | |
| **부 행위자** | x | |
| **사전 조건** | x | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 2. 지역, 학벌, 요금, 평점 등의 조건을 선택하여 조회 서비스를 요청한다. | | 1. 과외 조회서비스창을 열어준다.  3. 조건에 맞는 등록된 과외 정보를 보여준다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| E2. 조건을 선택하지 않고 조회 서비스를 요청했을 경우 | | 경고 메시지 “조건을 선택 하지 않으셨습니다.” 라고 뜬다. |
| **사후 조건** | | **해당 사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC\_N03과외 상세 조회 서비스 | |
| **개요** | 회원 , 비회원이 과외 시스템을 조회하기 위해 사용하는 유스 케이스이다. | |
| **주 행위자** | 회원 , 비회원 | |
| **부 행위자** | x | |
| **사전 조건** | x | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 2. 지역, 학벌, 요금, 평점 등의 조건을 선택하여 조회 서비스를 요청한다. | | 1. 과외 조회서비스창을 열어준다.  3. 조건에 맞는 등록된 과외 정보를 보여준다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| E2. 조건을 선택하지 않고 조회 서비스를 요청했을 경우 | | 경고 메시지 “조건을 선택 하지 않으셨습니다.” 라고 뜬다. |
| **사후 조건** | | **해당 사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC\_M01스케줄 서비스 | |
| **개요** | 회원인 사람에게 스케줄 서비스 제공한다 | |
| **주 행위자** | 회원 | |
| **부 행위자** | 해당상황 없음 | |
| **사전 조건** | 회원인 경우 | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 스케줄 서비스를 요청한다 2. 스케줄을 적고 저장한다. | | 1. 해당하는 회원의 스케줄 서비스를 열어준다 2. 스케줄을 저장한다.. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| E3. 이미 있는 스케줄에 또 잡을 경우. | | 겹치는 스케줄이 있을 경우 “스케줄이 겹칩니다” 라고 메시지를 띄웁니다. |
| **사후 조건** | | **해당 사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC\_M02 1:1 메시지 서비스 | |
| **개요** | 회원에게 다른 회원에게 1:1 메시지를 보낼 수 있는 기능을 제공한다. | |
| **주 행위자** | 회원 | |
| **부 행위자** | 해당상황 없음 | |
| **사전 조건** | 회원인 경우 | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 메시지 서비스를 요청한다. | | 1. 메시지 창을 보여준다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
|  | |  |
| **사후 조건** | | **해당 사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC\_M03 쪽지시험 서비스(선생용) | |
| **개요** | 선생 회원이 학생 회원에게 쪽지 시험을 치게 하여 해당 과목의 점수 향상을 시킨다. | |
| **주 행위자** | 선생 회원 | |
| **부 행위자** | 해당상황 없음 | |
| **사전 조건** | 회원인 경우 | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 쪽지 시험 서비스를 요청한다.   3. 쪽지시험을 등록한다.  5. 담당 학생의 점수를 확인한다. | | 2. 해당 과목의 쪽지 시험을 등록할 수 있는 페이지를 열어준다.  4. 학생 회원이 쪽지시험을 치고 난 후의 점수를 담당 선생회원에게 보여준다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| E3. 쪽지 시험을 등록 하지 않았을 경우 | | “쪽지 시험을 등록하지 않으셨습니다” 라고 메시지를 보여준다. |
| **사후 조건** | | **해당 사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC\_M03 쪽지시험 서비스(학생용) | |
| **개요** | 선생 회원이 학생 회원에게 쪽지 시험을 치게 하여 해당 과목의 점수 향상을 시킨다. | |
| **주 행위자** | 학생 회원 | |
| **부 행위자** | 해당상황 없음 | |
| **사전 조건** | 회원인 경우 | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 쪽지 시험 서비스를 요청한다.   3. 시험을 친다 | | 1. 해당하는 학생 회원의 과목에 대한 쪽지 시험 서비스 페이지를 열어준다.   4. 학생 회원의 시험 점수를 저장한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| E3. 쪽지 시험의 답을 입력하지 않은 경우 | | “정답을 입력하지 않았습니다” 라고 메시지가 뜬다. |
| **사후 조건** | | **해당 사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC\_M04과외 취소 서비스(선생용) | |
| **개요** | 선생회원이 과외를 취소하기 위해 사용하는 유스 케이스이다. | |
| **주 행위자** | 선생회원 | |
| **부 행위자** | 해당상황 없음 | |
| **사전 조건** | 학생이 과외를 신청 했는데 취소할 경우  선생이 과외 할 수 있는 여건이 되지 않는 경우  (단, 최종 수락을 누르지 않았을 경우만 해당.) | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 2. 수강 취소 서비스를 신청한다,  4. 수강 취소시킬 학생을 클릭한다.  5. 수강 취소 버튼을 누른다. | | 1. 로그인 한 선생 회원인지 확인한다.   1. 수강신청한 학생들의 테이블을 보여준다.   6. 학생 포인트를 반환 시킨다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
|  | |  |
| **사후 조건** | | **해당 사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC\_M04과외 취소 서비스(학생용) | |
| **개요** | 학생회원이 과외를 취소하기 위해 사용하는 유스 케이스이다. | |
| **주 행위자** | 학생회원 | |
| **부 행위자** | x | |
| **사전 조건** | 학생이 과외를 신청 했는데 취소할 경우(단, 최종 수락을 누르지 않았을 경우만 해당.) | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 2, 수강 취소 서비스를 신청한다.  4. 수강 취소 버튼을 누른다. | | 1. 로그인한 학생 회원인지 확인한다.  3. 수강 신청한 과목을 보여준다  5. 포인트 증가된다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| E2. 포인트가 없을 경우 | | 포인트결제 페이지로 이동한다. |
| **사후 조건** | | **해당 사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC\_T01 과외 등록 서비스 | |
| **개요** | 선생 회원이 과외를 등록 할 때 쓰는 유스케이스이다. | |
| **주 행위자** | 선생 회원 | |
| **부 행위자** | 해당상황 없음 | |
| **사전 조건** | 선생회원인 경우 | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 과외 등록 서비스를 요청한다,  3. 과외 정보(과목, 가격)를 등록한다. | | 2. 해당 서비스의 페이지를 열어준다.  4. 등록한 과외 정보를 저장한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| E3. 과외정보를 등록하지 않았을 경우 | | “수강등록이 정상적으로 처리되지 못했습니다.” 라고 메시지가 뜬다. |
| **사후 조건** | | **해당 사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC\_S01 과외신청 서비스 | |
| **개요** | 학생회원이 과외를 신청할 때 사용되는 유스케이스이다. | |
| **주 행위자** | 학생회원 | |
| **부 행위자** | 해당상황 없음 | |
| **사전 조건** | 학생회원인 경우 | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 과외신청 서비스를 요청한다.   3. 원하는 과외를 선택한다.  5. 과외를 신청한다. | | 1. 과외신청 페이지를 열어준다.   4. 과외를 선택하면 과외 상세정보(선생정보, 과목, 날짜, 가격)를 보여준다.  6. 신청한 과외를 저장한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| A1. 서비스 신청을 하다가 취소를 할 경우 | | “수강신청이 취소 되었습니다.” 메시지를 보여준 후 이전 페이지를 보여준다. |
| **사후 조건** | | **해당 사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC\_S02 평점등록 서비스 | |
| **개요** | 학생회원이 과외를 수강한 후 평점을 등록할 때 쓰이는 유스케이스이다. | |
| **주 행위자** | 학생회원 | |
| **부 행위자** | 해당상황 없음 | |
| **사전 조건** | 학생회원인 경우 | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. 평점등록 서비스를 요청한다 2. 평점등록을 하기 위한 선생을 클릭한다..   5. 평점등록을 한다 | | 2. 평점등록 페이지를 열어준다.   1. 평점등록을 할 수 있는 페이지를 열어준다.   6. 학생회원이 등록한 평점을 저장한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| A5. 평점등록을 하다가 취소를 할 경우 | | “평점등록이 취소 되었습니다” 라고 메시지가 뜬다. |
| **사후 조건** | | **해당 사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC-S03 신용카드 결제 유스케이스 | |
| **개요** | 학생 회원이 신용카드 결제를 진행하는 유스케이스다. | |
| **주 행위자** | 학생 회원 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 학생 회원이 로그인한 상태여야 한다. | |
| **정상 흐름** | | |
| **행위자** | | **시스템** |
| 2. 결제 방식으로 신용카드 결제를 선택한다.  4. 신용카드 방식의 결제를 요청한다.  6. 카드 정보를 입력한다. | | 1. 결제 방식을 선택할 수 있는 화면이 나타난다.  3. 카드 정보를 선택할 수 있는 창을 보여준다.  5. 결제한 금액 만큼 사이트 내에서 사용할 수 있는 포인트를 충전해준다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| A4. 회원이 결제하기 전에 취소를 요청한 경우 | | 결제를 취소하고 결제 요청 전의 화면을 출력한다. |
| **예외 흐름** | | |
| **행위자** | | **시스템** |
| E2. 필수 입력 정보를 입력하지 않을 경우  E2. 잘못된 카드 정보를 입력할 경우 | | 오류 메시지를 띄우고, 미입력된 항목에 포커스를 잡아준다.  오류 메시지를 띄우고, 잘못 입력한 항목에 포커스를 잡아준다. |
| **사후 조건** | | **해당사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC-S04 무통장 입금 유스케이스 | |
| **개요** | 학생 회원이 무통장 입금을 통해 결제를 진행하는 유스케이스다. | |
| **주 행위자** | 학생 회원 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 학생 회원이 로그인한 상태여야 한다. | |
| **정상 흐름** | | |
| **행위자** | | **시스템** |
| 2. 결제 방식으로 무통장 입금을 선택한다.  4. 결제할 금액, 입금할 은행, 입금자 성명을 입력한다.  6. 나타난 계좌번호로 입금한다. | | 1. 결제 방식을 선택할 수 있는 화면이 나타난다.  3. 결제할 금액과 입금할 은행, 입금자 성명을 입력할 수 있는 창을 보여준다.  5. 입금해야할 관리자의 계좌번호를 보여준다.  7. 결제한 금액 만큼 사이트 내에서 사용할 수 있는 포인트를 충전해준다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| A5. 회원이 입금하기 전에 결제를 취소하고 싶은 경우 | | 결제 조회 페이지에서 결제를 취소한다. |
| **예외 흐름** | | |
| **행위자** | | **시스템** |
| E6. 회원이 정해진 금액보다 적거나 많게 입금했을 경우 | | 회원 정보에 등록된 연락처로 정해진 금액보다 적거나 많이 입금했음을 알려주는 메시지를 보내준다. |
| **사후 조건** | | **해당사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC\_S05 PAYPAL 서비스 | |
| **개요** | 회원이 포인트를 충전하기 위해 PAYPAL 서비스를 이용한다. | |
| **주 행위자** | 회원 | |
| **부 행위자** | 해당상황 없음 | |
| **사전 조건** | 회원인 경우 | |
| **기본 흐름** | | |
| **행위자** | | **시스템** |
| 1. PAYPAL 서비스를 요청한다.  4. 기입란에 자신의 정보를 입력한다. | | 2. 해당 서비스의 페이지를 열어준다.  3, PAYPAL의 결제 사이트로 이동한다.  5. PAYPAL에 결제를 요청한다,  6. 포인트가 부여한다, |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
|  | |  |
| **사후 조건** | | **해당 사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC-A01 사용자 인증 유스케이스 | |
| **개요** | 관리자 및 회원이 사용자 인증을 수행하는 유스케이스다. | |
| **주 행위자** | 관리자, 학생 회원 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 없음 | |
| **정상 흐름** | | |
| **행위자** | | **시스템** |
| 1. 특정한 기능을 요청한다. | | 2. 로그인 세션이 만료됐는지 확인한다.  3. 기능을 요청한 사용자가 그에 맞는 권한을 가졌는지를 확인한다.  4. 요청한 기능을 수행한다. |
| **예외 흐름** | | |
| **행위자** | | **시스템** |
| E2. 로그인 세션이 만료됐을 경우  E3. 기능을 요청한 사용자가 그에 상응하는 권한을 가지지 않은 경우 | | 로그인을 할 수 있는 페이지로 이동시킨다.  예외 메시지를 보여주고 이전 페이지로 이동시킨다. |
| **사후 조건** | | **해당사항 없음** |

