객체지향분석 및 설계 HW.3

- 프로젝트 명 : GUI 기반 학습 관리 시스템 (투게더러닝)
- 팀원 소개

이름	학번	전공	업무경험	강점 소개
구여진	20212956	소프트웨어전공	Snake game project, PYQT를 이용한 간단한 게임 만들기.	특정 분야에 대한 깊은 지식은 없으나, 새로운 것을 배우는 데 거리낌이 없고 습득력이 좋은 것 같습니다.
강문정	20215205	소프트웨어전공	객지프, c++ 플젝	꼼꼼하고 습득력이 좋으며 집중력이 좋은 것 같습니다.

1. Project 주제

- 프로젝트 목표(vision)
    - 사용자들이 학습 환경을 보다 효율적으로 관리할 수 있는 서비스를 제공하여 학생의 학업 성취도를 높이는 것을 목표로 한다. 본 서비스를 통해 학부모가 학생의 학습 현황을 알고 이를 토대로, 학생에게 맞는 학업 환경 조성에 기여할 수 있도록 하고자 한다. 이에 따라 학부모와 교사가 함께 학생을 지도할 수 있는 분위기를 조성하고자 한다.
- 프로젝트 범위(scope)
    - 사용자들이 학습 환경을 보다 효율적으로 관리할 수 있는 GUI 서비스 구현
    - MySql을 이용한 간단 DB 구현
    - use case
        - 공통
            - 회원가입
            - 로그인
        - 교사
            - 클래스 생성
            - 숙제 등록

- 숙제 채점 및 코멘트 등록
- 공지사항 작성

▪ 학생
- 클래스 가입
- 과제 제출

▪ 학부모
- 클래스 가입
- 학생 학습 현황 확인

2. 기능적인 요구사항

Use case Description

Use case name	회원 가입	
Scenario	새로운 사용자를 위한 회원 가입 기능	
Triggering event	사용자가 서비스를 사용하기 위해 회원 가입을 시도한다.	
Brief description	사용자가 회원 가입을 시도한다. 사용자가 회원 가입 버튼을 누르자 시스템이 회원 가입 화면으로 전환한다. 사용자가 입력란에 사용자 정보를 입력하고 시스템은 해당 정보를 확인한다. 사용자가 모든 조건을 만족하도록 정보를 기입한 뒤, 회원 가입 완료 버튼을 누른다. 시스템은 입력된 정보를 토대로 사용자의 계정을 생성하고 메인 화면으로 전환한다.	
Actors	서비스를 사용해보지 않은 모든 사용자(교사, 학생, 학부모)	
Related use cases	-	
Stakeholders	교사, 학생, 학부모	
Preconditions	사용자의 계정이 존재하지 않아야 한다.	
Post conditions	사용자의 계정이 생성되어야 한다. 메인 화면으로 이동해야 한다.	
Flow of activities	**Actor**	**System**
	1. 사용자가 회원 가입 버튼을 클릭한다 2. 사용자가 이름, 아이디, 비밀번호, 이메일을 입력하고 사용자 유형을 선택한다. 3. 입력한 아이디와 이메일이 중복되었는지 확인한다. 4. 회원 가입 완료 버튼을 누른다.	1-1. 회원 가입 화면으로 이동한다. 2-1. 모든 입력 정보가 형식에 맞는지 확인한다. 3-1. 아이디와 이메일이 기존 정보에 등록되어있는지 확인한다. 4-1. 비어있는 입력란이 있는지 확인한다. 4-2. 사용자의 계정을 생성한다.

		4-3. 메인 화면으로 이동한다.
Exception conditions	2-1. 입력 정보가 지정된 형식에 맞지 않는다면 충족해야 하는 조건을 기입한 문구를 입력란 아래에 출력한다. 3-1. DB에 동일한 아이디나 이메일이 존재한다면 "사용 중인 아이디/이메일 입니다." 라는 문구를 팝업창으로 띄운다. 4-1. 비어있는 입력란이 있다면, "모든 정보를 기입해주세요." 라는 문구를 팝업창으로 띄운다.	

