**Kingo Learning**

**Software Design Specification**

**Team 12**

**팀장 조건희**

**조원 박성빈**

**조원 박민지**

**조원 송재현**

**조원 이태기**

**조원 김정안**

Contents

**1. Preface7**

1.1. Readership7

1.2. Scope7

1.3. Objective7

1.4. Document Structure7

**2. Introduction8**

2.1. Objectives8

2.2. Applied Diagrams8

2.2.1. UML8

2.2.2. Use Case Diagram8

2.2.3. Sequence Diagram9

2.2.4. State Diagram10

2.2.5. Class Diagram10

2.2.6. Context Diagram11

2.3. Applied Tools11

2.3.1. 파워포인트11

2.3.2. diagrams.net11

2.4. Project Scope12

2.5. References12

**3. System Architecture - Overall12**

3.1. Objectives12

3.2. Reuse and Relationships to Other Products13

3.2.1. LearningX13

3.2.2. HelloLMS13

3.2.3. Coursemos13

3.3. System Organization14

3.3.1. Context Diagram15

3.3.2. Sequence Diagram15

3.3.3. Use Case Diagram16

**4. System Architecture - Frontend16**

4.1. Objectives16

4.2. Subcomponents16

4.2.1. 로그인16

4.2.1.1. Attributes16

4.2.1.2. Methods16

4.2.1.3. Class Diagram17

4.2.1.4. Sequence Diagram17

4.2.2. 메인 페이지17

4.2.2.1. Attributes18

4.2.2.2. Methods18

4.2.2.3. Class Diagram18

4.2.2.4. Sequence Diagram19

4.2.3. 강의 상세 페이지19

4.2.3.1. Attributes19

4.2.3.2. Methods19

4.2.3.3. Class Diagram20

4.2.3.4. Sequence Diagram20

4.2.4. 실시간 강의21

4.2.4.1. Attributes21

4.2.4.2. Methods21

4.2.4.3. Class Diagram21

4.2.4.4. Sequence Diagram22

4.2.5. 익명 게시판22

4.2.5.1. Attributes22

4.2.5.2. Methods22

4.2.5.3. Class Diagram23

4.2.5.4. Sequence Diagram24

4.2.6. 채팅방24

4.2.6.1. Attributes24

4.2.6.2. Methods24

4.2.6.3. Class Diagram25

4.2.6.4. Sequence Diagram26

4.2.7. 할 일26

4.2.7.1. Attributes26

4.2.7.2. Methods26

4.2.7.3. Class Diagram27

4.2.7.4. Sequence Diagram28

4.2.8. 캘린더28

4.2.8.1. Attributes28

4.2.8.2. Methods28

4.2.8.3. Class Diagram29

4.2.8.4. Sequence Diagram30

**5. System Architecture - Backend30**

5.1. Objectives30

5.2. Overall Architecture30

5.3. Subcomponents31

5.3.1. Application Server31

5.3.1.1. Class Diagram31

5.3.2. 강의 시스템32

5.3.2.1. Class Diagram32

5.3.2.2. Sequence Diagram33

5.3.3. 익명 게시판 & 채팅방 시스템33

5.3.3.1. Class Diagram33

5.3.3.2. Sequence Diagram34

5.3.4. 알림 시스템35

5.3.4.1. Class Diagram35

5.3.4.2. Sequence Diagram36

5.3.5. 실시간 강의 시스템37

5.3.5.1. Class Diagram37

5.3.5.2. Sequence Diagram38

5.3.6. 할 일 & 캘린더 시스템39

5.3.6.1. Class Diagram39

5.3.6.2. Sequence Diagram40

**6. Testing Plan40**

6.1. Objectives40

6.2. Testing Policy40

6.2.1. Development Testing40

6.2.1.1. Performance41

6.2.1.2. Reliability41

6.2.1.3. Security41

6.2.2. Release Testing41

6.2.3. User Testing41

6.2.4. Testing Case41

**7. Development Plan42**

7.1. Objectives42

7.2. Frontend Environment42

7.2.1. Android Studio42

7.2.2. Adobe Photoshop43

7.2.3. Xcode43

7.3. Backend Environment43

7.3.1. Github43

7.3.2. GLS 인트라넷44

7.4. Constraints44

7.5. Assumptions and Dependencies45

1. **Preface**
   1. **Readership**

본 디자인 명세서의 독자는 킹고 러닝 프로젝트의 소프트웨어공학개론 12조 팀원들과 주된 이용자인 성균관대학교 교수, 학생 및 조교들로 상정한다. 자연어를 이용해 시스템을 표현하며, 도표를 활용해 요구사항을 구체화 한다.

* 1. **Scope**

본 디자인 명세서는, 킹고 러닝 프로젝트에 필요한 정보들을 제공해 준다. 이 정보에는 시스템 아키텍처, 테스트 케이스, 개발 계획 등이 포함된다.

* 1. **Purpose**

본 디자인 명세서의 목적은, 킹고 러닝 프로젝트의 전체적인 윤곽과, 기술적인 설명을 제공해 본 프로젝트의 가이드라인이 되기 위함이다. 본 디자인 명세서는 이전에 작성한 요구사항 명세서를 구체화하며, 프로젝트를 진행하며 바탕이 될 시스템 아키텍처, 프론트엔드와 백엔드의 아키텍처, 진행할 테스트들에 대해 설명해 준다.

* 1. **Document Structure**

1. Preface  
   해당 장에선, 본 디자인 명세서의 의도를 알려주고, 예상 독자를 상정하며, 다루게 될 정보들을 간략히 소개한다. 또한 문서의 전체적인 구조와 각 장에 대한 간단한 설명을 통해 전체적인 윤곽을 보여준다.
2. Introduction  
   해당 장에선, 본 디자인 명세서의 도표를 그릴 때 활용한 도구들을 설명한다. 또한 이 프로젝트의 범위를 명료화 하며, 참고한 레퍼런스를 기록한다.
3. Overall System Architecture  
   해당 장에선, 전체적인 시스템의 개요를 context diagram, sequence diagram, use case diagram 을 활용해 표현한다.
4. System Architecture – Frontend  
   해당 장에선, Frontend의 아키텍처를 class diagram, sequence diagram 을 활용해 표현한다.
5. System Architecture – Backend  
   해당 장에선, Backend의 아키텍처를 class diagram, sequence diagram을 활용해 표현한다.
6. Testing Plan  
   해당 장에서 시스템이 요구사항에 맞게 잘 작동하는지 확인하는 방법이 기술된다. 관점과 시기에 따라 구분되는 Development testing, Release testing, User testing 이 세가지 테스트를 테스트 하는 방법과, 테스트케이스를 통해 설명한다.
7. Development Plan  
   해당 장은, 디자인 이후에 진행될 개발에 대한 계획을 설명한다. 개발을 하며 필요한 환경, 인지해야 하는 제한 사항들과 규칙에 대해 기술된다.
8. **Introduction**
   1. **Objectives**

Introduction 장은 본 디자인 명세서에 활용된 도표의 종류와, 그 도표를 그릴 때 사용한 도구들에 대한 설명이다.

* 1. **Applied Diagrams**
     1. **UML**

통합 모델링 언어(Unified Modeling Language)는 소프트웨어 공학에서 시스템의 디자인을 시각적으로 보여주기 위해 사용되는 표준화된 모델링 언어이다. 통합 모델링 언어는 특정 작업, 시스템을 구성하는 특정 요소와 이 요소들 끼리 어떻게 상호작용하는지 등의 내용이 포함된 도표를 통해 시스템의 전체적인 구조를 보여준다. UML Diagram에는 두가지 종류가 있는데, 하나는 class diagram 등이 포함된 Structural UML Diagram이고, 다른 하나는 use case diagram, sequence diagram, state diagram 등이 포함된 Behavioral UML Diagram이다.

* + 1. **Use Case Diagram**

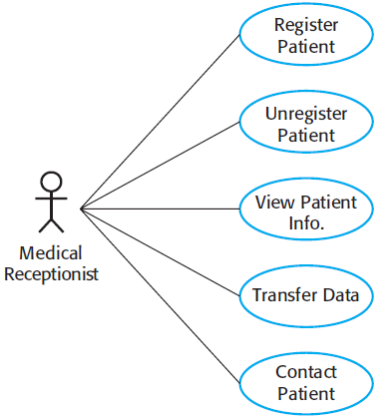


Figure 1. use case diagram

Use Case Diagram은 사용자와 시스템 간의 가능한 상호작용들을 시각적으로 묘사한 도표이다. Use Case Diagram은 시스템의 다양한 use case와 여러 종류의 시스템의 사용자를 보여 주는데, 때로는 다른 종류의 diagram을 함께 사용하며 설명하기도 한다. 이 use case들은 주로 원이나 타원으로 표현되고, 이 때의 actor들은 막대를 이용해 그려진다. Use case 만으로는 시스템과 사용자 간에 가능한 모든 경우에 대한 자세한 정보를 알려주기 힘든데, 이 때 Use Case Diagram을 활용해 시스템에 대한 더 자세한 정보를 알려주는데 활용한다.