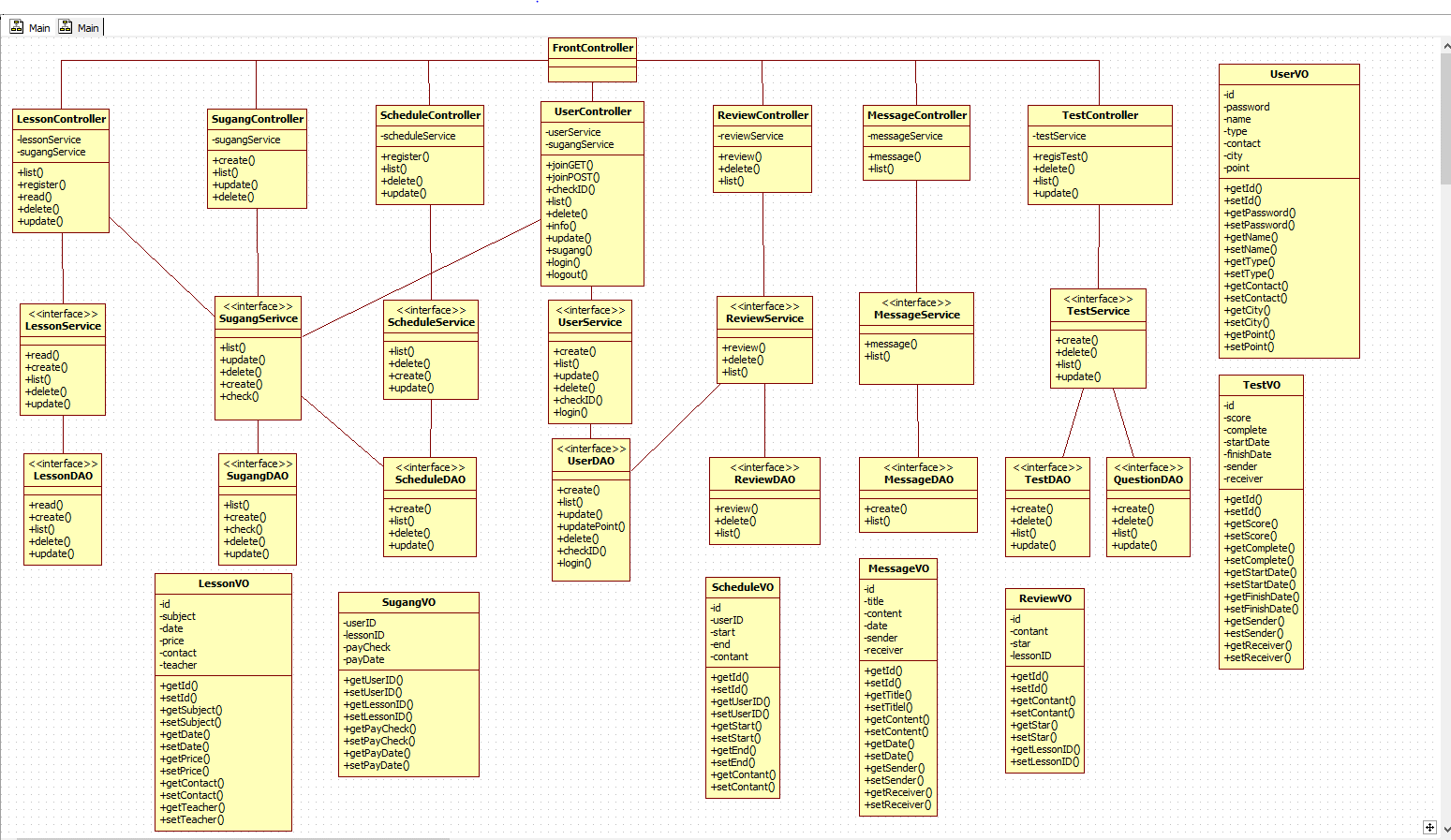
|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC-A02 카테고리 등록 유스케이스 | |
| **개요** | 관리자가 상세 조회를 하기 위한 카테고리를 등록하는 유스케이스다. | |
| **주 행위자** | 관리자 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 관리자가 관리자 계정으로 로그인한 상태여야 한다. | |
| **정상 흐름** | | |
| **행위자** | | **시스템** |
| 2. 최상위 카테고리의 이름을 정한 뒤 등록한다.  4. 하위 카테고리를 등록한다.  5. 원하는 만큼 4번을 반복한 뒤 등록한다. | | 1. 등록된 최상위 카테고리 목록을 보여준다.  3. 등록할 최상위 카테고리가 가질 하위 카테고리를 등록할 수 있는 입력 창을 보여준다.  6. 등록한 후의 최상위 카테고리 목록을 보여준다. |
| **예외 흐름** | | |
| **행위자** | | **시스템** |
| E2, E4. 카테고리 등록시 중복되는 이름을 사용하는 경우 | | 예외 메시지를 보여주고 등록을 취소한다. |
| **사후 조건** | | **해당사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC-A03 카테고리 수정 유스케이스 | |
| **개요** | 관리자가 등록된 카테고리를 수정하는 유스케이스다. | |
| **주 행위자** | 관리자 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 관리자가 관리자 계정으로 로그인한 상태여야 한다. | |
| **정상 흐름** | | |
| **행위자** | | **시스템** |
| 2. 수정할 카테고리를 선택한다.  4. 카테고리를 원하는 이름으로 수정한다. | | 1. 등록된 최상위 카테고리 목록과 하위 카테고리 목록을 보여준다.  3. 카테고리를 수정할 수 있는 창을 보여준다.  5. 수정된 카테고리 목록을 보여준다. |
| **예외 흐름** | | |
| **행위자** | | **시스템** |
| E2, E4. 카테고리 수정시 중복되는 이름을 사용하는 경우 | | 예외 메시지를 보여주고 수정을 취소한다. |
| **사후 조건** | | **해당사항 없음** |

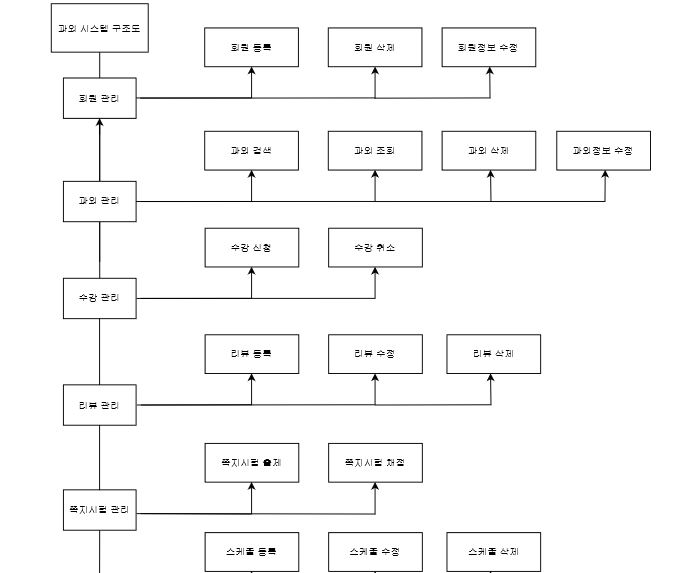
|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC-A04 카테고리 삭제 유스케이스 | |
| **개요** | 관리자가 등록된 카테고리를 삭제하는 유스케이스다. | |
| **주 행위자** | 관리자 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 관리자가 관리자 계정으로 로그인한 상태여야 한다. | |
| **정상 흐름** | | |
| **행위자** | | **시스템** |
| 2. 삭제할 카테고리를 선택한다.  4. 삭제를 결정한다. | | 1. 등록된 최상위 카테고리 목록과 하위 카테고리 목록을 보여준다.  3. 선택한 카테고리를 정말로 삭제할지 여부를 물어보는 메시지 창을 띄운다.  5. 선택한 카테고리를 삭제한다.  6. 삭제된 카테고리 목록을 보여준다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| A4. 삭제를 취소한다. | | 삭제를 취소하고, 등록된 카테고리 목록을 보여주는 페이지로 이동시킨다. |
| **사후 조건** | | **해당사항 없음** |

|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC-A05 회원 조회 유스케이스 | |
| **개요** | 관리자가 가입된 회원을 조회하는 유스케이스다. | |
| **주 행위자** | 관리자 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 관리자가 관리자 계정으로 로그인한 상태여야 한다. | |
| **정상 흐름** | | |
| **행위자** | | **시스템** |
| 2. 상세 정보를 조회할 회원을 선택한다.  4. 관리자가 원하는 경우 회원을 삭제 할 수도 있다. (UC-A06 회원 삭제 유스케이스) | | 1. 가입된 회원의 목록을 보여준다.  3. 선택한 회원의 상세 정보(아이디, 이름, 가입유형(학생,선생), 연락처, 주소, point)를 보여준다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| A3. 회원 상세조회 중에 ‘목록’ 기능을 요청하는 경우 | | 다시 가입된 회원 목록을 보여준다. |
| **사후 조건** | | **해당사항 없음** |