Use case name	로그인	
Scenario	사용자 로그인 기능	
Triggering event	사용자가 서비스를 사용하기 위해 로그인을 시도한다.	
Brief description	사용자가 로그인을 시도한다. 사용자가 로그인 버튼을 누르자 시스템이 로그인 화면으로 전환한다. 사용자가 입력란에 아이디와 패스워드를 입력한 뒤 로그인 버튼을 누른다. 시스템은 사용자가 입력한 정보를 확인하고 사용자 계정에 로그인 시킨 뒤 메인 화면으로 전환한다.	
Actors	모든 사용자(교사, 학생, 학부모)	
Related use cases	-	
Stakeholders	교사, 학생, 학부모	
Preconditions	사용자의 계정이 존재해야 한다.	
Post conditions	유형에 맞는 화면으로 전환되어야 한다.	
Flow of activities	**Actor**	**System**
	1. 사용자가 로그인 버튼을 클릭한다. 2. 입력란에 아이디, 패스워드를 입력한다. 3. 로그인 버튼을 누른다.	1-1. 로그인 화면으로 이동한다. 3-1. 비어있는 입력란이 있는지 확인한다. 3-2. DB에 저장되어있는 정보와 동일한지 확인한다. 3-3. 사용자 계정에 로그인 시킨다. 3-4. 메인 화면으로 이동한다.
Exception conditions	3-1. 비어있는 입력란이 존재한다면 "아이디/비밀번호를 입력해주세요." 라는 문구를 버튼 아래에 출력한다. 3-2. 사용자가 DB에 저장되어있는 정보와 동일하지 않은 정보를 입력했다면 "아이디 또는 비밀번호를 잘못 입력했습니다. 다시 시도해주세요." 라는 문구를 팝업창으로 띄운다.	

Use case name	숙제 등록	
Scenario	교사가 학생을 대상으로 숙제를 등록하는 기능	
Triggering event	교사가 학생에게 숙제를 내려고 한다.	
Brief description	교사가 해당 클래스 내의 숙제 등록 버튼을 누른다. 시스템이 숙제 등록 화면으로 전환한다. 교사는 숙제에 대한 정보를 기입한다. 숙제의 마감 기한을 선택하고 완료 버튼을 누른다. 시스템은 해당 숙제를 등록한다.	
Actors	교사	
Related use cases	로그인	
Stakeholders	교사, 학부모, 학생	
Preconditions	'교사' 계정으로 로그인 되어있어야 한다. 클래스가 존재해야 한다.	
Post conditions	숙제가 등록되어야 한다.	
Flow of activities	**Actor**	**System**
	1. 클래스 화면 내 숙제 등록 버튼을 클릭한다. 2. 숙제 제목과 내용을 입력한다. 3. 완료 버튼을 클릭한다.	1-1. 숙제 등록 화면으로 전환한다. 2-1. 숙제 제목과 내용이 모두 입력되었는지 확인한다. 3-1. 숙제를 등록한다.
Exception conditions	3-1. 숙제 제목 및 내용이 비어있을 경우 "제목/내용을 입력해주세요." 라는 문구를 팝업창으로 띄운다. 3-1. 오류로 숙제 등록에 실패했을 경우 "잠시 후 다시 시도해주세요."라는 문구를 팝업창으로 띄운다.	

Use case name	숙제 채점 및 코멘트 등록
Scenario	교사가 학생이 제출한 숙제를 채점하고 코멘트를 등록하는 기능
Triggering event	교사가 학생의 숙제를 채점하려 한다.
Brief description	교사가 학생의 숙제를 채점하기 위해 학생이 제출한 숙제를 클릭한다. 시스템은 해당 숙제 제출 화면으로 전환한다. 교사는 숙제에 대한 점수와 코멘트를 입력하고 완료 버튼을 클릭한다. 시스템은 해당 내용을 등록한다.
Actors	교사
Related use cases	로그인, 숙제 등록

Stakeholders	교사, 학부모, 학생	
Preconditions	'교사' 계정으로 로그인 되어있어야 한다. 클래스 내에 들어와 있어야 한다. 학생이 제출한 숙제가 존재해야 한다.	
Post conditions	숙제를 채점한 내용이 등록되어야 한다.	
Flow of activities	**Actor**	**System**
	1. 클래스에서 등록한 숙제 중 채점할 숙제를 클릭한다. 2. 학생들이 제출한 숙제 중 채점할 학생의 숙제를 클릭한다. 3. 학생이 제출한 숙제 아래, 점수와 코멘트를 입력한다. 4. 완료 버튼을 누른다.	1-1. 숙제를 제출한 학생들의 목록이 나열된 화면으로 전환한다. 2-1. 해당 학생의 숙제 화면으로 전환한다. 4-1. 입력한 정보의 형식을 확인한다. 4-2. 점수 입력란이 비어있는지 확인한다. 4-3. 채점 결과를 등록한다.
Exception conditions	4-1. 형식이 올바르지 않다면 "올바른 형식으로 입력해주세요." 라는 문구를 입력란 아래에 출력한다. 4-2. 점수 입력란이 비어있다면 "점수를 입력해주세요." 라는 문구를 입력란 아래에 출력한다. 4-3. 등록에 실패했을 경우, "등록에 실패했습니다. 잠시 후 다시 시도해주세요." 라는 문구를 팝업창으로 띄운다.	