* + 1. **Sequence Diagram**

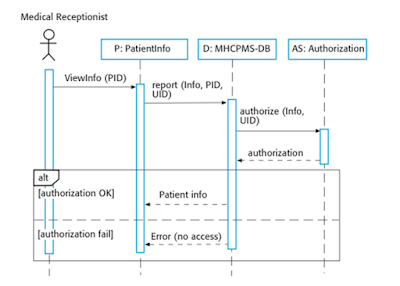


Figure 2. sequence diagram

Sequence Diagram은 시간에 따른 객체 간의 상호작용을 시각화한 도표이다. Sequence Diagram은 시스템의 기능을 실행하기 위해 필요한 객체들과, 그들의 흐름, 또 그 사이의 상호작용을 설명해 준다. 이 도표는 객체나 프로세스들을 세로 선으로 표현하고, 그들 사이에 상호작용을 가로 화살표로 표현한다. 이 도표는 런타임 중일 때의 시스템의 흐름을 한 눈에 보기에 유용하다.

* + 1. **State Diagram**

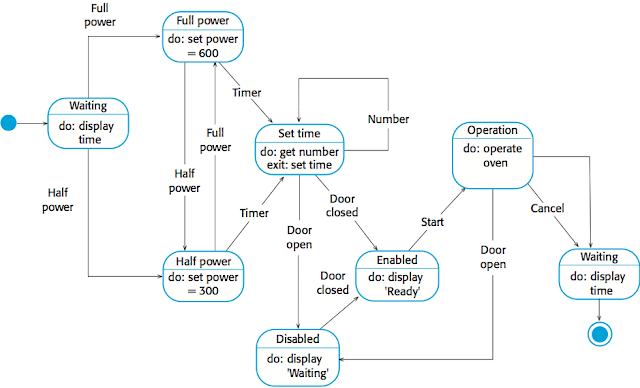


Figure 3. state diagram

State Diagram은 하나 이상의 state에서 일어날 수 있는 event의 연쇄를 묘사하기 위해 사용된다.이 도표는 시스템의 state을 정점으로, event를 간선으로 하는 방향 그래프의 형태로 그려진다. 이 도표는 외부 자극에 대한 시스템의 반응을 한눈에 보여줘서 real time system을 모델링할 때 자주 이용된다.

* + 1. **Class Diagram**

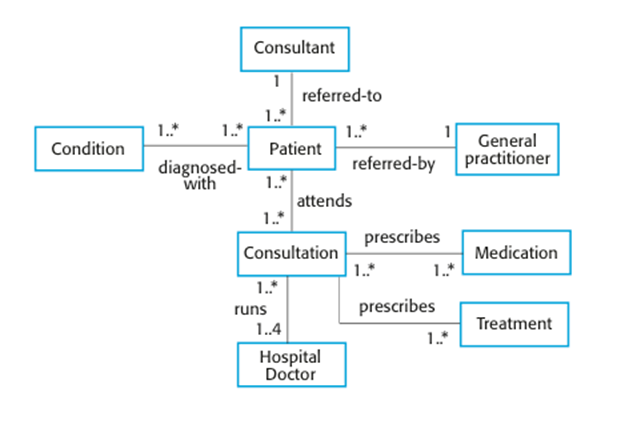


Figure 4. classes and association

Class Diagram은 시스템의 구조와 그들의 동작을 시스템의 class들을 보여줌으로서 설명하는 도표이다. Object Oriented Modeling에서 활용할 수 있는데, 시스템의 구조에 대한 개념적인 틀을 만드는데 사용이 된다. Class는 시스템의 객체라고 생각해도 되는데, 개발 초기 단계에서 실제 세상에서 존재하는 무언가를 본 떠 만든다.

**2.2.6. Context Diagram**

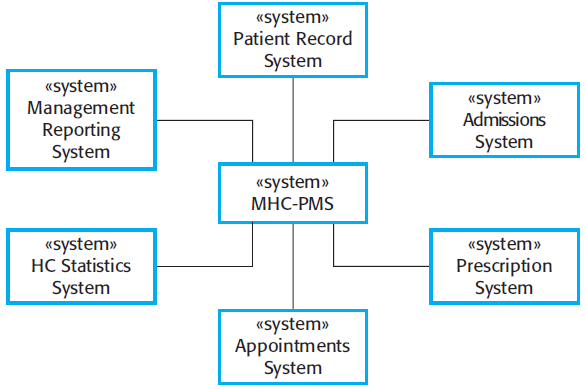


Figure 5. context diagram

Context Diagram은 시스템의 범위를 보여 주는 도표이다. 범위 뿐만 아니라, 요소들 끼리 어떻게 상호작용하는지 볼 수 있어, 시스템에 대한 높은 수준의 이해를 도울 수 있다. 가운데에 시스템을 대표하는 요소와, 그 주변에 외부적인 요소를 둬서 선으로 그들 간의 관계를 표현한다.

* 1. **Applied Tools**

**2.3.1. 파워포인트**



사무용 발표 자료 제작 도구이지만, 여러 도형과 마그넷 기능 들을 제공하여 도표를 만들기 편리하다.

**2.3.2. diagrams.net**



웹 기반의 도표 제작 도구이다.

* 1. **Project Scope**

기존의 온라인 강의 사이트는, 그냥 동영상 재생 사이트와 다름없었다. 킹고 러닝은 과제의 존재를 기억나게 해주는 친구가 되기도 하고, 수업 전후 강의실의 북적거리는 소음이 되기도 하고, 강의실 그 자체가 되기도 하는 등 말 그대로 컴퓨터에 캠퍼스를 구현하고자 하는 시스템이다. 이 시스템을 통해 전면 비대면 강의의 쉽게 녹화되어 반복해 볼 수 있다는 장점을 살리며, 다른 학생들과 단절된다는 단점을 보완해, 이전보다 더 나은 학습 경험을 제공해 줄 수 있다. iOS와 안드로이드 환경에서 실행되는 것을 가정하고, 지원하는 대표적인 기능은 다음과 같다.

* 녹화 강의 수강
* 실시간 강의 수강
* 성적 관리
* 구성원 간의 채팅
* 해야 할 일 한 눈에 보기
* 캘린더 관리
* 익명 게시판 지원

**2.5 References**

본 디자인 명세서를 작성하며 참고한 자료는 아래와 같다.

* Ian Sommerville, *Software Engineering*.
* IEEE Std 830-1998
* Skkuse, <https://github.com/skkuse>
* Wikipedia

1. **System Architecture – Overall**
   1. **Objectives**

이 장에서는 시스템의 frontend design과 backend design을 포함한 전체적인 구조에 대해 설명하고 그림으로 보여줄 것이다.

* 1. **Reuse and Relationships to Other Products**
     1. **LearningX**

****

우리가 개발하려는 앱은 기본적으로 LearningX의 구조와 인터페이스를 가져온다. LearningX는 현재 성균관대학교에서 사용하고 있는 아이캠퍼스 앱이다. 하지만 복잡한 할 일 기능, 불필요한 캘린더 기능, 난잡한 알림 등의 여러 단점들이 존재한다. 우리는 이 기능들을 개선하고 여기에 덧붙여서 익명문의게시판, 실시간 강의 기능 등을 추가할 것이다.

* + 1. **HelloLMS**



HelloLMS는 서울과학기술대학교에서 사용하고 있는 아이캠퍼스 앱이다. 여기서 사용하고 있는 실시간 강의 연동 기능은 ZOOM의 바로 가기 기능까지만 지원한다. 우리는 여기에 덧붙여서 실시간 강의 기능을 아이캠퍼스 내에 아예 추가할 것이다.