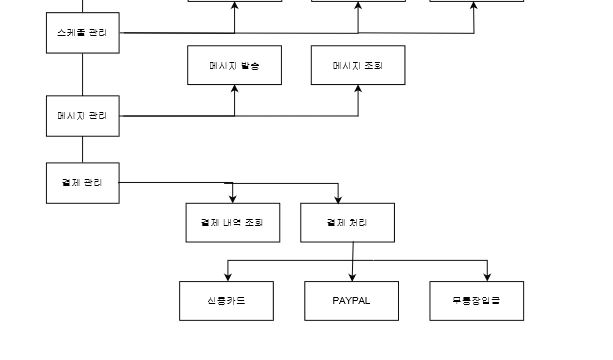
|  |  |  |
| --- | --- | --- |
| **유스케이스 식별자명** | UC-A06 회원 삭제 유스케이스 | |
| **개요** | 관리자가 가입된 삭제을 조회하는 유스케이스다. | |
| **주 행위자** | 관리자 | |
| **부 행위자** | 해당사항 없음 | |
| **사전 조건** | 관리자가 관리자 계정으로 로그인한 상태에서 회원 상세 조회를 한 상태여야 한다. (UC-A05 회원 조회 유스케이스) | |
| **정상 흐름** | | |
| **행위자** | | **시스템** |
| 1. 회원 삭제를 요청한다.  3. 삭제를 결정한다. | | 2. 선택한 회원을 정말로 삭제할지 물어보는 메시지 창을 띄운다.  5. 선택한 회원를 삭제한다. |
| **대안 흐름** | | |
| **행위자** | | **시스템** |
| A3. 삭제를 취소한다. | | 삭제를 취소하고, 회원 목록을 보여주는 페이지로 이동시킨다. |
| **사후 조건** | | **해당사항 없음** |



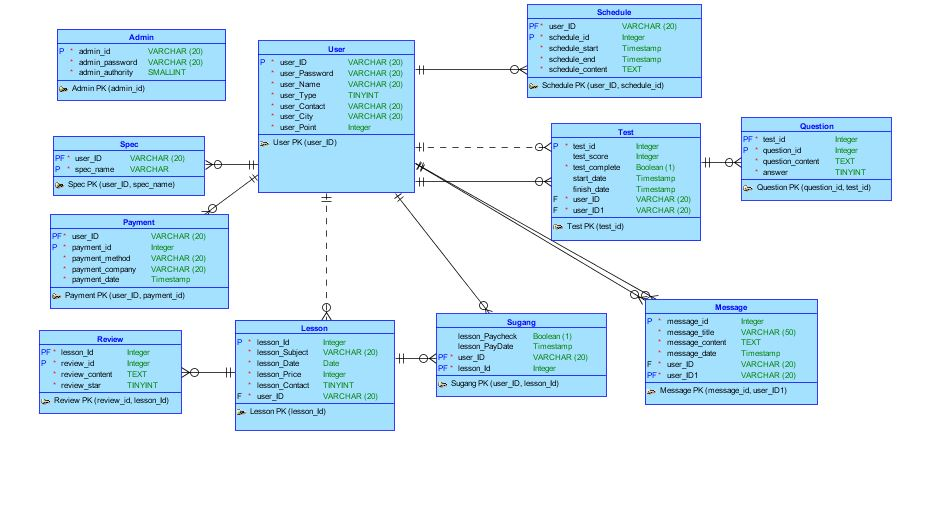
|  |
| --- |
| **클래스 다이어그램** |

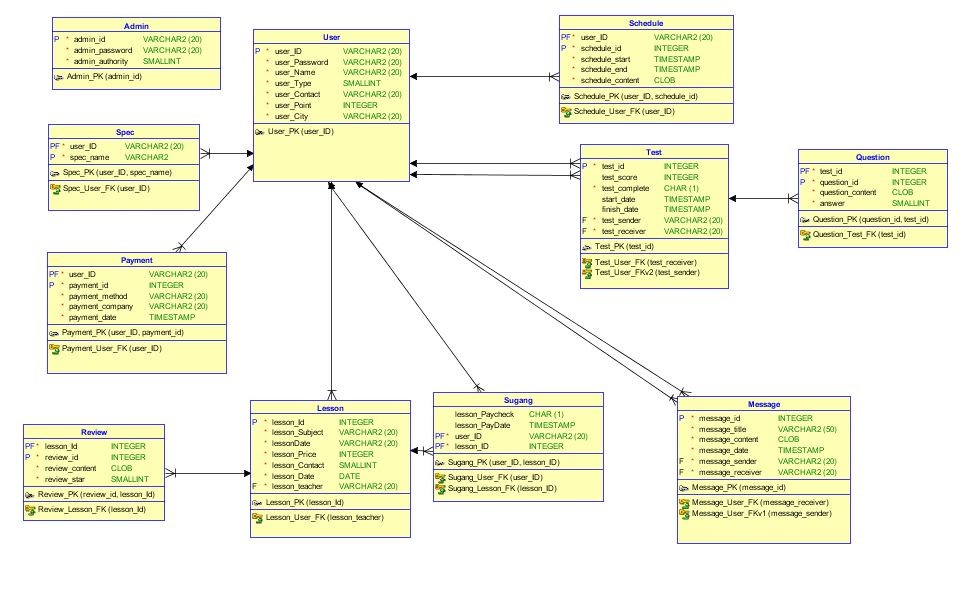


|  |
| --- |
| **시스템 구조도** |



|  |
| --- |
| **ER 다이어 그램** |





|  |
| --- |
| **데이터베이스 테이블 기술서** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table**  **Name** | | Admin | **Table 기술서** | | | 작성일 | 2018.11.12 |
| **System** | | 과외시스템 | 작성자 | 허준석 |
| **No.** | **Column name** | | **Data type** | **not null** | **Key** | **default** | **Description** |
| 1 | admin\_id | | VARCHAR(20) | ✔ | PK |  | 관리자 아이디 |
| 2 | admin\_password | | VARCHAR(20) | ✔ |  |  | 관리자 비밀번호 |
| 3 | admin\_authority | | SMALLINT | ✔ |  |  | 관리자 권한 |
| **비고** | | | | | | | |
|  | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table**  **Name** | | User | **Table 기술서** | | | 작성일 | 2018.11.12 |
| **System** | | 과외시스템 | 작성자 | 허준석 |
| **No.** | **Column name** | | **Data type** | **not null** | **Key** | **default** | **Description** |
| 1 | user\_ID | | VARCHAR(20) | ✔ | PK |  | 회원아이디 |
| 2 | user\_Password | | VARCHAR(20) | ✔ |  |  | 회원비밀번호 |
| 3 | user\_Name | | VARCHAR(20) | ✔ |  |  | 회원이름 |
| 4 | user\_Type | | SMALLINT | ✔ |  |  | 회원타입 |
| 5 | user\_Contact | | VARCHAR(20) | ✔ |  |  | 회원전화번호 |
| 6 | user\_City | | VARCHAR(20) | ✔ |  |  | 회원주소정보 |
| 7 | user\_Point | | INTEGER | ✔ |  | 0 | 회원포인트 |
| **비고** | | | | | | | |
|  | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table**  **Name** | | Spec | **Table 기술서** | | | 작성일 | 2018.11.12 |
| **System** | | 과외시스템 | 작성자 | 허준석 |
| **No.** | **Column name** | | **Data type** | **not null** | **Key** | **default** | **Description** |
| 1 | user\_ID | | VARCHAR(20) | ✔ | PFK |  | 회원아이디 |
| 3 | spec\_name | | VARCHAR(20) | ✔ | PK |  | 스펙내용 |
| **비고** | | | | | | | |
| \* PFK(user\_ID) -> User(user\_ID) | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table**  **Name** | | Schedule | **Table 기술서** | | | 작성일 | 2018.11.12 |
| **System** | | 과외시스템 | 작성자 | 허준석 |
| **No.** | **Column name** | | **Data type** | **not null** | **Key** | **default** | **Description** |
| 1 | user\_ID | | VARCHAR(20) | ✔ | PFK |  | 회원아이디 |
| 2 | schedule\_id | | INTEGER | ✔ | PK |  | 스케줄아이디 |
| 3 | schedule\_start | | TIMESTAMP | ✔ |  |  | 스케줄시작일 |
| 4 | schedule\_end | | TIMESTAMP | ✔ |  |  | 스케줄종료일 |
| 5 | schedule\_content | | TEXT | ✔ |  |  | 스케줄내용 |
| **비고** | | | | | | | |
| \* PFK(user\_ID) -> User(user\_ID) | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table**  **Name** | | Payment | **Table 기술서** | | | 작성일 | 2018.11.12 |
| **System** | | 과외시스템 | 작성자 | 허준석 |
| **No.** | **Column name** | | **Data type** | **not null** | **Key** | **default** | **Description** |
| 1 | user\_ID | | VARCHAR(20) | ✔ | PFK |  | 회원아이디 |
| 2 | payment\_id | | INTEGER | ✔ | PK |  | 결제아이디 |
| 3 | payment\_method | | VARCHAR(20) | ✔ |  |  | 결제방법 |
| 4 | payment\_company | | VARCHAR(20) | ✔ |  |  | 결제회사 |
| 5 | payment\_date | | TIMESTAMP | ✔ |  | NOW() | 결제날짜 |
| **비고** | | | | | | | |
| \* PFK(user\_ID) -> User(user\_ID) | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table**  **Name** | | Message | **Table 기술서** | | | 작성일 | 2018.11.12 |
| **System** | | 과외시스템 | 작성자 | 허준석 |
| **No.** | **Column name** | | **Data type** | **not null** | **Key** | **default** | **Description** |
| 1 | message\_ID | | Integer | ✔ | PK |  | 메시지아이디 |
| 2 | message\_Title | | VARCHAR(50) | ✔ |  |  | 메시지제목 |
| 3 | message\_Content | | TEXT | ✔ |  |  | 메시지내용 |
| 4 | message\_Date | | Timestamp | ✔ |  | NOW() | 메시지보낸날짜 |
| 5 | message\_Sender | | VARCHAR(20) | ✔ | FK |  | 메시지보낸사람 |
| 6 | message\_Receiver | | VARCHAR(20) | ✔ | FK |  | 메시지받은사람 |
| **비고** | | | | | | | |
| \* FK(message\_receiver) -> User(user\_ID)  \* FK(message\_sender) -> User(user\_ID) | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table**  **Name** | | Test | **Table 기술서** | | | 작성일 | 2018.11.12 |
| **System** | | 과외시스템 | 작성자 | 허준석 |
| **No.** | **Column name** | | **Data type** | **not null** | **Key** | **default** | **Description** |
| 1 | test\_id | | Integer | ✔ | PK |  | 시험아이디 |
| 2 | test\_score | | Integer |  |  |  | 시험점수 |
| 3 | test\_complete | | Boolean(1) | ✔ |  | 0 | 시험완료여부 |
| 4 | test\_start | | Timestamp |  |  |  | 시험시작날짜 |
| 5 | test\_finish | | Timestamp |  |  |  | 시험끝난날짜 |
| 6 | test\_sender | | VARCHAR(20) | ✔ | FK |  | 시험보낸사람 |
| 7 | test\_receiver | | VARCHAR(20) | ✔ | FK |  | 시험받은사람 |
| **비고** | | | | | | | |
| \* FK(test\_receiver) -> User(user\_ID)  \* FK(test\_sender) -> User(user\_ID) | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table**  **Name** | | Question | **Table 기술서** | | | 작성일 | 2018.11.12 |
| **System** | | 과외시스템 | 작성자 | 허준석 |
| **No.** | **Column name** | | **Data type** | **not null** | **Key** | **default** | **Description** |
| 1 | test\_id | | VARCHAR(20) | ✔ | PFK |  | 시험아이디 |
| 2 | question\_id | | Integer | ✔ | PK |  | 문제아이디 |
| 3 | question\_content | | TEXT | ✔ |  |  | 문제내용 |
| 4 | question\_answer | | TINYINT | ✔ |  |  | 문제답 |
| **비고** | | | | | | | |
| \* PFK(test\_id) -> Test(test\_id) | | | | | | | |