Use case name	클래스 생성	
Scenario	교사가 클래스를 생성하는 기능	
Triggering event	사용자가 클래스를 생성하려고 시도한다.	
Brief description	교사가 화면의 '클래스 생성' 버튼을 클릭하면 클래스의 이름과 확인코드를 설정할 수 있는 화면이 나온다. 클래스의 이름과 확인코드를 입력하고 '완료'버튼을 클릭하면 시스템이 클래스를 생성한다.	
Actors	교사	
Related use cases	로그인	
Stakeholders	교사, 학부모, 학생	
Preconditions	'교사' 계정으로 로그인 되어있어야 한다.	
Post conditions	클래스가 생성되어 있어야 한다.	
Flow of	**Actor**	**System**

activities	1. 화면의 '클래스 생성' 버튼을 클릭한다. 2. 클래스의 이름과 확인코드를 입력한다. 3. '완료' 버튼을 클릭한다.	1-1. 클래스 생성 화면으로 이동한다. 3-1. 입력란이 비어있는지 확인한다. 3-2. 입력 정보가 형식에 맞는지 확인한다. 3-3. 사용자의 클래스를 생성한다. 3-4. 생성된 클래스의 홈 화면으로 이동한다.
Exception conditions	3-1. 입력란이 비어있으면 입력란이 비어있다는 문구를 입력란 아래에 출력한다. 3-2. 입력 정보가 형식에 맞지 않으면 입력 정보가 형식에 맞지 않다는 문구를 입력란 아래에 출력한다.	

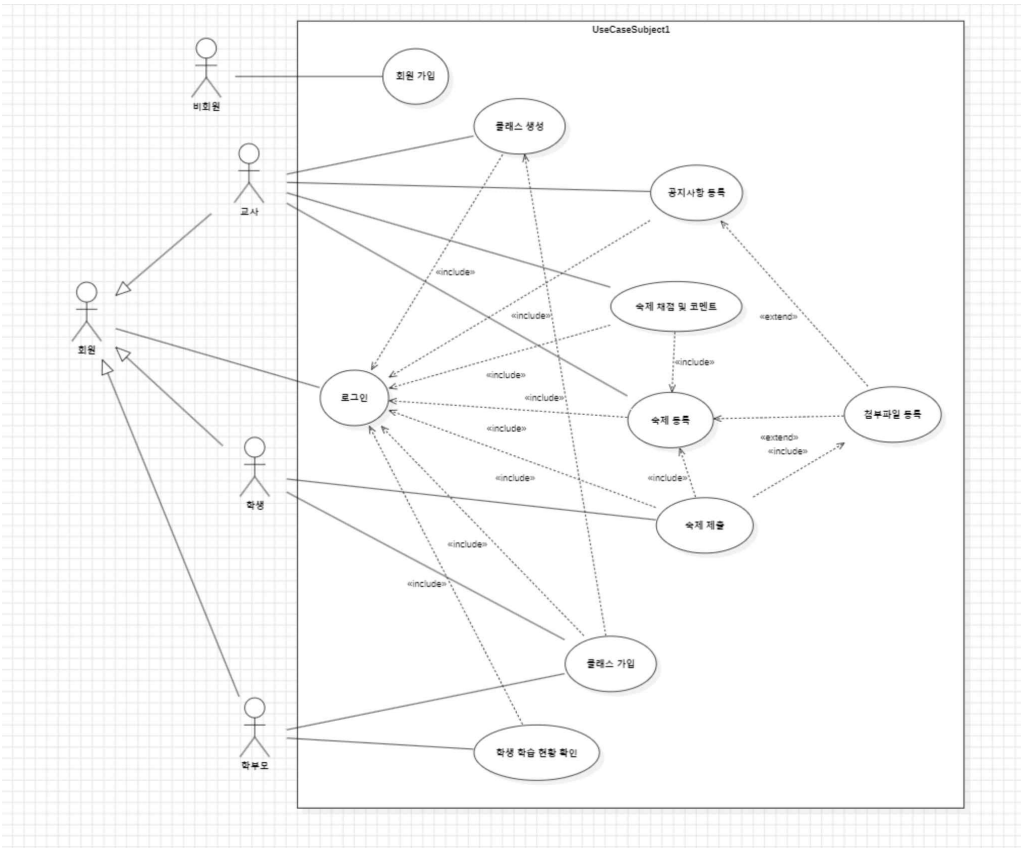
Use case name	클래스 가입	
Scenario	사용자(학생, 학부모)가 클래스에 가입하는 기능	
Triggering event	사용자(학생, 학부모)가 클래스에 가입을 하려고 시도한다.	
Brief description	학부모와 학생 사용자는 로그인 후, 검색을 통해 원하는 클래스를 찾는다. 클래스를 클릭하면 '클래스 가입 버튼'이 활성화된다. '클래스 가입 버튼'을 클릭하면 확인코드를 입력하는 화면이 나온다. 교사 사용자에게 제공받은 확인 코드를 입력하면 클래스에 가입된다.	
Actors	학부모, 학생	
Related use cases	로그인, 클래스 생성	
Stakeholders	교사, 학부모, 학생	
Preconditions	클래스가 생성되어 있어야 한다. '학부모'나 '학생' 계정으로 로그인 되어있어야 한다.	
Post conditions	사용자가 클래스에 가입되어 있어야 한다.	
Flow of activities	**Actor**	**System**
	1. '검색 버튼'을 클릭해 가입하려는 클래스의 이름을 검색한다. 2. 가입하려는 클래스를 선택해 클릭한다. 3. '클래스 가입 버튼'을 클릭한다.	1-1. 검색 할 수 있는 입력란을 보여준다. 1-2. 입력란이 비어있는지 확인한다. 2-1. 클래스를 클릭하면 '클래스 가입 버튼'이 활성화된다.