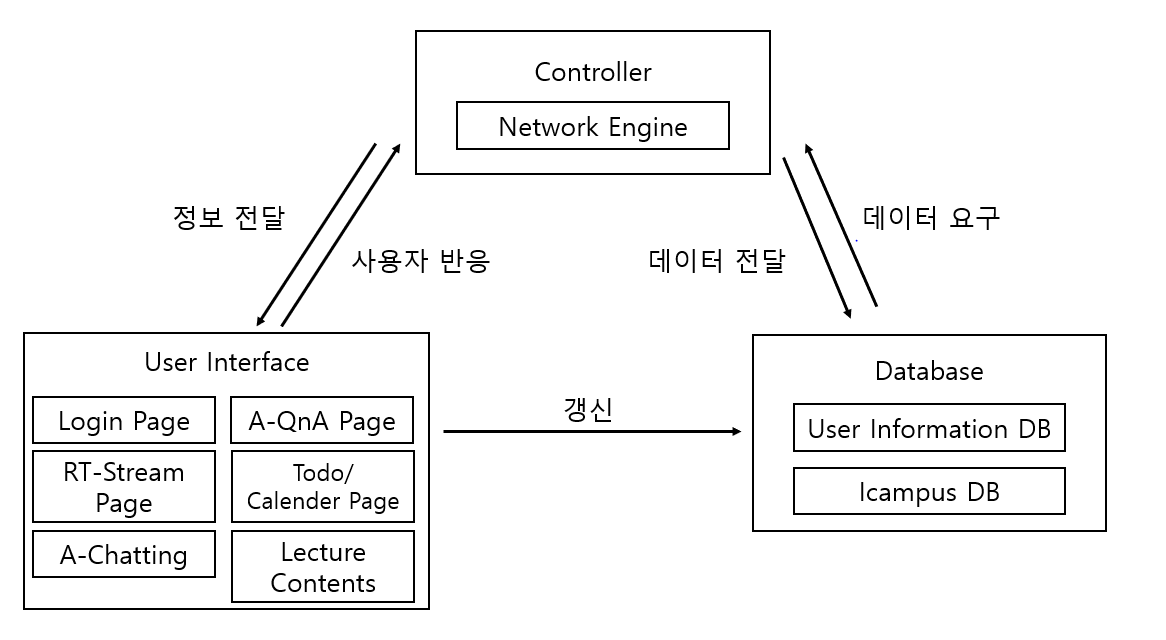
* + 1. **Coursemos**

****

Coursemos는 다양한 대학에서 사용하고 있는 아이캠퍼스 앱이다. 앱 내에서 대화 형식의 메시지를 보낼 수 있는 기능을 지원하고 있다. 우리는 이 기능을 본받아서 익명채팅방이라는 새로운 아이디어를 도출해 냈다.

* 1. **System Organization**

이 시스템은 기본적으로 client-server model 구조를 따르고 있다. Frontend는 user와의 상호 작용을 맡고 있고 HTTP 프로토콜을 이용하여 backend에 필요한 정보를 요청한다. Backend에서는 유저에 대한 정보와 유저가 앱 내에서 적은 내용 등을 저장하고 frontend에서 정보를 요청할 때마다 DB에서 내용을 받아와 전송해준다. 또한 backend에서 실시간 강의를 위한 데이터 공간을 마련하고 유저들에게 정보를 전달한다. 그 이외의 로그인 승인, 익명게시판 및 할 일/캘린더 내용 관리 등도 backend에서 전담한다.

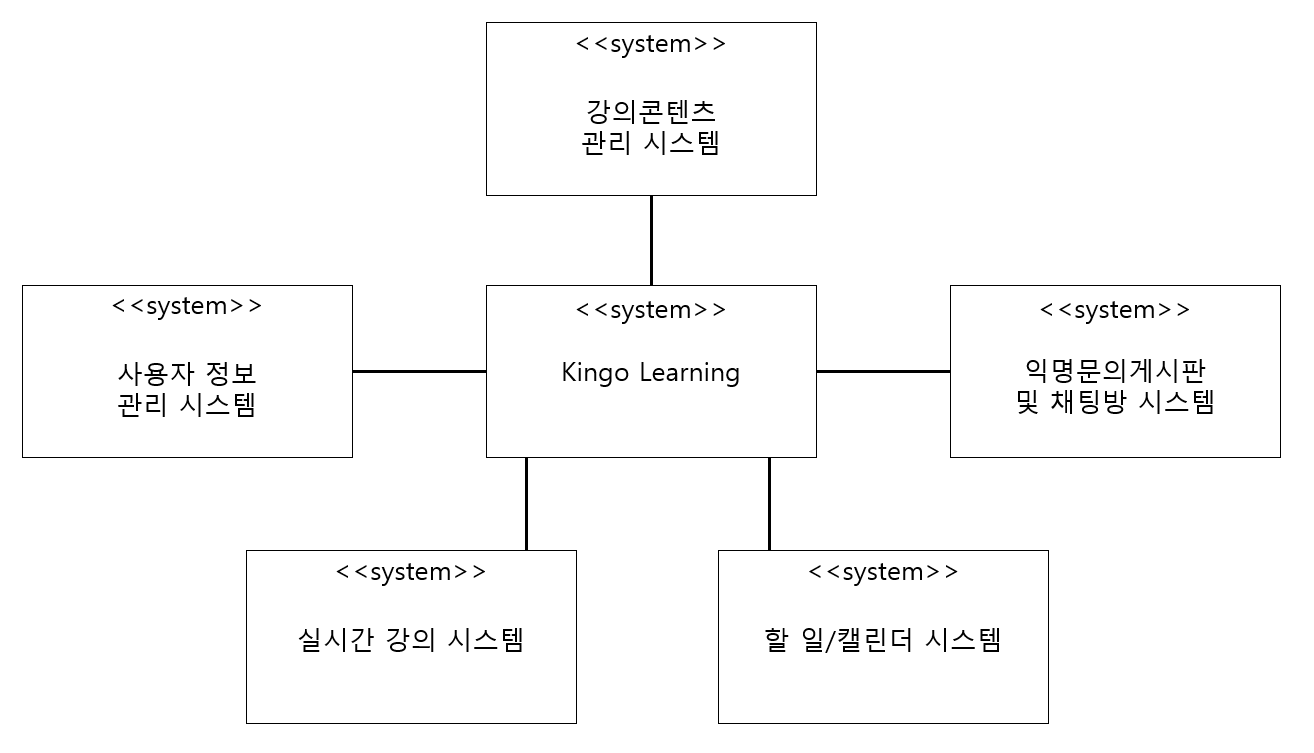
****

\* RT-Stream: Real-Time Stream (실시간 강의)

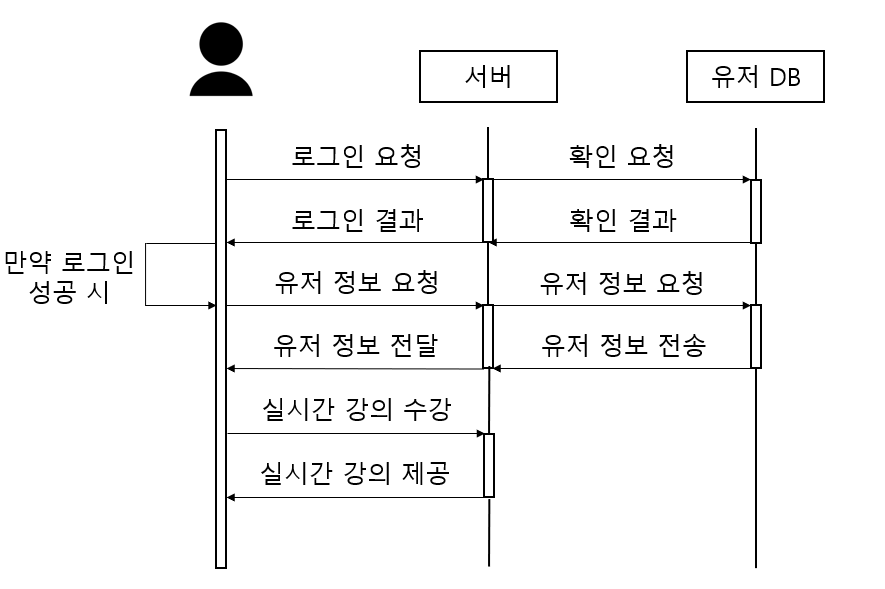
\* A-QnA: Anonymous-QnA (익명게시판)

\* A-Chatting: Anonymous Chatting(익명채팅방)