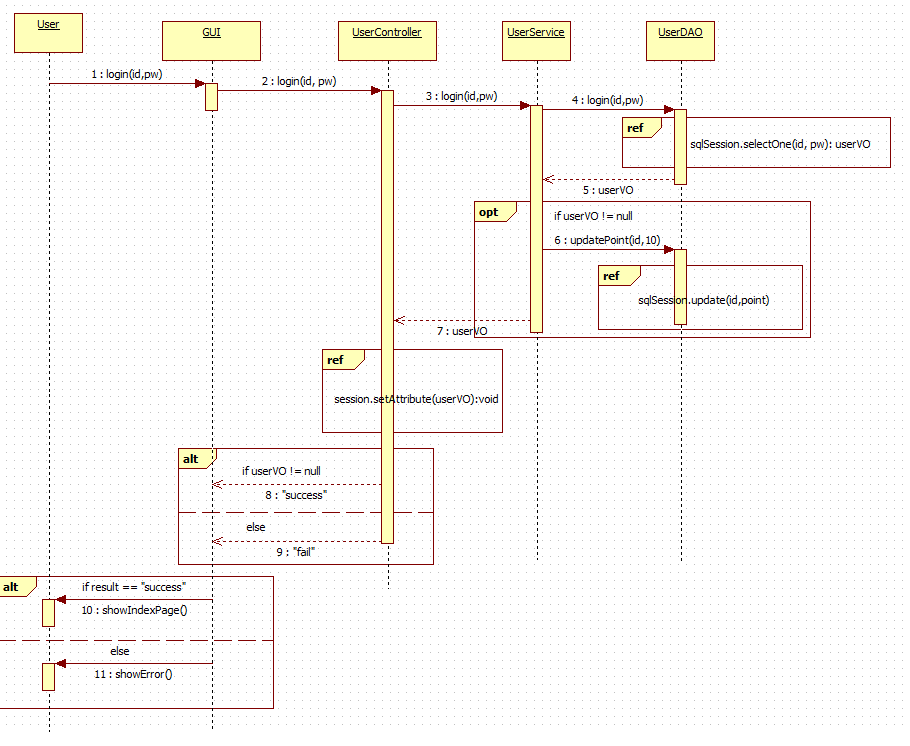
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table**  **Name** | | Lesson | **Table 기술서** | | | 작성일 | 2018.11.12 |
| **System** | | 과외시스템 | 작성자 | 허준석 |
| **No.** | **Column name** | | **Data type** | **not null** | **Key** | **default** | **Description** |
| 1 | lesson\_Id | | Integer | ✔ | PK |  | 과외아이디 |
| 2 | lesson\_Subjcet | | VARCHAR(20) | ✔ |  |  | 과외과목 |
| 3 | lesson\_Date | | VARCHAR(20) | ✔ |  |  | 과외시간 |
| 3 | lesson\_price | | Integer | ✔ |  |  | 과외가격 |
| 4 | lesson\_contact | | TINYINT | ✔ |  |  | 과외접선방법 |
| 5 | lesson\_Teacher | | VARCHAR(20) | ✔ | FK |  | 과외교사 |
| **비고** | | | | | | | |
| \* FK(lesson\_Teacher) -> User(user\_ID) | | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table**  **Name** | | Review | **Table 기술서** | | | 작성일 | 2018.11.12 |
| **System** | | 과외시스템 | 작성자 | 허준석 |
| **No.** | **Column name** | | **Data type** | **not null** | **Key** | **default** | **Description** |
| 1 | lesson\_Id | | Integer | ✔ | PFK |  | 과외아이디 |
| 2 | review\_id | | Integer | ✔ | PK |  | 리뷰아이디 |
| 3 | review\_content | | TEXT | ✔ |  |  | 리뷰내용 |
| 4 | review\_star | | TINYINT | ✔ |  |  | 리뷰별점 |
| **비고** | | | | | | | |
| \* PFK(lesson\_Id) -> Lesson(lesson\_Id) | | | | | | | |

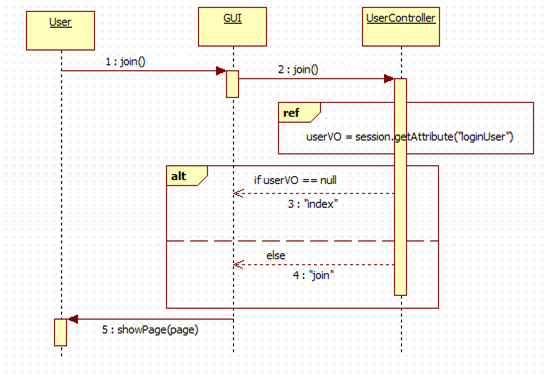
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table**  **Name** | | Sugang | **Table 기술서** | | | 작성일 | 2018.11.12 |
| **System** | | 과외시스템 | 작성자 | 허준석 |
| **No.** | **Column name** | | **Data type** | **not null** | **Key** | **default** | **Description** |
| 1 | user\_ID | | VARCHAR(20) | ✔ | PFK |  | 멤버아이디 |
| 2 | lesson\_Id | | Integer | ✔ | PFK |  | 과외아이디 |
| 3 | lesson\_Paycheck | | boolean | ✔ |  | 0 | 수강결제 완료확인 |
| 4 | lesson\_PayDate | | TIMESTAMP | ✔ |  |  | 수강날짜 |
| **비고** | | | | | | | |
| \* PFK(user\_ID) -> User(user\_ID)  \* PFK(lesson\_Id) -> Lesson(lesson\_Id) | | | | | | | |