	4. 교사 사용자에게 제공받은 확인 코드를 입력한다. 5. '완료' 버튼을 클릭한다.	3-1. '클래스 가입 버튼'을 클릭하면 확인 코드를 입력할 수 있는 화면을 보여준다. 4-1. 입력란이 비어있는지 확인한다. 4-2. 확인 코드가 일치하는지 확인한다. 5-1. 사용자를 해당 클래스에 가입시킨다. 5-2. 가입한 클래스의 홈 화면으로 이동한다.
Exception conditions	2-1. 입력란이 비어있으면 입력란이 비어있다는 문구를 입력란 아래에 출력한다. 2-2. 검색한 클래스가 없으면 해당 클래스가 없다는 화면을 보여준다. 4-1. 입력란이 비어있으면 입력란이 비어있다는 문구를 입력란 아래에 출력한다. 4-2. 확인 코드가 일치하지 않으면 확인코드가 일치하지 않는다는 문구가 팝업창으로 나오고 팝업창을 닫으면 확인코드를 입력할 수 있는 화면도 닫힌다.	

Use case name	숙제 제출	
Scenario	학생 사용자가 숙제를 제출하는 기능	
Triggering event	사용자가 클래스에서 숙제를 제출하려고 시도한다.	
Brief description	학생 사용자는 로그인 후, 클래스를 선택해 들어가면 등록된 숙제를 확인할 수 있다. 숙제 수행한 후, '숙제 제출' 버튼을 클릭해 숙제 파일을 업로드한다.	
Actors	학생	
Related use cases	로그인, 숙제 등록	
Stakeholders	교사, 학부모, 학생	
Preconditions	클래스가 생성되어 있어야 한다. 과제가 등록되어 있어야 한다. '학생' 계정으로 로그인 되어있어야 한다.	
Post conditions	사용자가 등록된 숙제를 제출한 상태가 되어야 한다.	
Flow of activities	**Actor**	**System**
	1. 사용자가 등록된 숙제를 클릭한다. 2. 숙제 내용을 확인한다. 3. 수행한 숙제 파일을'완료 버튼'을	1-1. 숙제를 클릭하면 해당 숙제의 내용을 확인할 수 있는 화면으로 이동한다.

	클릭해 업로드한다.	3-1. '숙제 제출 버튼'을 클릭하면 업로드 할 수 있는 상태가 된다. 3-2. '완료 버튼'을 클릭하면 숙제가 제출된다.
Exception conditions	3-2. 숙제 파일을 업로드하지 않고 '완료 버튼'을 클릭하면 제출한 파일이 없다는 문구가 팝업창으로 나온다.	

Use case Diagram



3. 비기능적 요구사항(NFR)

해당 Use Case / 기능	Non-Functional Requirement 내역	Quality	Quality Attributes
모든 Use Case	결과가 신속하게 화면에 나타나도록 한다.	Performance Efficiency	평균 2초 안에 결과가 화면에 나타나도록 한다.
모든 Use Case	운영체제에 구애받지 않고 접속할 수 있게 한다.	Portability	다양한 운영 체제(마이크로소프트 윈도우, macOS 및 Android)에서 사용자가 접속할 수 있어야 한다.

모든 Use Case	사용자가 오류를 적게 경험하도록 한다.	Usability	사용자의 오류 발생율이 5% 이하가 되도록 한다.
모든 Use Case	시스템을 무단 액세스로부터 보호해야 한다.	Security	사용자는 계정에 로그인하기 위해 아이디와 비밀번호를 입력해야 합니다
모든 Use Case	시스템 장애가 생기면 시스템 복구를 신속하게 하도록 한다.	Maintainability	시스템 장애 후 평균 MTTRS는 5분 이내여야 한다.