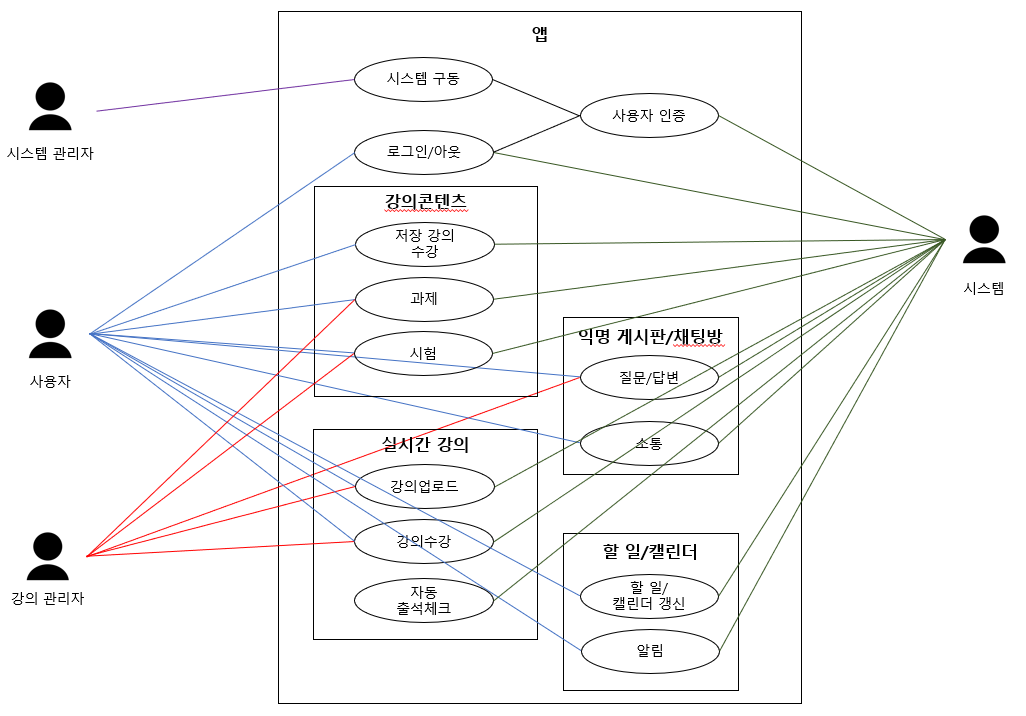
* + 1. **Context Diagram**



* + 1. **Sequence Diagram**



* + 1. **Use Case Diagram**

****

1. **System Architecture - Frontend**
   1. **Objectives**

이 장에서는 frontend system의 attributes와 methods을 설명하고 각 component의 관계를 class diagram과 sequence diagram을 통해 설명한다.

* 1. **Subcomponents**
     1. **로그인**

로그인 class에서는 전체 사용자의 로그인 세션을 관리한다. 자동 로그인을 허용한 사용자는 최초 로그인 이후 이 subcomponent를 거치지 않고 바로 4.2.2 강의 페이지로 이동한다.

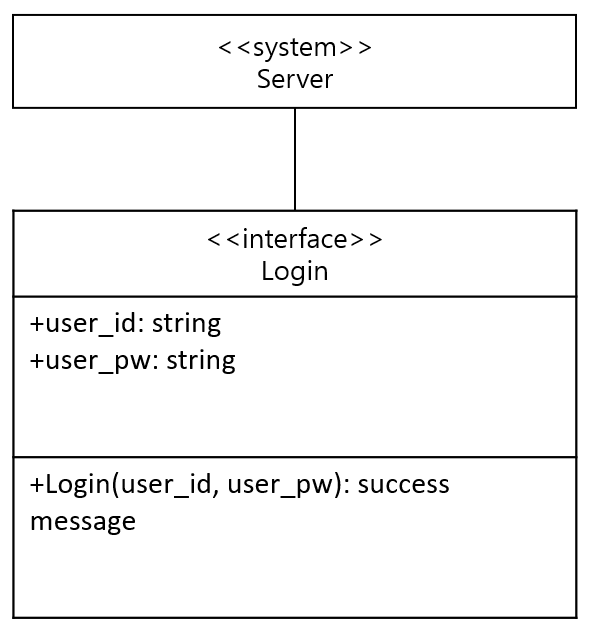
* + - 1. **Attributes**

input으로 받은 학번과 비밀번호를 바로 사용하기 때문에 별도의 로그인 객체를 따로 가지지 않는다.

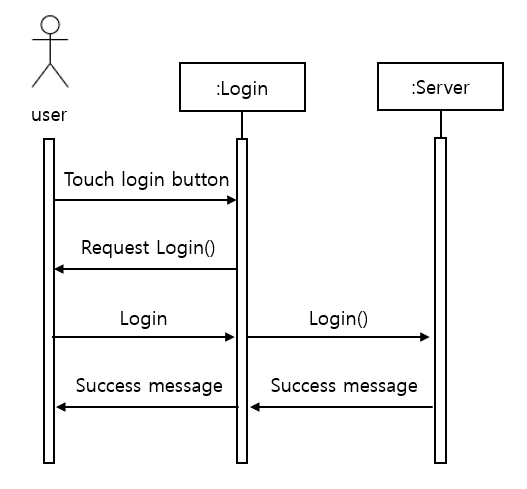
* + - 1. **Methods**

로그인 class가 가지는 methods는 다음과 같다.

* Login()
  + - 1. **Class Diagram**

****

* + - 1. **Sequence Diagram**



* + 1. **메인 페이지**

메인 페이지 class에서는 사용자의 수강 정보를 가져온다. 현재 수강중인 과목명을 리스트 형태로 보여주며 각 과목의 정보를 보여준다.

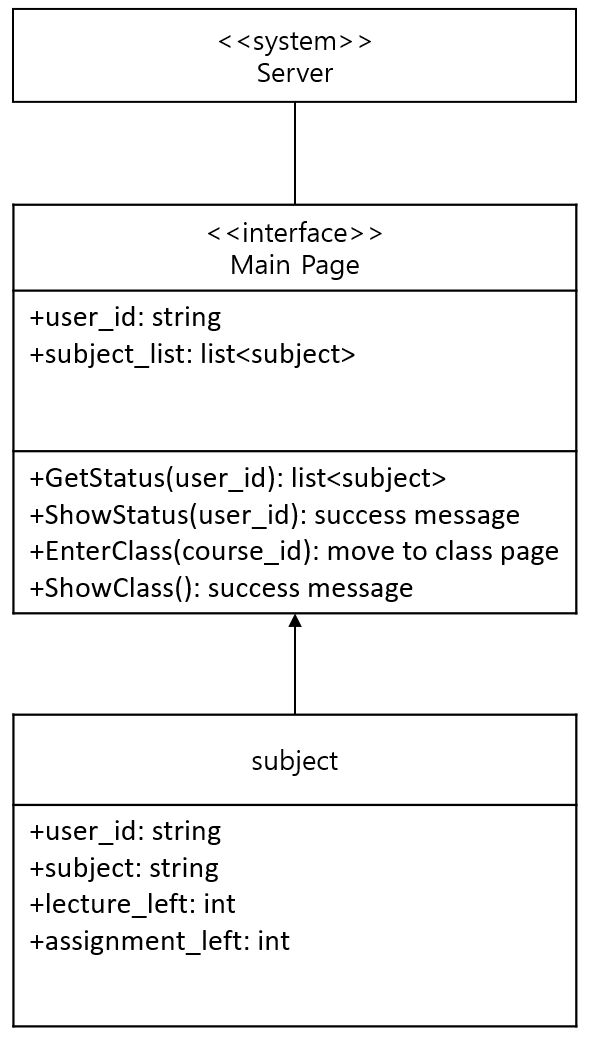
* + - 1. **Attributes**

메인 페이지 객체가 가지는 attributes는 다음과 같다.

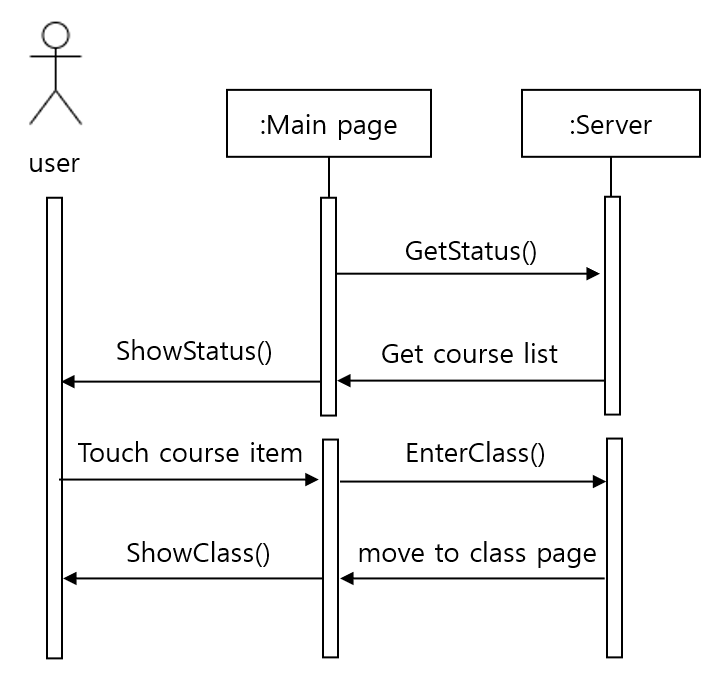
* User id: 사용자의 학번
* Subject: 과목명
* Lecture Left: 수강해야 하는 강의 수
* Assignment Left: 제출해야 하는 과제 수
  + - 1. **Methods**

메인 페이지 class가 가지는 methods는 다음과 같다.

* GetStatus()
* ShowStatus()
* EnterClass()
* ShowClass()
  + - 1. **Class Diagram**

****

* + - 1. **Sequence Diagram**

****

* + 1. **강의 상세 페이지**

강의 상세 페이지 class에서는 실시간 강의룸으로 들어갈 수 있는 버튼과 각 강의 메뉴의 항목을 리스트 형태로 보여준다. 이 항목은 공지, 수업자료실, 익명 게시판, 강의 컨텐츠, 과제, 시험 등으로 이루어져 있다. 사용자는 원하는 항목을 터치하여 각 페이지로 이동할 수 있다.