|  |
| --- |
| **시퀀스 다이어그램** |

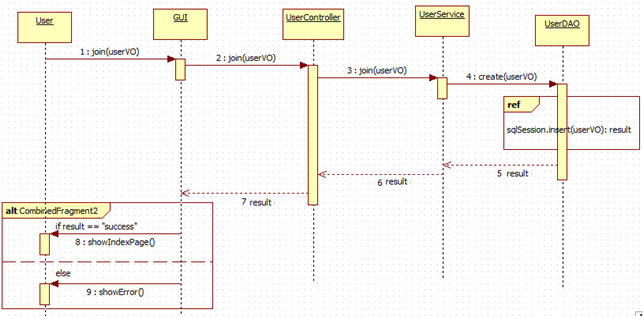
- 로그인



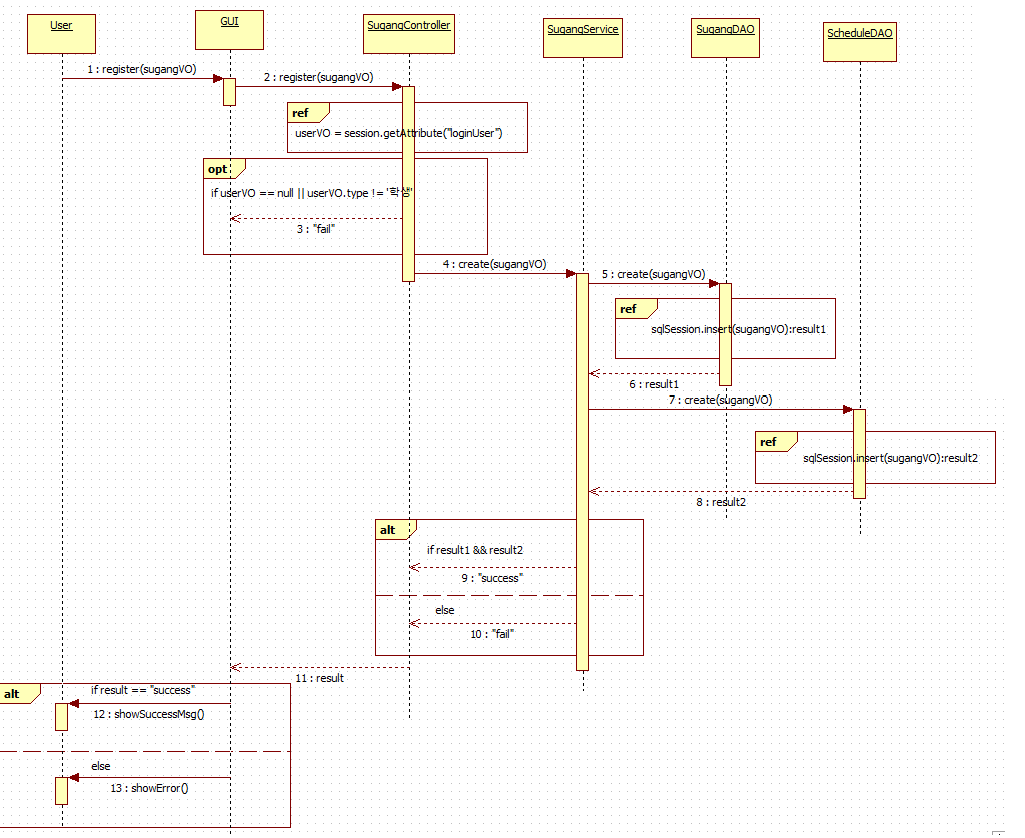
- 회원가입 페이지 요청



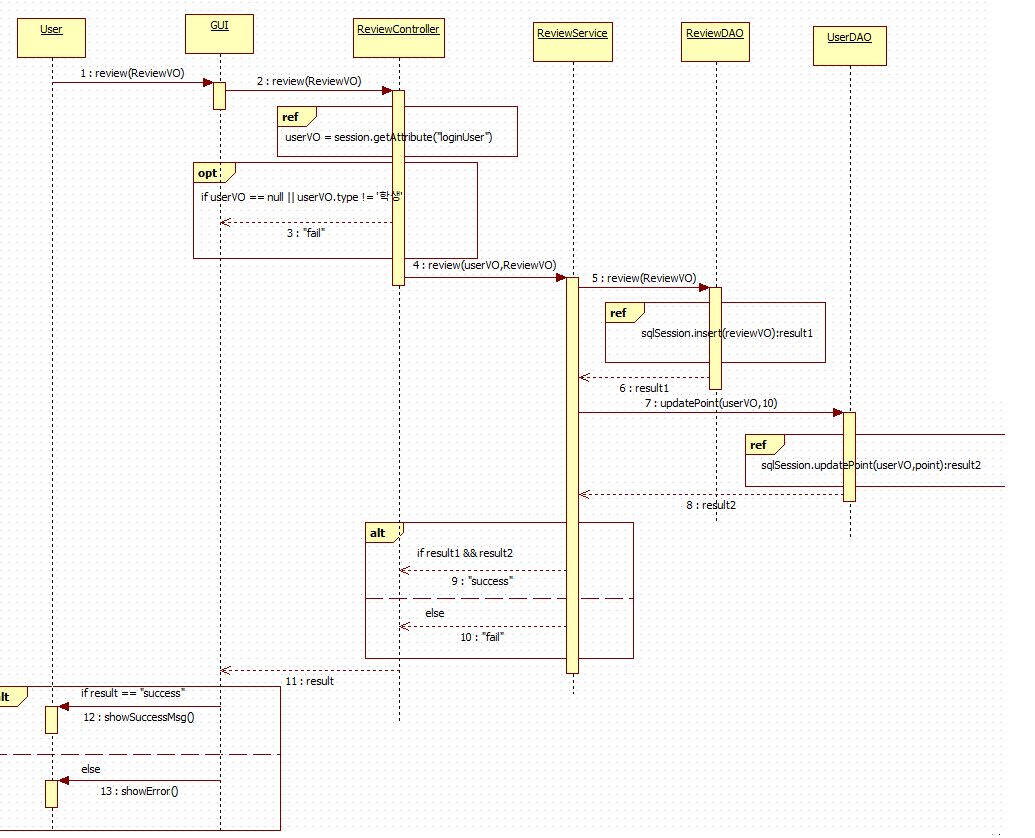
- 회원가입 요청



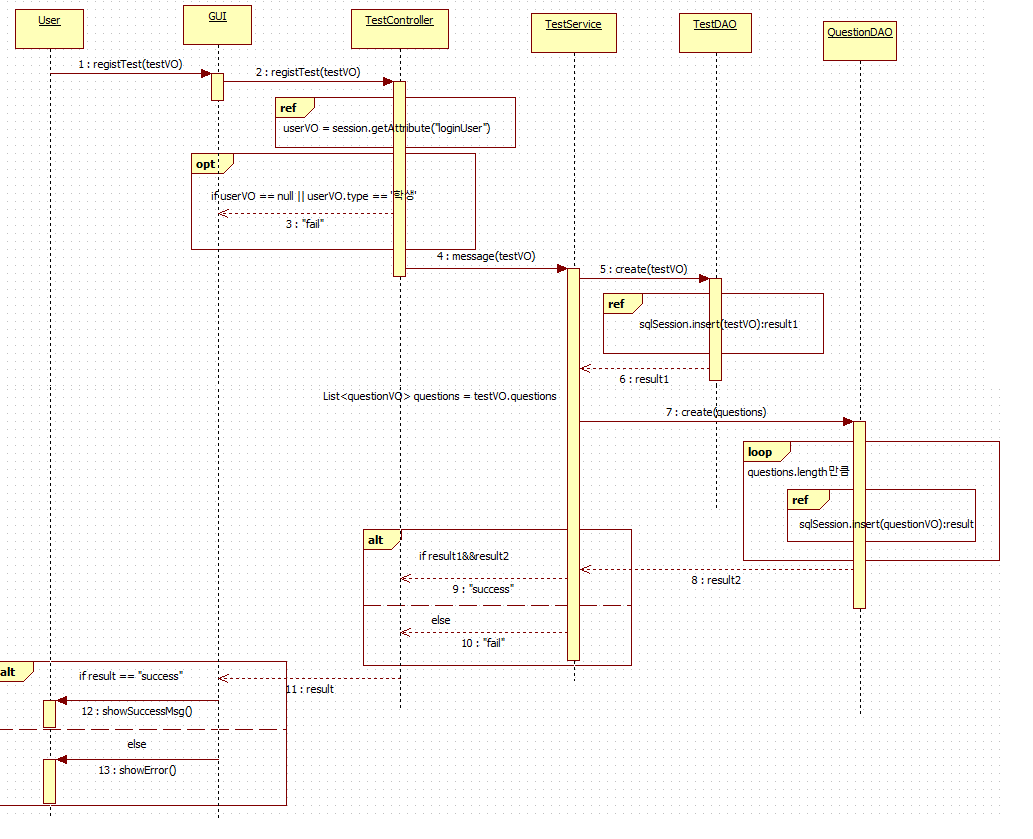
- 수강 신청



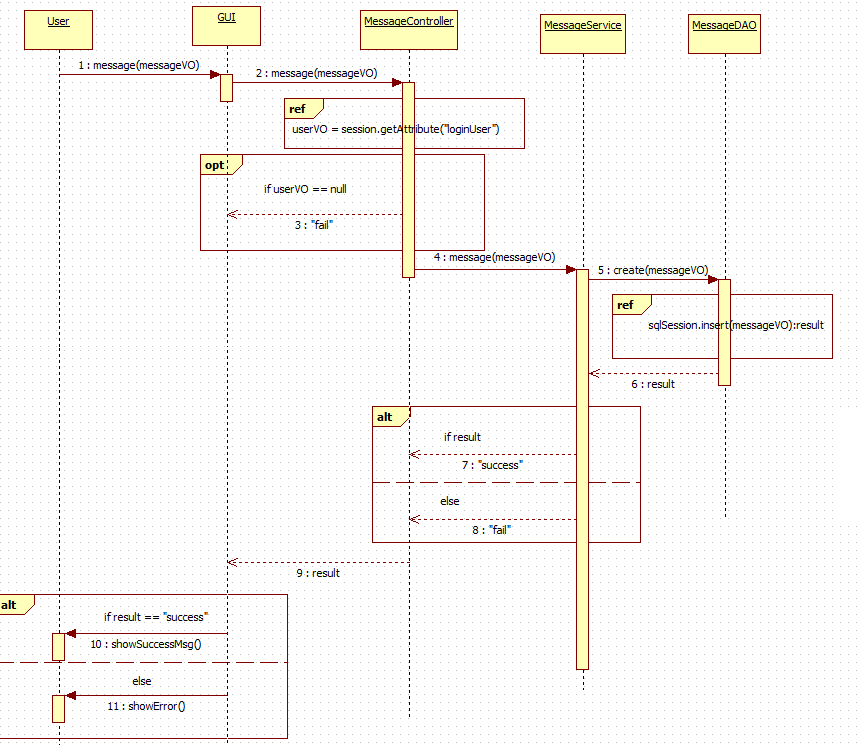
- 리뷰 등록



- 테스트 등록

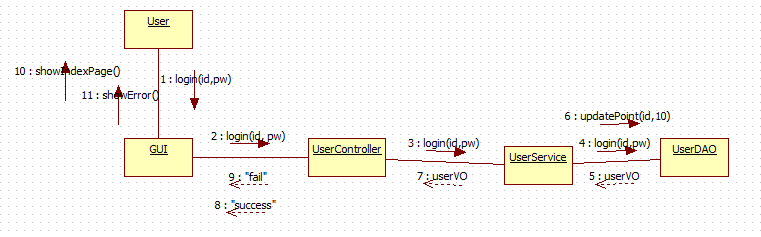


- 메시지 전송

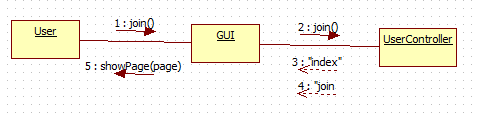


|  |
| --- |
| **커뮤니케이션 다이어그램** |

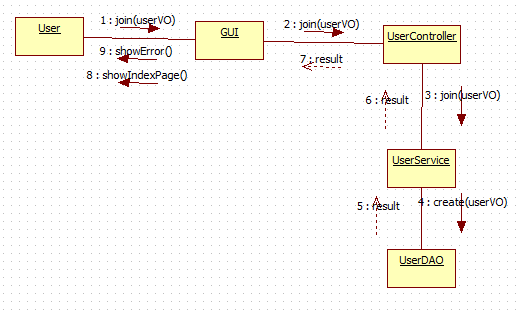
- 로그인



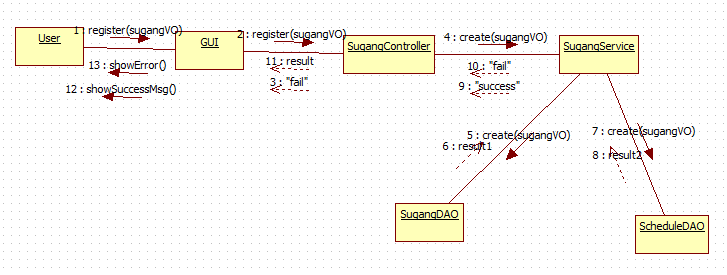
- 회원가입 페이지 요청



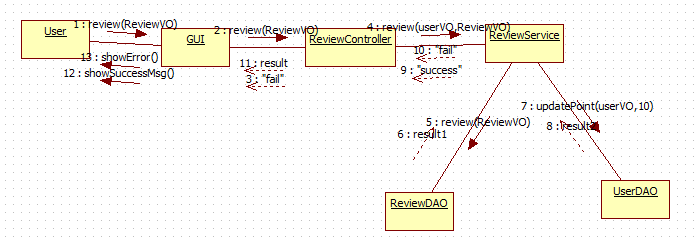
- 회원가입 요청



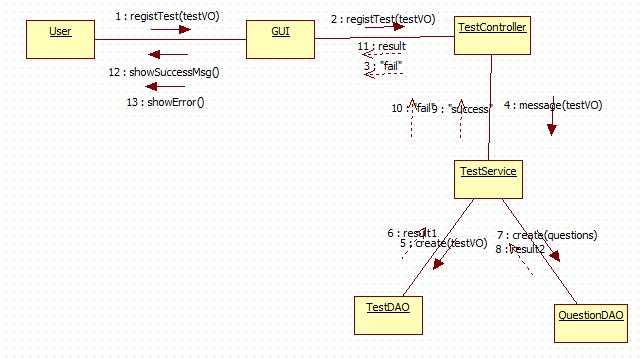
- 수강 신청



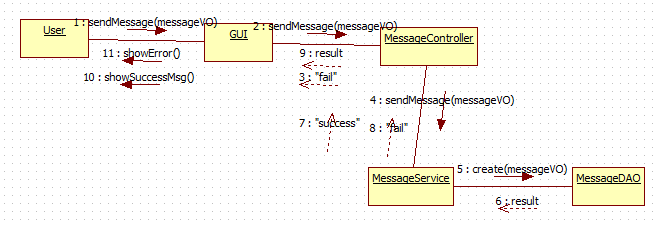
- 리뷰 등록



- 테스트 등록



- 메시지 전송



|  |
| --- |
| **소스 프로그램의 헤더 주석** |

/\*\*

\* 파일명: UserController.java

\* 버전번호: 1.0.0

\* 작성자명: 김성건

\* 프로젝트명: sogong

\* 연관 클래스 리스트: UserService, SugangService

\* 기능설명: /user/\*에 해당하는 요청을 받아서 처리한다.

\*/

**package** com.sogong.controller;

**import** javax.inject.Inject;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpSession;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** com.sogong.domain.UserVO;

**import** com.sogong.service.SugangService;

**import** com.sogong.service.UserService;

@Controller

@RequestMapping(value = "/user/\*")

**public** **class** UserController {

// 스프링이 주입해준다.

@Inject

**private** UserService userService;

// 스프링이 주입해준다.

@Inject

**private** SugangService sugangService;

/\*\* /user/join (GET)요청시 호출되는 메서드다.

\* 회원가입 페이지를 리턴한다.

\* 이미 로그인이 된 상태에서는 인덱스 페이지를 리턴한다.

\*/

@RequestMapping(value = "/join", method = RequestMethod.***GET***)

**public** String joinGET(HttpSession session) {

UserVO userVO = (UserVO) session.getAttribute("loginUser");

**if** (userVO == **null**) {

**return** "/user/join";

} **else** {

**return** "redirect:/";

}

}

/\*\* /user/join (POST)요청시 호출되는 메서드다.

\* 파라미터로 userVO 객체를 받는다.

\* DB에 userVO를 등록하고 정상적으로 성공했는지를 리턴한다.

\*/

@RequestMapping(value = "/join", method = RequestMethod.***POST***)

**public** ResponseEntity<String> joinPOST(UserVO userVO) {

ResponseEntity<String> entity = **null**;

**try** {

userService.create(userVO);

entity = **new** ResponseEntity<String>("SUCCESS", HttpStatus.***OK***);

} **catch** (Exception e) {

entity = **new** ResponseEntity<String>("FAIL", HttpStatus.***BAD\_REQUEST***);

}

**return** entity;

}

/\*\* /user/checkID (POST)요청시 호출되는 메서드다.

\* DB에 파라미터로 넘긴 id가 이미 있는지를 확인한 뒤 리턴한다.

\*/

@RequestMapping(value = "/checkID", method = RequestMethod.***POST***)

**public** ResponseEntity<String> checkID(@RequestParam String id) {

ResponseEntity<String> entity = **null**;

**if** (userService.checkID(id)) {

entity = **new** ResponseEntity<String>("SUCCESS", HttpStatus.***OK***);

} **else** {

entity = **new** ResponseEntity<String>("FAIL", HttpStatus.***OK***);

}

**return** entity;

}

/\*\* /user/list (GET)요청시 호출되는 메서드다.

\* 유저 목록 페이지를 리턴한다.