4. Domain Model 분석하기

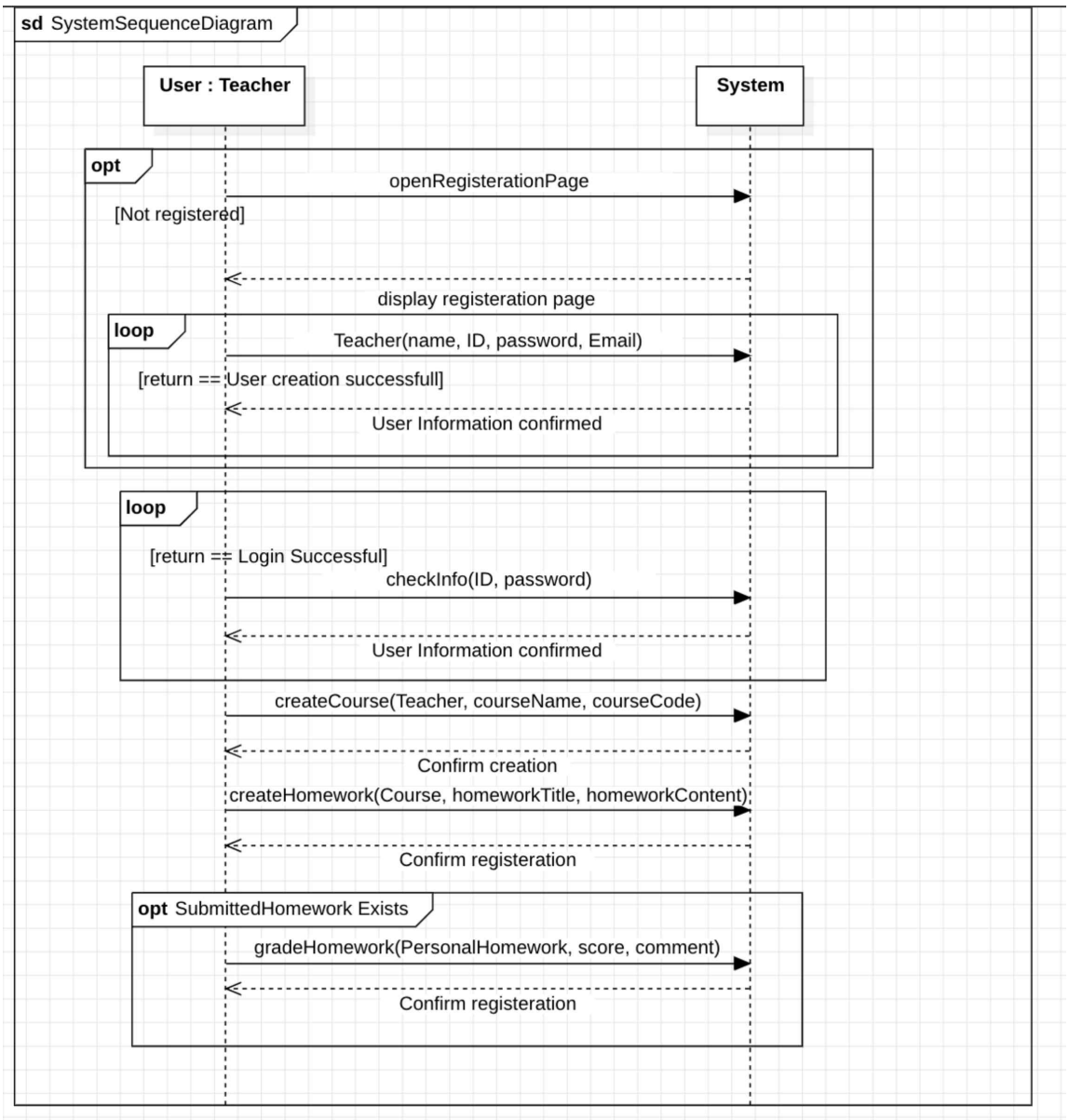
Class 명	Concept 설명	주요 attributes
User	사용자에 대한 클래스, teacher, student, parent의 상위 클래스.	userName, userID, userPassword, userEmail, courseList
Teacher	교사의 정보와 교사가 수행할 method를 다루는 class, user class를 상속 받는다.	
Student	학생의 정보와 학생이 수행할 method를 다루는 class, user class를 상속 받는다.	parentName
Parent	학부모의 정보와 학부모가 수행할 method를 다루는 class, user class를 상속 받는다.	childName
Course	Course 정보를 다루는 class	courseName, courseCode , courseTeacher, courseParentList, courseStudentList, homeworkList, noticeList
Homework	숙제에 대한 정보와 이에 대한 method를 다루는 class	homeworkTitle, homeworkContent, submitStudentsList
PersonalHomework	학생들이 제출한 숙제에 대한 class	submitFile, score, comment
Notice	공지에 대한 정보와 이에 대한 method를 다루는 class	noticeTitle, noticeDate, noticeContent, readCount