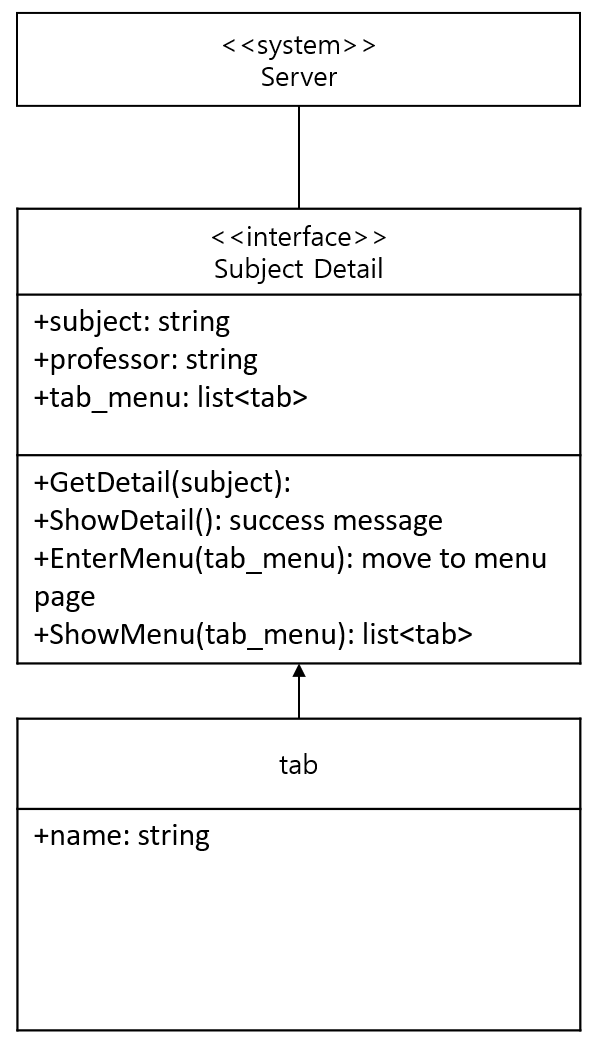
* + - 1. **Attributes**

강의 상세 페이지 객체가 가지는 attributes는 다음과 같다.

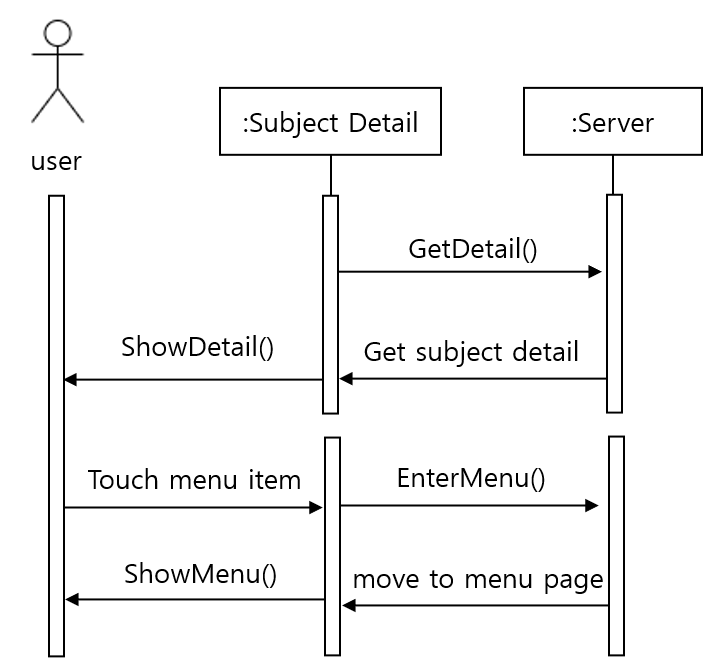
* name: 탭 메뉴의 이름
  + - 1. **Methods**

강의 상세 페이지 class가 가지는 methods는 다음과 같다.

* GetDetail()
* ShowDetail()
* EnterMenu()
* ShowMenu()
  + - 1. **Class Diagram**

****

* + - 1. **Sequence Diagram**

****

* + 1. **실시간 강의**

실시간 강의 class에서는 실시간으로 강의를 진행하거나 수강할 수 있다. 강의를 하는 사용자는 수강생의 출석 현황을 확인하고 카메라 및 화면 공유를 통해 강의를 할 수 있다. 수강을 하는 사용자는 강의자 화면을 볼 수 있고 채팅 기능을 이용할 수 있다.

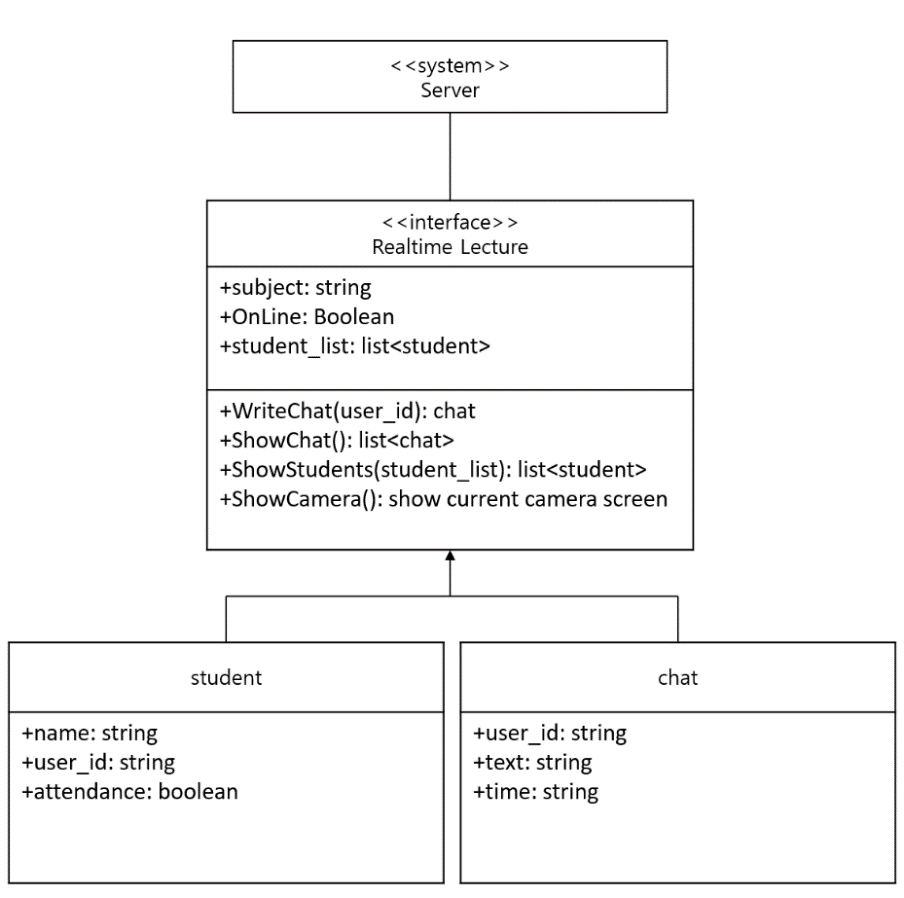
* + - 1. **Attributes**

실시간 강의 객체가 가지는 attributes는 다음과 같다.

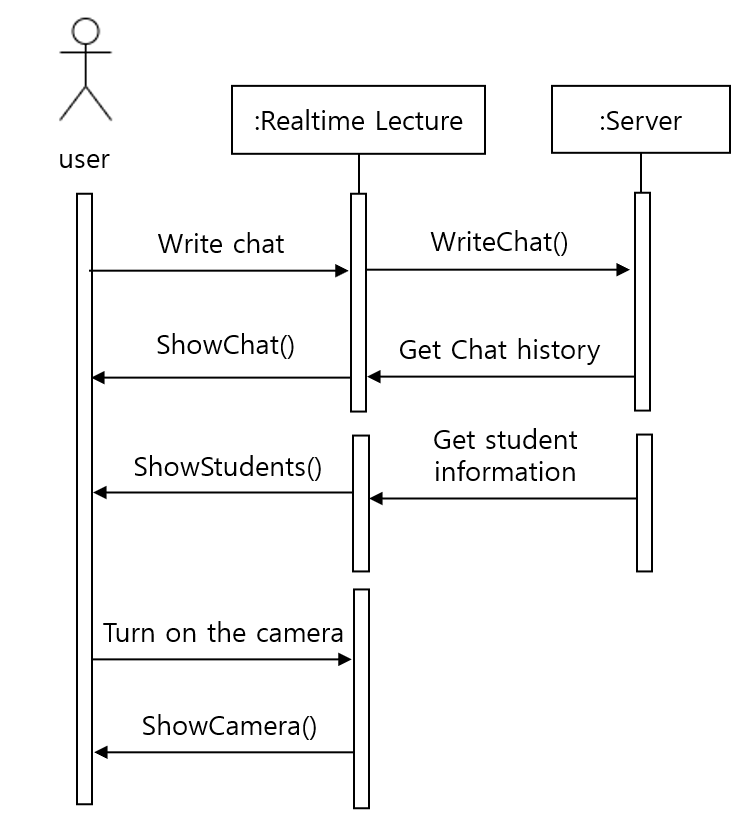
* User id: 사용자의 학번
* Name: 사용자의 이름
* Attendance: 출석 여부
* Text: 채팅창에 입력한 내용
* Time: 채팅을 입력한 시각
  + - 1. **Methods**

실시간 강의 class가 가지는 methods는 다음과 같다.