\* 만약 로그인한 유저가 admin이 아니라면 인덱스 페이지를 리턴한다.

\*/

@RequestMapping(value = "/list", method = RequestMethod.***GET***)

**public** String list(HttpSession session, Model model) {

UserVO userVO = (UserVO) session.getAttribute("loginUser");

**if** (userVO != **null** && userVO.getId().equals("admin")) {

model.addAttribute("userList", userService.list());

**return** "/user/list";

} **else** {

**return** "redirect:/";

}

}

/\*\* /user/delete (POST)요청시 호출되는 메서드다.

\* 파라미터로 id값을 받는다.

\* DB에서 id값에 해당하는 유저를 삭제하고 그 결과를 리턴한다.

\*/

@RequestMapping(value = "/delete", method = RequestMethod.***POST***)

**public** ResponseEntity<String> delete(@RequestParam String id) {

ResponseEntity<String> entity = **null**;

**try** {

userService.delete(id);

entity = **new** ResponseEntity<String>("SUCCESS", HttpStatus.***OK***);

} **catch** (Exception e) {

entity = **new** ResponseEntity<String>("FAIL", HttpStatus.***BAD\_REQUEST***);

}

**return** entity;

}

/\*\* /user/info (GET)요청시 호출되는 메서드다.

\* 유저 정보 페이지를 리턴한다.

\* 로그인 체크는 com.sogong.interceptor.Authinterceptor 클래스에서 인터셉트해서 확인한다.

\*/

@RequestMapping(value = "/info", method = RequestMethod.***GET***)

**public** **void** info() {

}

/\*\* /user/update (POST)요청시 호출되는 메서드다.

\* 파라미터로 userVO 객체를 받는다.

\* DB에 userVO.id에 해당하는 객체를 수정한 뒤 결과를 리턴한다.

\*/

@RequestMapping(value = "/update", method = RequestMethod.***POST***)

**public** ResponseEntity<String> update(HttpSession session, UserVO userVO) {

ResponseEntity<String> entity = **null**;

**try** {

userService.update(userVO);

session.setAttribute("loginUser", userVO);

entity = **new** ResponseEntity<String>("SUCCESS", HttpStatus.***OK***);

} **catch** (Exception e) {

e.printStackTrace();

entity = **new** ResponseEntity<String>("FAIL", HttpStatus.***BAD\_REQUEST***);

}

**return** entity;

}

/\*\* /user/sugang (GET)요청시 호출되는 메서드다.

\* 유저 수강 정보 페이지를 리턴한다.

\* 로그인 체크는 com.sogong.interceptor.Authinterceptor 클래스에서 인터셉트해서 확인한다.

\*/

@RequestMapping(value = "/sugang", method = RequestMethod.***GET***)

**public** **void** sugang(HttpSession session, Model model) {

UserVO userVO = (UserVO) session.getAttribute("loginUser");

model.addAttribute("sugangList", sugangService.list(userVO));

}

/\*\* /user/login (POST)요청시 호출되는 메서드다.

\* 파라미터로 id와 password값을 받는다.

\* DB에서 id와 password가 일치하는 유저가 있는지 확인한 뒤 세션에 등록한다.

\*/

@RequestMapping(value = "/login", method = RequestMethod.***POST***)

**public** ResponseEntity<String> login(HttpSession session, @RequestParam String id, @RequestParam String password) {

ResponseEntity<String> entity = **null**;

UserVO userVO = userService.login(id, password);

**if** (userVO != **null**) {

session.setAttribute("loginUser", userVO);

entity = **new** ResponseEntity<String>("SUCCESS", HttpStatus.***OK***);

} **else** {

entity = **new** ResponseEntity<String>("FAIL", HttpStatus.***OK***);

}

**return** entity;

}

/\*\* /user/logout (POST)요청시 호출되는 메서드다.

\* 로그인시 등록했던 세션을 지운다.