UserAndPostCtrl	user와 homework, notice에 대한 기능을 다루는 control class	
CourseCtrl	course에 대한 기능을 다루는 control class	
DBManager	DB와 관련된 기능에 대한 method를 다루는 class	

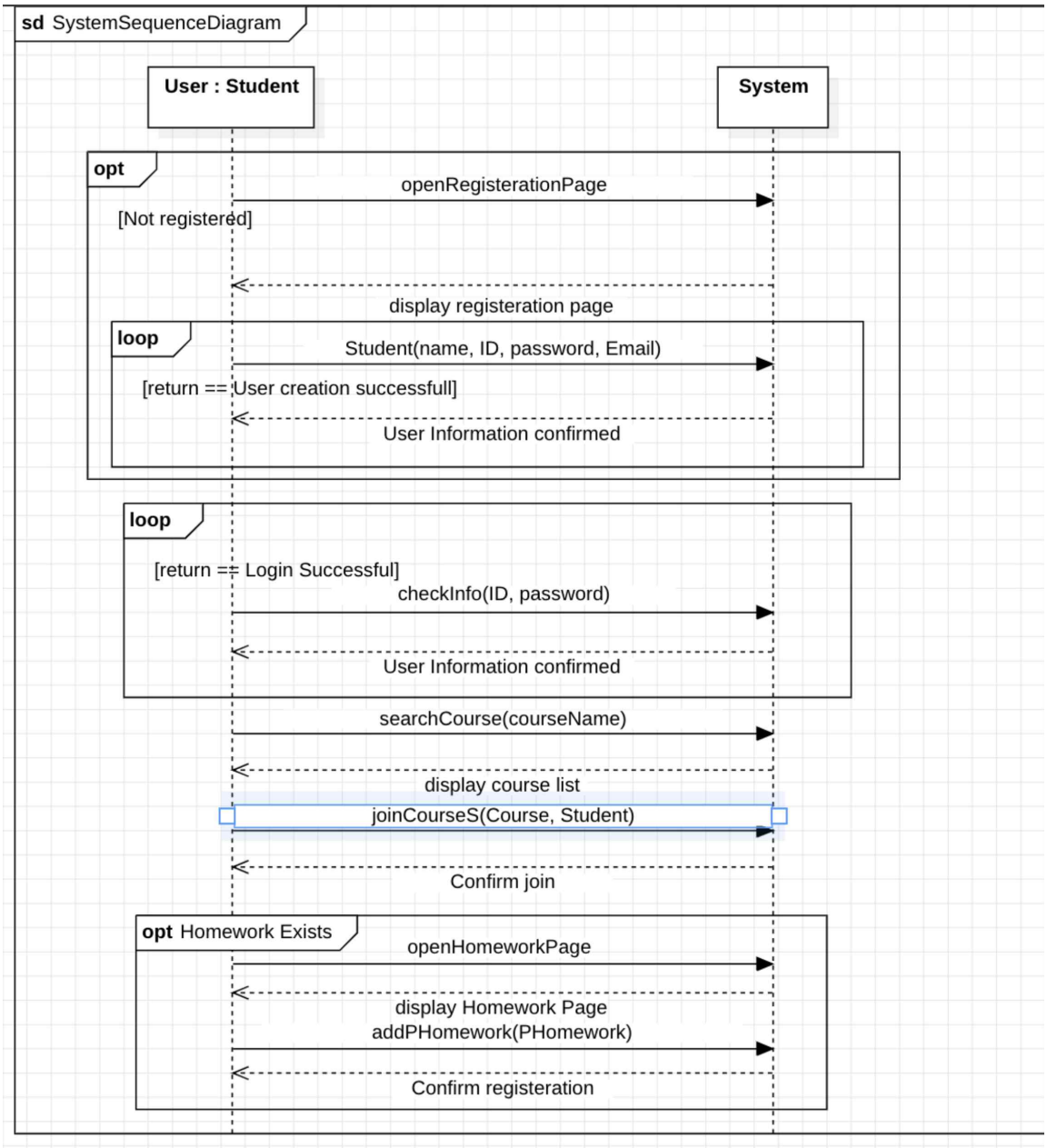
5. Design Model 설계하기

1) System Sequence Diagram

- teacher



- student



2) **Operation Contract**

- Contract : CreateCourse

Operation : createCourse(Teacher, name, code)

Cross Reference : Use Cases : 클래스 생성

Preconditions : Actor was logged in with a 'teacher' account. a teacher instance tch was created.

Postconditions : A course instance cos was created(instance creation). cos.parentList, cos.studentList, cos.Teacher, cos.homeworkList, cos.noticeList were initialized. cos.name, cos.code became name, code. cos was added to tch.courseList(attribute modification). cos was associated with Teacher(association formed).