* WriteChat()
* ShowChat()
* ShowStudents()
* ShowCamera()
  + - 1. **Class Diagram**

****

* + - 1. **Sequence Diagram**

****

* + 1. **익명 게시판**

익명 게시판 class에서는 작성자 부분이 익명 처리된 사용자들이 글을 업로드할 수 있는 기능을 제공한다. 글 작성버튼을 누르면 사용자는 익명 게시판에 글을 업로드할 수 있다.

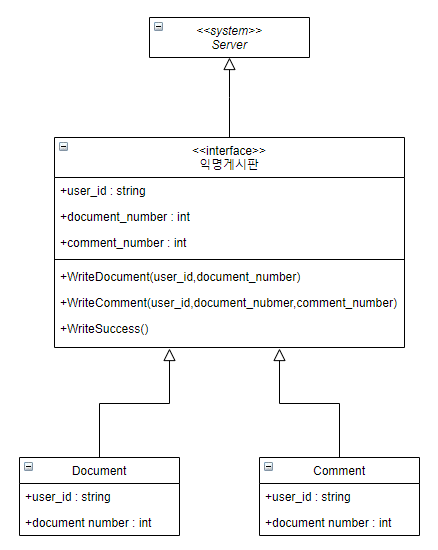
* + - 1. **Attributes**

익명 게시판 객체가 가지는 attributes는 다음과 같다.

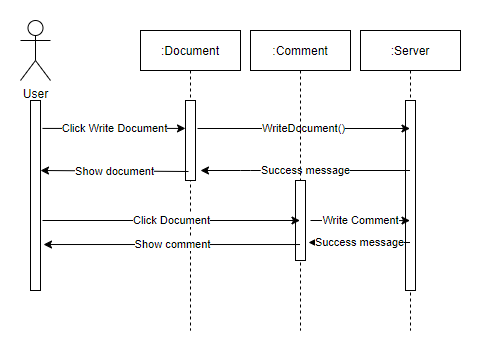
* User id: 사용자의 학번
* Document number : 게시판 글의 번호
* Comment number : 댓글 번호
  + - 1. **Methods**

익명 게시판 class가 가지는 methods는 다음과 같다.

* WriteDocument()
* WriteComment()
* WriteSuccess()
  + - 1. **Class Diagram**

****

* + - 1. **Sequence Diagram**

****

* + 1. **채팅방**

채팅방 class에서는 수강 중인 강의별로 익명 채팅방을 이용할 수 있다. 사용자가 채팅방을 클릭하면, 수강인원 모두 참가 중인 채팅방에 접속할 수 있다. 푸시 알림 on/off를 통해 새로운 채팅의 알림을 받을지 설정할 수 있다.

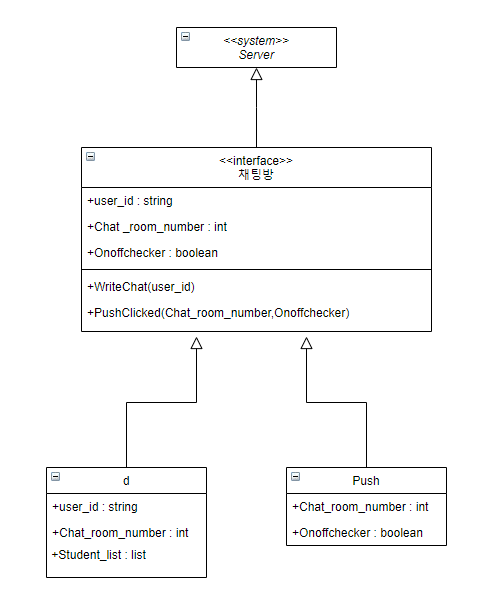
* + - 1. **Attributes**

채팅방 객체가 가지는 attributes는 다음과 같다.

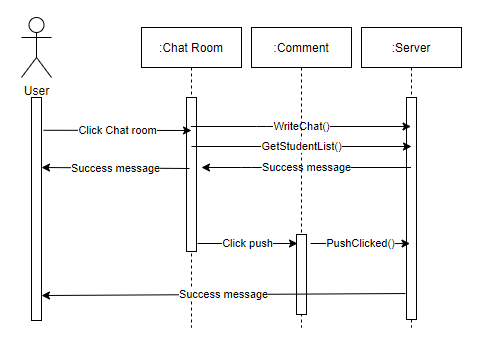
* User id: 사용자의 학번
* Chat room number : 강의의 채팅방 번호
* OnOffChecker : 푸시알림 체크
* Student\_list : 수강생 목록
  + - 1. **Methods**

채팅방 class가 가지는 methods는 다음과 같다.

* WriteChat()
* PushClicked()
* GetStudentList(subject)
  + - 1. **Class Diagram**



* + - 1. **Sequence Diagram**



* + 1. **할 일**

할 일 class에서는 사용자가 아직 수강하지 않은 과제 등을 표시한다. 할 일 리스트 중 항목을 클릭하면 관련된 강의 게시판과 연결된다. 사용자가 할 일을 완료할 시 그 할일은 완료로 표시된다.

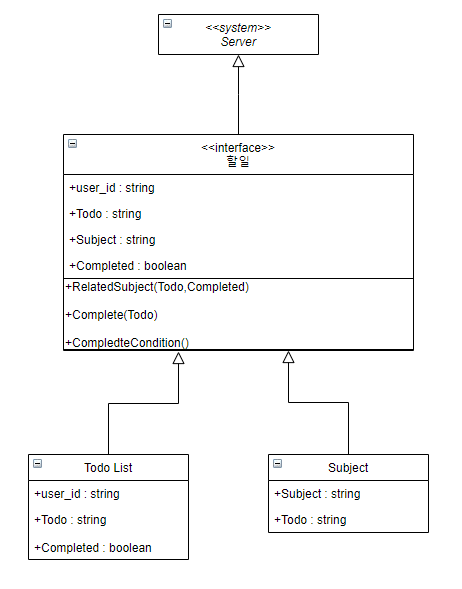
* + - 1. **Attributes**

할 일 객체가 가지는 attributes는 다음과 같다.

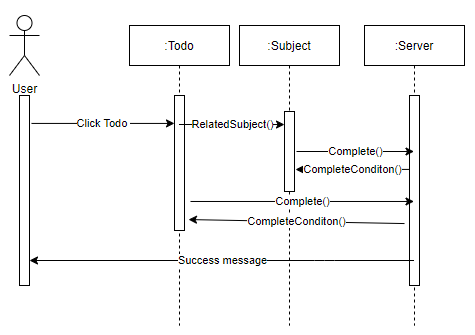
* User id: 사용자의 학번
* Todo: 할 일
* Subject : 과목명
* Completed : 완료여부
  + - 1. **Methods**

할 일 class가 가지는 methods는 다음과 같다.

* RelatedSubject()
* Complete()
* CompleteCondition()
  + - 1. **Class Diagram**

****

* + - 1. **Sequence Diagram**



* + 1. **캘린더**

캘린더 class에서는 사용자가 아직 완료하지 않은 과제나 수강하지 않은 강의 등을 표시한다. 사용자가 캘린더 버튼을 누르면 강의 일정을 달력에 표시한다. 사용자가 달력 특정 날을 클릭하면, 해당 날까지 해야 하는 작업을 확인할 수 있다. 사용자가 일정 추가 버튼을 누르면 자신만의 일정을 추가할 수 있다.