\*/

@RequestMapping(value = "/logout", method = RequestMethod.***POST***)

**public** String logout(HttpSession session, HttpServletRequest request) {

session.removeAttribute("loginUser");

**return** "redirect:" + request.getHeader("Referer");

}

}

[붙임1]

public class Account {

public Object id;

public Object pw;

public void sendMessage() {

}

}

public class Admin extends Account {

public Object admin\_id;

public Object admin\_password;

public Object admin\_authority;

public void deleteLesson() {

}

public void deleteMember() {

}

public void searchMember() {

}

}

public class Category {

public Object name;

}

public class CreditCardPayment extends Payment {

public Object companyName;

}

public class Lesson {

public Object lesson\_id;

public Object lesson\_subject;

public Object lesson\_date;

public Object lesson\_price;

public Object lesson\_contact;

public Object lessoon\_teacher;

}

public class LessonService {

public void searchLesson() {

}

public void registAccount() {

}

public void login() {

}

public void logout() {

}

}

public class Member extends Account {

public Object user\_id;

public Object user\_passsword;

public Object user\_name;

public Object user\_type;

public Object user\_contact;

public Object user\_city;

public Object user\_point;

public void inquirySchedule() {

}

public void updateSchedule() {

}

}

public class Message {

public Object message\_id;

public Object message\_title;

public Object message\_date;

public Object message\_sender;

public Object message\_receiver;

public void sendmessage() {

}

}

public class OnlinePayment extends Payment {

public Object depositor;

public Object accountNumber;

}

public class Payment {

public Object payment\_id;

public Object payment\_method;

public Object payment\_company;

public Object payment\_date;

}

public class Question {

public Object question\_id;

public Object question\_content;

public Object question\_answer;

}

public class review {

public Object review\_id;

public Object review\_content;

public Object review\_star;

}

public class Schedule {

public Object schedule\_id;

public Object schedule\_start;

public Object schedule\_end;

public Object schedule\_content;

}

public class spec {

public Object spec\_name;

}

public class StudentMember extends Member {

public Object economy;

public Object interestSubject;

public void signUpLesson() {

}

public void solveQuiz() {

}

public void chargePoint() {

}

public void searchLesson() {

}

public void wirtereview() {

}

}

public class TeacherMember extends Member, Member {

public Object accountNumber;

public void registLesson() {

}

public void submitQuiz() {

}

}

public class Test {

public Object test\_id;

public Object test\_score;

public Object test\_complete;

public Object test\_start;

public Object test\_end;

public Object test\_finish;

public Object test\_sender;

public Object test\_receiver;

public void addQuestion() {

}

}

[붙임2]

CREATE TABLE admin (

admin\_id VARCHAR2(20) NOT NULL,

admin\_password VARCHAR2(20) NOT NULL,

admin\_authority SMALLINT NOT NULL

);

ALTER TABLE admin ADD CONSTRAINT admin\_pk PRIMARY KEY ( admin\_id );

CREATE TABLE lesson (

lesson\_id INTEGER NOT NULL,

lesson\_subject VARCHAR2(20) NOT NULL,

lessondate VARCHAR2(20) NOT NULL,

lesson\_price INTEGER NOT NULL,

lesson\_contact SMALLINT NOT NULL,

lesson\_date DATE NOT NULL,

lesson\_teacher VARCHAR2(20) NOT NULL

);

ALTER TABLE lesson ADD CONSTRAINT lesson\_pk PRIMARY KEY ( lesson\_id );

CREATE TABLE message (

message\_id INTEGER NOT NULL,

message\_title VARCHAR2(50) NOT NULL,

message\_content CLOB NOT NULL,

message\_date TIMESTAMP NOT NULL,

message\_sender VARCHAR2(20) NOT NULL,

message\_receiver VARCHAR2(20) NOT NULL

);

ALTER TABLE message ADD CONSTRAINT message\_pk PRIMARY KEY ( message\_id );

CREATE TABLE payment (

user\_id VARCHAR2(20) NOT NULL,

payment\_id INTEGER NOT NULL,

payment\_method VARCHAR2(20) NOT NULL,

payment\_company VARCHAR2(20) NOT NULL,

payment\_date TIMESTAMP NOT NULL

);

ALTER TABLE payment ADD CONSTRAINT payment\_pk PRIMARY KEY ( user\_id,

payment\_id );

CREATE TABLE question (

test\_id INTEGER NOT NULL,

question\_id INTEGER NOT NULL,

question\_content CLOB NOT NULL,

answer SMALLINT NOT NULL

);

ALTER TABLE question ADD CONSTRAINT question\_pk PRIMARY KEY ( question\_id,

test\_id );

CREATE TABLE review (

lesson\_id INTEGER NOT NULL,

review\_id INTEGER NOT NULL,

review\_content CLOB NOT NULL,

review\_star SMALLINT NOT NULL

);

ALTER TABLE review ADD CONSTRAINT review\_pk PRIMARY KEY ( review\_id,

lesson\_id );

CREATE TABLE schedule (

user\_id VARCHAR2(20) NOT NULL,

schedule\_id INTEGER NOT NULL,

schedule\_start TIMESTAMP NOT NULL,

schedule\_end TIMESTAMP NOT NULL,

schedule\_content CLOB NOT NULL

);

ALTER TABLE schedule ADD CONSTRAINT schedule\_pk PRIMARY KEY ( user\_id,

schedule\_id );

CREATE TABLE spec (

user\_id VARCHAR2(20) NOT NULL,

spec\_name VARCHAR2(20) NOT NULL

);

ALTER TABLE spec ADD CONSTRAINT spec\_pk PRIMARY KEY ( user\_id,

spec\_name );

CREATE TABLE sugang (

lesson\_paycheck CHAR(1),

lesson\_paydate TIMESTAMP,

user\_id VARCHAR2(20) NOT NULL,

lesson\_id INTEGER NOT NULL

);

ALTER TABLE sugang ADD CONSTRAINT sugang\_pk PRIMARY KEY ( user\_id,

lesson\_id );

CREATE TABLE test (

test\_id INTEGER NOT NULL,

test\_score INTEGER,

test\_complete CHAR(1) NOT NULL,

start\_date TIMESTAMP,

finish\_date TIMESTAMP,

test\_sender VARCHAR2(20) NOT NULL,

test\_receiver VARCHAR2(20) NOT NULL

);

ALTER TABLE test ADD CONSTRAINT test\_pk PRIMARY KEY ( test\_id );

CREATE TABLE "User" (

user\_id VARCHAR2(20) NOT NULL,

user\_password VARCHAR2(20) NOT NULL,

user\_name VARCHAR2(20) NOT NULL,

user\_type SMALLINT NOT NULL,

user\_contact VARCHAR2(20) NOT NULL,

user\_point INTEGER NOT NULL,

user\_city VARCHAR2(20) NOT NULL

);

ALTER TABLE "User" ADD CONSTRAINT user\_pk PRIMARY KEY ( user\_id );

ALTER TABLE lesson

ADD CONSTRAINT lesson\_user\_fk FOREIGN KEY ( lesson\_teacher )

REFERENCES "User" ( user\_id );

ALTER TABLE message

ADD CONSTRAINT message\_user\_fk FOREIGN KEY ( message\_receiver )

REFERENCES "User" ( user\_id );

ALTER TABLE message

ADD CONSTRAINT message\_user\_fkv1 FOREIGN KEY ( message\_sender )

REFERENCES "User" ( user\_id );

ALTER TABLE payment

ADD CONSTRAINT payment\_user\_fk FOREIGN KEY ( user\_id )

REFERENCES "User" ( user\_id );

ALTER TABLE question

ADD CONSTRAINT question\_test\_fk FOREIGN KEY ( test\_id )

REFERENCES test ( test\_id );

ALTER TABLE review

ADD CONSTRAINT review\_lesson\_fk FOREIGN KEY ( lesson\_id )

REFERENCES lesson ( lesson\_id );

ALTER TABLE schedule

ADD CONSTRAINT schedule\_user\_fk FOREIGN KEY ( user\_id )

REFERENCES "User" ( user\_id );

ALTER TABLE spec

ADD CONSTRAINT spec\_user\_fk FOREIGN KEY ( user\_id )

REFERENCES "User" ( user\_id );

ALTER TABLE sugang

ADD CONSTRAINT sugang\_lesson\_fk FOREIGN KEY ( lesson\_id )

REFERENCES lesson ( lesson\_id );

ALTER TABLE sugang

ADD CONSTRAINT sugang\_user\_fk FOREIGN KEY ( user\_id )

REFERENCES "User" ( user\_id );

ALTER TABLE test

ADD CONSTRAINT test\_user\_fk FOREIGN KEY ( test\_receiver )

REFERENCES "User" ( user\_id );

ALTER TABLE test

ADD CONSTRAINT test\_user\_fkv2 FOREIGN KEY ( test\_sender )

REFERENCES "User" ( user\_id );

[붙임3]

/\*\*

\* 파일명: UserController.java

\* 버전번호: 1.0.0

\* 작성자명: 김성건

\* 프로젝트명: sogong

\* 연관 클래스 리스트: UserService, SugangService

\* 기능설명: /user/\*에 해당하는 요청을 받아서 처리한다.

\*/

**package** com.sogong.controller;

**import** javax.inject.Inject;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpSession;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** com.sogong.domain.UserVO;

**import** com.sogong.service.SugangService;

**import** com.sogong.service.UserService;

@Controller

@RequestMapping(value = "/user/\*")

**public** **class** UserController {

// 스프링이 주입해준다.

@Inject

**private** UserService userService;

// 스프링이 주입해준다.

@Inject

**private** SugangService sugangService;

/\*\* /user/join (GET)요청시 호출되는 메서드다.

\* 회원가입 페이지를 리턴한다.

\* 이미 로그인이 된 상태에서는 인덱스 페이지를 리턴한다.

\*/

@RequestMapping(value = "/join", method = RequestMethod.***GET***)

**public** String joinGET(HttpSession session) {

UserVO userVO = (UserVO) session.getAttribute("loginUser");

**if** (userVO == **null**) {

**return** "/user/join";

} **else** {

**return** "redirect:/";

}

}

/\*\* /user/join (POST)요청시 호출되는 메서드다.

\* 파라미터로 userVO 객체를 받는다.

\* DB에 userVO를 등록하고 정상적으로 성공했는지를 리턴한다.