- Contract : CreateHomework

Operation : createHomework(Course, title, content)

Cross Reference : Use Cases : 숙제 등록

Preconditions : Actor was logged in with a 'teacher' account. A course instance cos was created.

Postconditions : A homework instance hw was created(instance creation). hw.submitStudentsList was initialized. hw.title, hw.content became title, content(attribute modification). hw was associated with Course(association formed)

- Contract : JoinCourseS

Operation : joinCourseS(Course, Student)

Cross Reference : Use Cases : 클래스 가입

Preconditions : Actor was logged in with a 'student' account. A course instance cos was created.

Postconditions : cos was associated with Student(association formed). A student instance std was added to cos.studentList(attribute modification).

- Contract : GradeHomework

Operation : gradeHomework(PersonalHomework, score, comment)

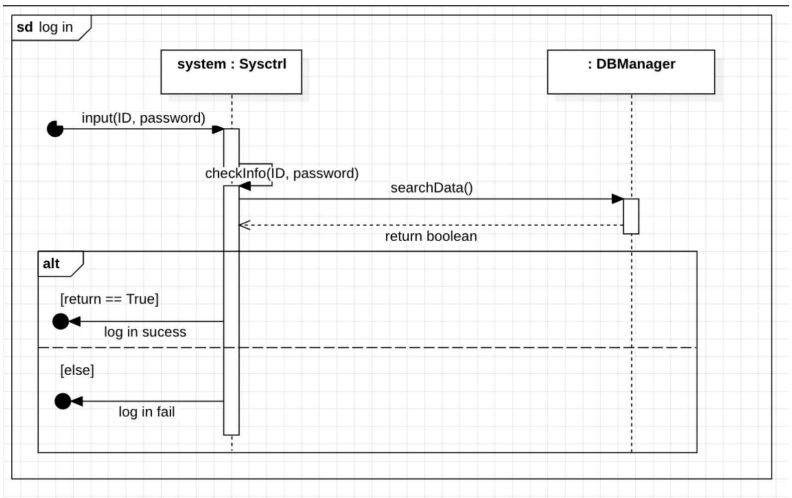
Cross Reference : Use Cases : 숙제 채점 및 코멘트 등록

Preconditions : Actor was logged in with a 'teacher' account. A course instance cos was created. A homework instance hw was created. A personalHomework instance phw was created.

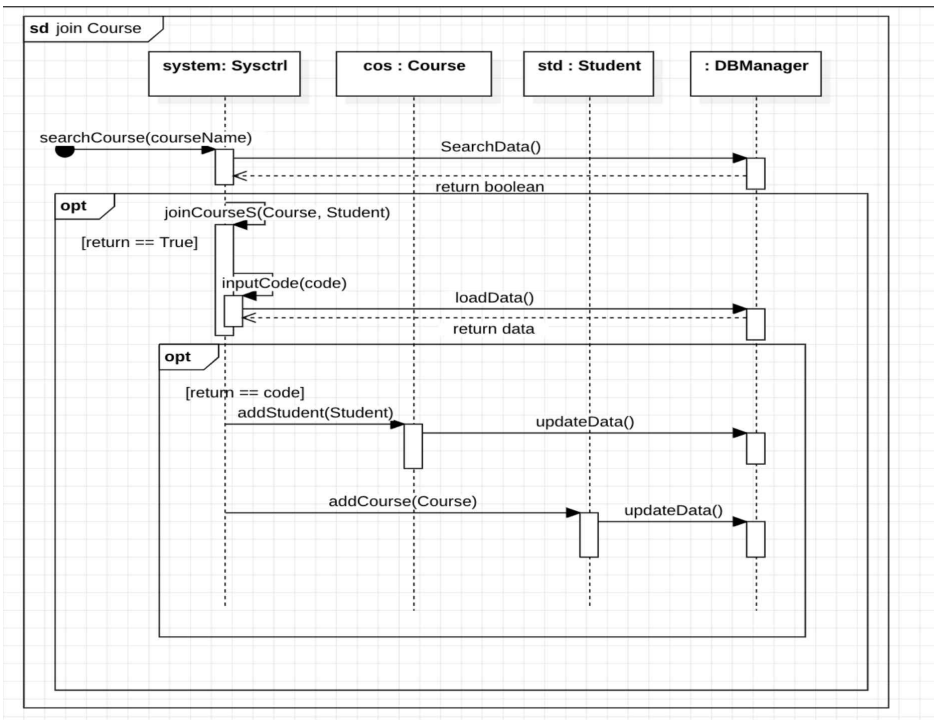
Postconditions : phw.score, paw.comment became score, comment(attribute modification).