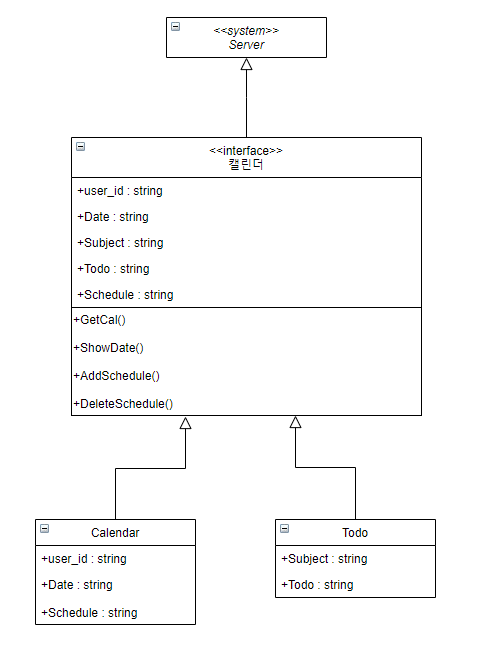
* + - 1. **Attributes**

캘린더 객체가 가지는 attributes는 다음과 같다.

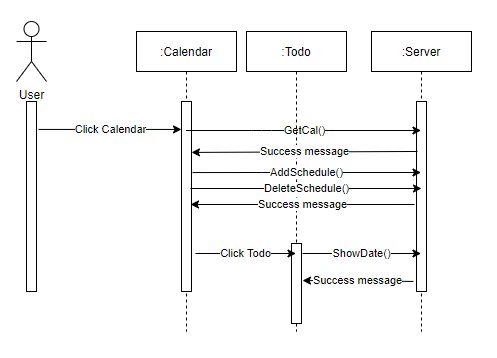
* User id: 사용자의 학번
* Date : 날짜
* Subject : 과목명
* Todo : 할일
* Schedule : 일정
  + - 1. **Methods**

캘린더 class가 가지는 methods는 다음과 같다.

* GetCal()
* ShowDate()
* AddSchedule()
* DeleteSchedule()
  + - 1. **Class Diagram**



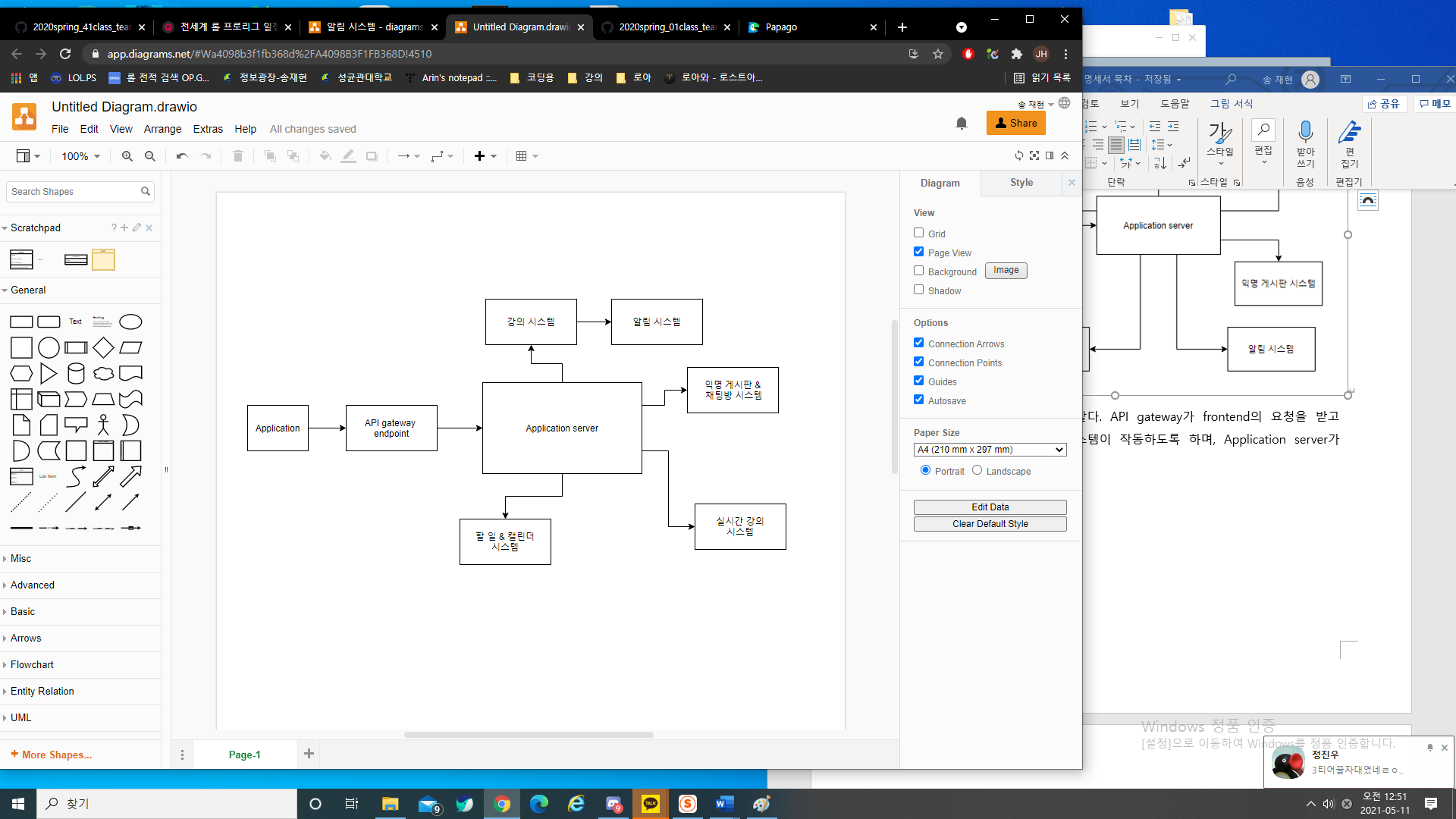
* + - 1. **Sequence Diagram**



1. **System architecture-Backend**
   1. **Objective**

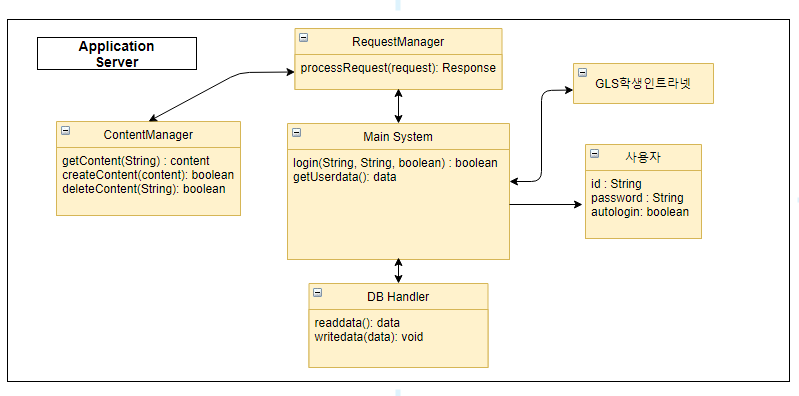
이 장에서는 시스템의 backend system의 구조를 설명할 것이다.

* 1. **Overall architecture**



전반적인 시스템의 구조는 위와 같습니다. API gateway가 frontend의 요청을 받고 application server에서 적절한 시스템이 작동하도록 하며, Application server가 controller의 역할을 수행합니다. 또한 Application server는 DB와 연결 되어있어 각 시스템이 필요로 하는 정보를 제공하고 변경된 정보들을 반영합니다.

* 1. **Subcomponents**
     1. **Application Server**
        1. **Class Diagram**



**RequestManager**

processRequest(request): API gateway로 부터 받은 요청을 처리하는 함수

**Main System**

login(String, String, boolean): 입력받은 id, password와 GLS학생인트라넷의 데이터를 비교하여 로그인이 성공했는지 확인하고 최초 로그인시 자동로그인 여부를 나타내는 boolean변수를 포함한 사용자 객체 생성