\*/

@RequestMapping(value = "/join", method = RequestMethod.***POST***)

**public** ResponseEntity<String> joinPOST(UserVO userVO) {

ResponseEntity<String> entity = **null**;

**try** {

userService.create(userVO);

entity = **new** ResponseEntity<String>("SUCCESS", HttpStatus.***OK***);

} **catch** (Exception e) {

entity = **new** ResponseEntity<String>("FAIL", HttpStatus.***BAD\_REQUEST***);

}

**return** entity;

}

/\*\* /user/checkID (POST)요청시 호출되는 메서드다.

\* DB에 파라미터로 넘긴 id가 이미 있는지를 확인한 뒤 리턴한다.

\*/

@RequestMapping(value = "/checkID", method = RequestMethod.***POST***)

**public** ResponseEntity<String> checkID(@RequestParam String id) {

ResponseEntity<String> entity = **null**;

**if** (userService.checkID(id)) {

entity = **new** ResponseEntity<String>("SUCCESS", HttpStatus.***OK***);

} **else** {

entity = **new** ResponseEntity<String>("FAIL", HttpStatus.***OK***);

}

**return** entity;

}

/\*\* /user/list (GET)요청시 호출되는 메서드다.

\* 유저 목록 페이지를 리턴한다.

\* 만약 로그인한 유저가 admin이 아니라면 인덱스 페이지를 리턴한다.

\*/

@RequestMapping(value = "/list", method = RequestMethod.***GET***)

**public** String list(HttpSession session, Model model) {

UserVO userVO = (UserVO) session.getAttribute("loginUser");

**if** (userVO != **null** && userVO.getId().equals("admin")) {

model.addAttribute("userList", userService.list());

**return** "/user/list";

} **else** {

**return** "redirect:/";

}

}

/\*\* /user/delete (POST)요청시 호출되는 메서드다.

\* 파라미터로 id값을 받는다.

\* DB에서 id값에 해당하는 유저를 삭제하고 그 결과를 리턴한다.

\*/

@RequestMapping(value = "/delete", method = RequestMethod.***POST***)

**public** ResponseEntity<String> delete(@RequestParam String id) {

ResponseEntity<String> entity = **null**;

**try** {

userService.delete(id);

entity = **new** ResponseEntity<String>("SUCCESS", HttpStatus.***OK***);

} **catch** (Exception e) {

entity = **new** ResponseEntity<String>("FAIL", HttpStatus.***BAD\_REQUEST***);

}

**return** entity;

}

/\*\* /user/info (GET)요청시 호출되는 메서드다.

\* 유저 정보 페이지를 리턴한다.

\* 로그인 체크는 com.sogong.interceptor.Authinterceptor 클래스에서 인터셉트해서 확인한다.

\*/

@RequestMapping(value = "/info", method = RequestMethod.***GET***)

**public** **void** info() {

}

/\*\* /user/update (POST)요청시 호출되는 메서드다.

\* 파라미터로 userVO 객체를 받는다.

\* DB에 userVO.id에 해당하는 객체를 수정한 뒤 결과를 리턴한다.

\*/

@RequestMapping(value = "/update", method = RequestMethod.***POST***)

**public** ResponseEntity<String> update(HttpSession session, UserVO userVO) {

ResponseEntity<String> entity = **null**;

**try** {

userService.update(userVO);

session.setAttribute("loginUser", userVO);

entity = **new** ResponseEntity<String>("SUCCESS", HttpStatus.***OK***);

} **catch** (Exception e) {

e.printStackTrace();

entity = **new** ResponseEntity<String>("FAIL", HttpStatus.***BAD\_REQUEST***);

}

**return** entity;

}

/\*\* /user/sugang (GET)요청시 호출되는 메서드다.

\* 유저 수강 정보 페이지를 리턴한다.

\* 로그인 체크는 com.sogong.interceptor.Authinterceptor 클래스에서 인터셉트해서 확인한다.

\*/

@RequestMapping(value = "/sugang", method = RequestMethod.***GET***)

**public** **void** sugang(HttpSession session, Model model) {

UserVO userVO = (UserVO) session.getAttribute("loginUser");

model.addAttribute("sugangList", sugangService.list(userVO));

}

/\*\* /user/login (POST)요청시 호출되는 메서드다.

\* 파라미터로 id와 password값을 받는다.

\* DB에서 id와 password가 일치하는 유저가 있는지 확인한 뒤 세션에 등록한다.

\*/

@RequestMapping(value = "/login", method = RequestMethod.***POST***)

**public** ResponseEntity<String> login(HttpSession session, @RequestParam String id, @RequestParam String password) {

ResponseEntity<String> entity = **null**;

UserVO userVO = userService.login(id, password);

**if** (userVO != **null**) {

session.setAttribute("loginUser", userVO);

entity = **new** ResponseEntity<String>("SUCCESS", HttpStatus.***OK***);

} **else** {

entity = **new** ResponseEntity<String>("FAIL", HttpStatus.***OK***);

}

**return** entity;

}

/\*\* /user/logout (POST)요청시 호출되는 메서드다.

\* 로그인시 등록했던 세션을 지운다.

\*/

@RequestMapping(value = "/logout", method = RequestMethod.***POST***)

**public** String logout(HttpSession session, HttpServletRequest request) {

session.removeAttribute("loginUser");

**return** "redirect:" + request.getHeader("Referer");

}

}

**package** com.sogong.interceptor;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** javax.servlet.http.HttpSession;

**import** org.springframework.web.servlet.handler.HandlerInterceptorAdapter;

/\*\*

\* 파일명: AuthInterceptor.java

\* 버전번호: 1.0.0

\* 작성자명: 김성건

\* 프로젝트명: sogong

\* 기능설명: servlet-context.xml에서 매핑한 url에 해당하는 요청이 들어오면 인터셉트해서 전처리 또는 후처리를 한다.

\*/

**public** **class** AuthInterceptor **extends** HandlerInterceptorAdapter {

/\*\* servlet-context.xml에서 매핑한 url에 해당하는 요청이 들어왔을 시에 인터셉트해서 전처리 하는 메서드다.

\* 세션에 로그인 유저에 대한 정보가 있으면 true, 없으면 false를 리턴한다.

\* 리턴 값이 true면 본래의 url에 해당하는 컨트롤러로 이어주고, false면 여기서 끝낸다.

\*/

@Override

**public** **boolean** preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) **throws** Exception {

HttpSession session = request.getSession();

**if** (session.getAttribute("loginUser") == **null**) {

response.sendRedirect("/");

**return** **false**;

}

**return** **true**;

}

}

/\*\*

\* 파일명: LessonController.java

\* 버전번호: 1.0.0

\* 작성자명: 김성건

\* 프로젝트명: sogong

\* 연관 클래스 리스트: lessonService, SugangService

\* 기능설명: /lesson/\*에 해당하는 요청을 받아서 처리한다.

\*/

**package** com.sogong.controller;

**import** javax.inject.Inject;

**import** javax.servlet.http.HttpSession;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** com.sogong.domain.LessonVO;

**import** com.sogong.domain.UserVO;

**import** com.sogong.service.LessonService;

**import** com.sogong.service.SugangService;

@Controller

@RequestMapping(value = "/lesson/\*")

**public** **class** LessonController {

// 스프링이 주입해준다.

@Inject

**private** LessonService lessonService;

// 스프링이 주입해준다.

@Inject

**private** SugangService sugangService;

/\*\* /lesson/list (GET)요청시 호출되는 메서드다.

\* DB에서 과외 리스트를 가져온 후 모델에 등록하고 과외 리스트 페이지로 이동시킨다.

\* 만약 옵션 파라미터로 검색 타입과 검색 키워드를 주면 그에 맞는 결과를 모델에 담아준다.

\*/

@RequestMapping(value = "/list", method = RequestMethod.***GET***)

**public** **void** list(@RequestParam(required=**false**) String type, @RequestParam(required=**false**) String keyword, @RequestParam(required=**false**) String contact, Model model) {

model.addAttribute("lessonList", lessonService.list(type, keyword, contact));

}

/\*\* /lesson/register (GET)요청시 호출되는 메서드다.

\* 과외 등록 페이지를 리턴한다.

\* 로그인 체크는 com.sogong.interceptor.Authinterceptor 클래스에서 인터셉트해서 확인한다.

\* 만약 로그인한 유저가 학생 타입이라면 과외 목록 페이지를 리턴한다.

\*/

@RequestMapping(value = "/register", method = RequestMethod.***GET***)

**public** String register(HttpSession session) {

UserVO userVO = (UserVO) session.getAttribute("loginUser");

**if** (userVO.getType() == 1) {

**return** "redirect:/lesson/list";

} **else** {

**return** "/lesson/register";

}

}

/\*\* /lesson/read (GET)요청시 호출되는 메서드다.

\* 파라미터로 과외 id값을 받는다.

\* DB에서 id값에 해당하는 과외 정보를 읽어와서 모델에 등록한 뒤 과외 조회 페이지를 리턴한다.

\* 만약 로그인한 상태라면 해당 과외를 수강했는지 또한 체크해서 그 결과를 모델에 등록한다.

\*/

@RequestMapping(value="/read", method = RequestMethod.***GET***)

**public** **void** read(**int** id, Model model, HttpSession session) {

model.addAttribute("lesson", lessonService.read(id));

UserVO userVO = (UserVO) session.getAttribute("loginUser");

**if** (userVO != **null**) {

model.addAttribute("sugang", sugangService.check(userVO.getId(), String.*valueOf*(id)));

}

}

/\*\* /lesson/register (POST)요청시 호출되는 메서드다.

\* 파라미터로 과외 과목과 가격 값을 받는다.

\* DB에 과외를 등록한 뒤 그 결과를 리턴한다.

\*/

@RequestMapping(value = "/register", method = RequestMethod.***POST***)

**public** ResponseEntity<String> register(String subject, String price, HttpSession session) {

ResponseEntity<String> entity = **null**;

**try** {

UserVO userVO = (UserVO) session.getAttribute("loginUser");

LessonVO lessonVO = **new** LessonVO();

lessonVO.setSubject(subject);

lessonVO.setPrice(Integer.*parseInt*(price));

lessonVO.setTeacher(userVO.getId());

lessonService.create(lessonVO);

entity = **new** ResponseEntity<String>("SUCCESS", HttpStatus.***OK***);

} **catch** (Exception e) {

e.printStackTrace();

entity = **new** ResponseEntity<String>("FAIL", HttpStatus.***BAD\_REQUEST***);

}

**return** entity;

}

}