3) **Sequence Diagram**

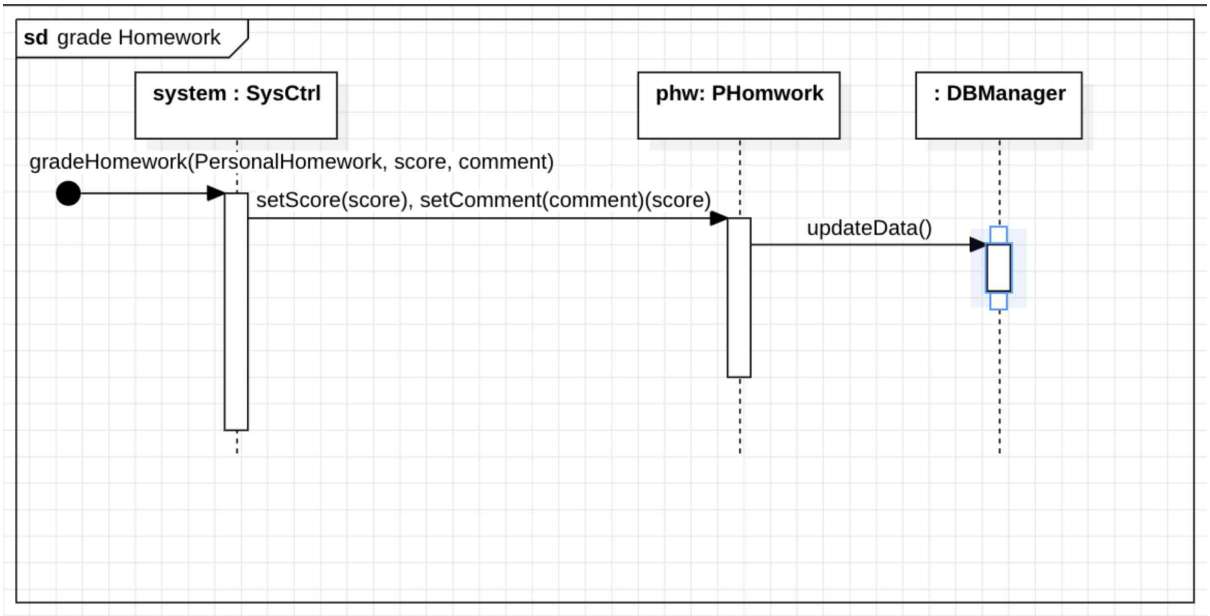
- 로그인 (Student)



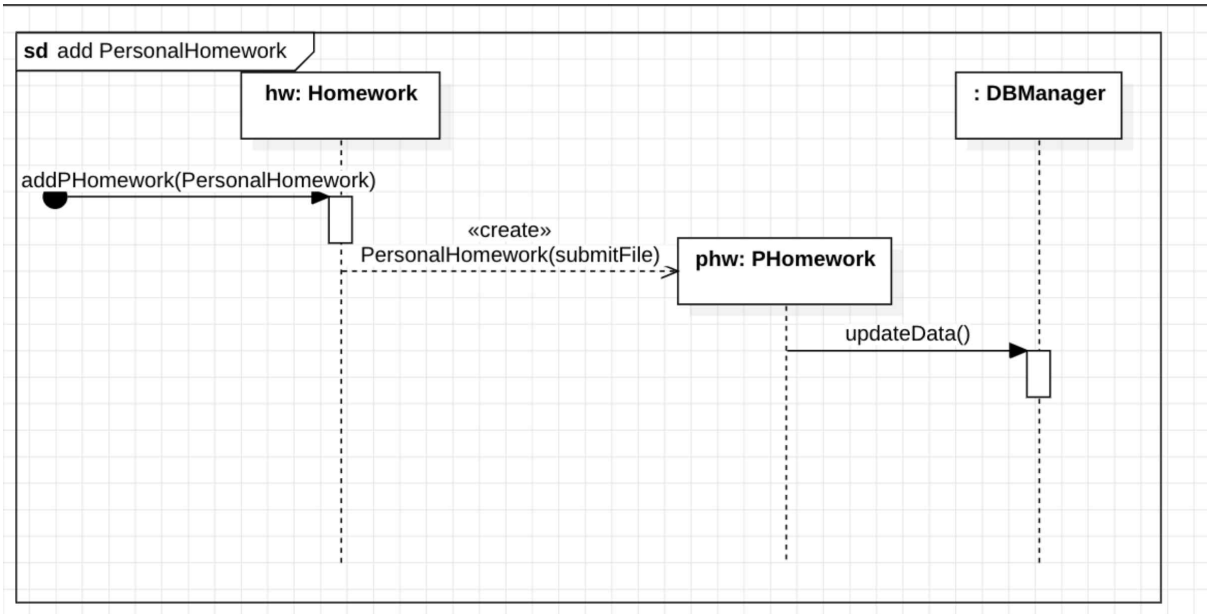
- 클래스 가입



- 숙제 채점 및 코멘트



- 숙제 제출



4) Class Diagram