getUserdata(): GLS학생인트라넷으로부터 사용자의 데이터를 받아오는 함수

**ContentManager**

getContent(String): 컨텐츠 고유 ID를 입력받아 컨텐츠 내용을 반환하는 함수

createContent(content): 컨텐츠 객체 생성 함수

deleteContent(String): 컨텐츠 객체 삭제 함수

**DB Handler**

readdata(): 데이터 불러오기 함수

writedata(data): 데이터 작성 함수

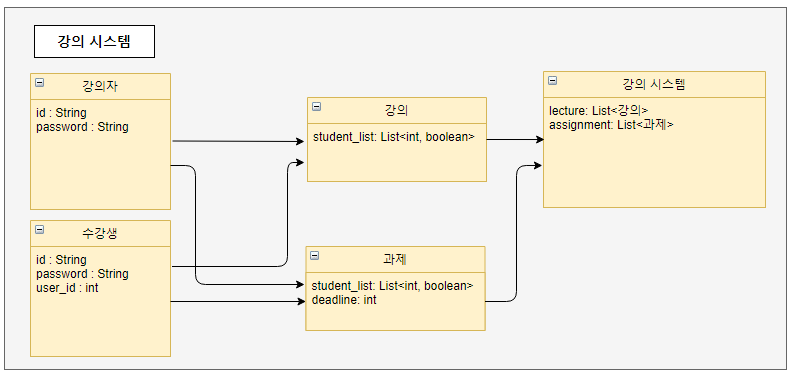
**사용자**

id: 사용자 id

password: 사용자 password

autologin: 사용자가 설정한 자동 로그인 여부

* + 1. **강의 시스템**
       1. **Class Diagram**



**강의자**

id: 강의자가 설정한 자신의 id

password: 강의자가 설정한 자신의 password

**수강생**

id: 수강생이 설정한 자신의 id

password: 수강생이 설정한 자신의 password

user\_id: 수강생의 학번

**강의**

student\_list: [학번, 출석여부]로 이루어진 list

**과제**

student\_list: [학번, 과제 제출 여부]로 이루어진 list

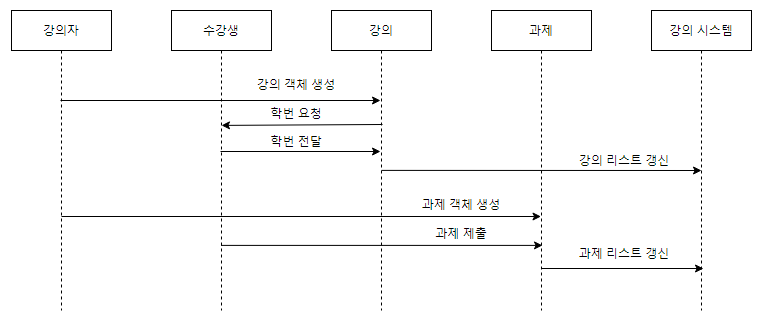
deadline: 과제 제출 기한

**강의 시스템**

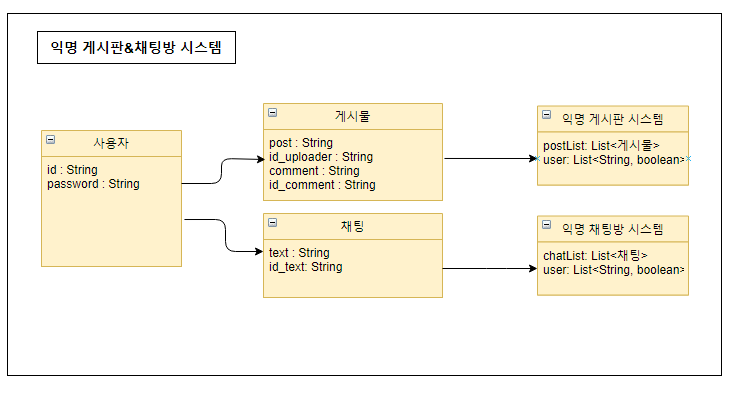
lecture: 강의 객체 리스트

assignment: 과제 객체 리스트

* + - 1. **Sequence Diagram**



* + 1. **익명 게시판 & 채팅방 시스템**
       1. **Class diagram**

****

**사용자**

id: 사용자가 설정한 자신의 id

password: 사용자가 설정한 자신의 password

**게시물**

post: 게시물 내용

id\_uploader: 게시물 작성자의 id

comment: 게시물 댓글 내용

id\_comment: 댓글 작성자의 id

**채팅**

text: 채팅 내용

id\_text: 채팅 작성자의 id

**익명 게시판 시스템**

postList: 게시물 객체 리스트

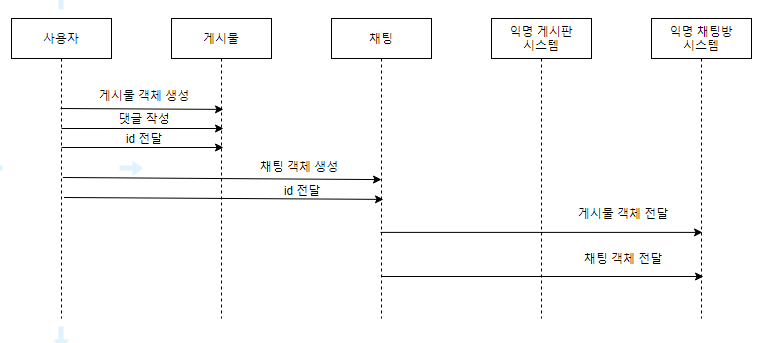
user: [사용자 id, 게시판 알림 수신 여부] 리스트

**익명 채팅방 시스템**

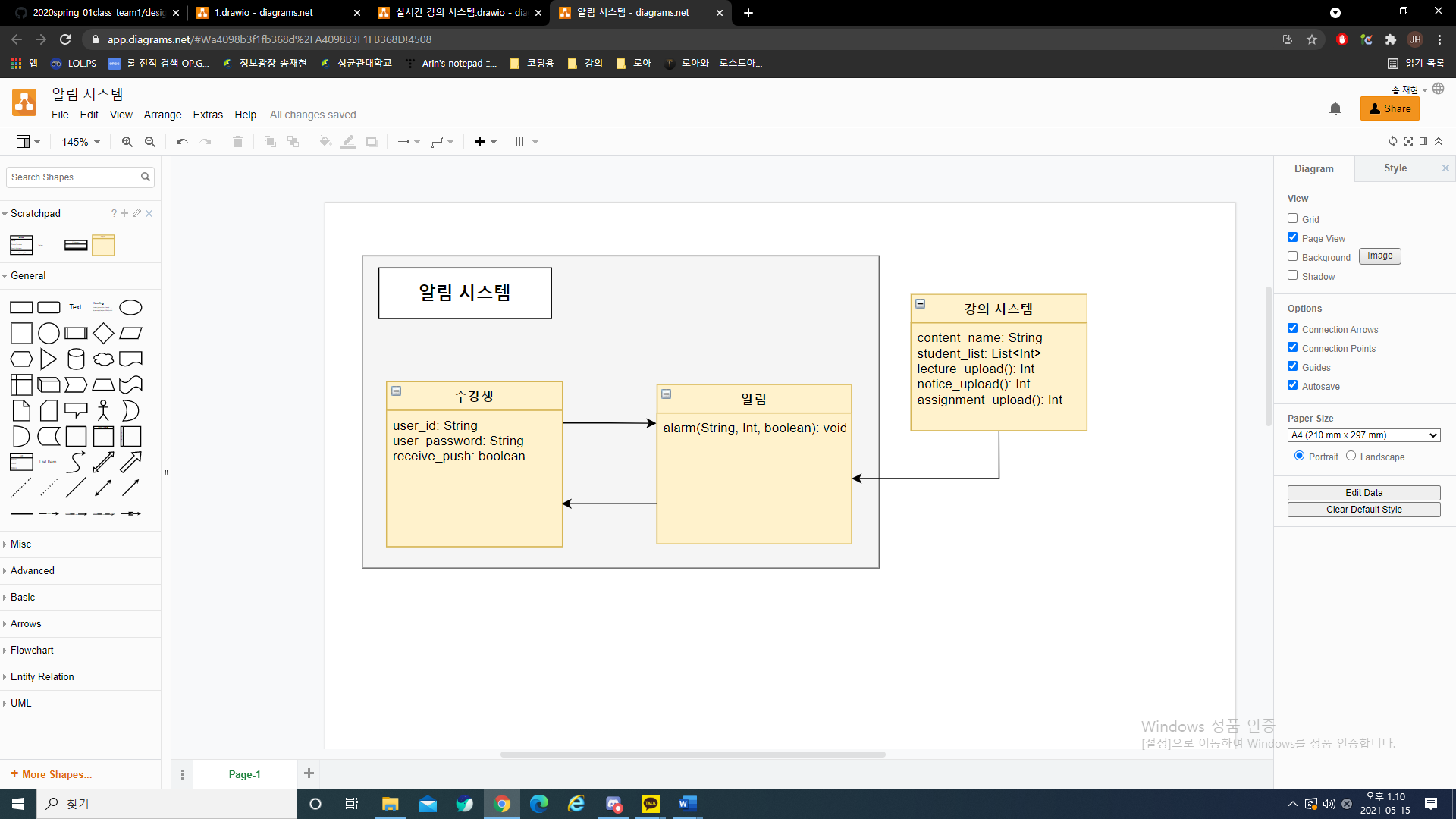
chatList: 채팅 객체 리스트

user: [사용자 id, 채팅방 알림 수신 여부] 리스트

* + - 1. **Sequence diagram**



* + 1. **알림 시스템**
       1. **Class diagram**



**수강생**

user\_id: 수강생의 학번

user\_password: 수강생이 설정한 자신의 password

receive\_push: 해당 수강생의 푸시 알림 수신 여부

**강의 시스템**

content\_name: 강의, 과제 혹은 공지의 제목. 알림으로 전달

student\_list: 해당 강의를 듣는 학생의 학번 list

lecture\_upload(): 강의가 업로드 된 경우, 해당 function이 작동하여 이에 맞는 int 변수를 알림으로 전달

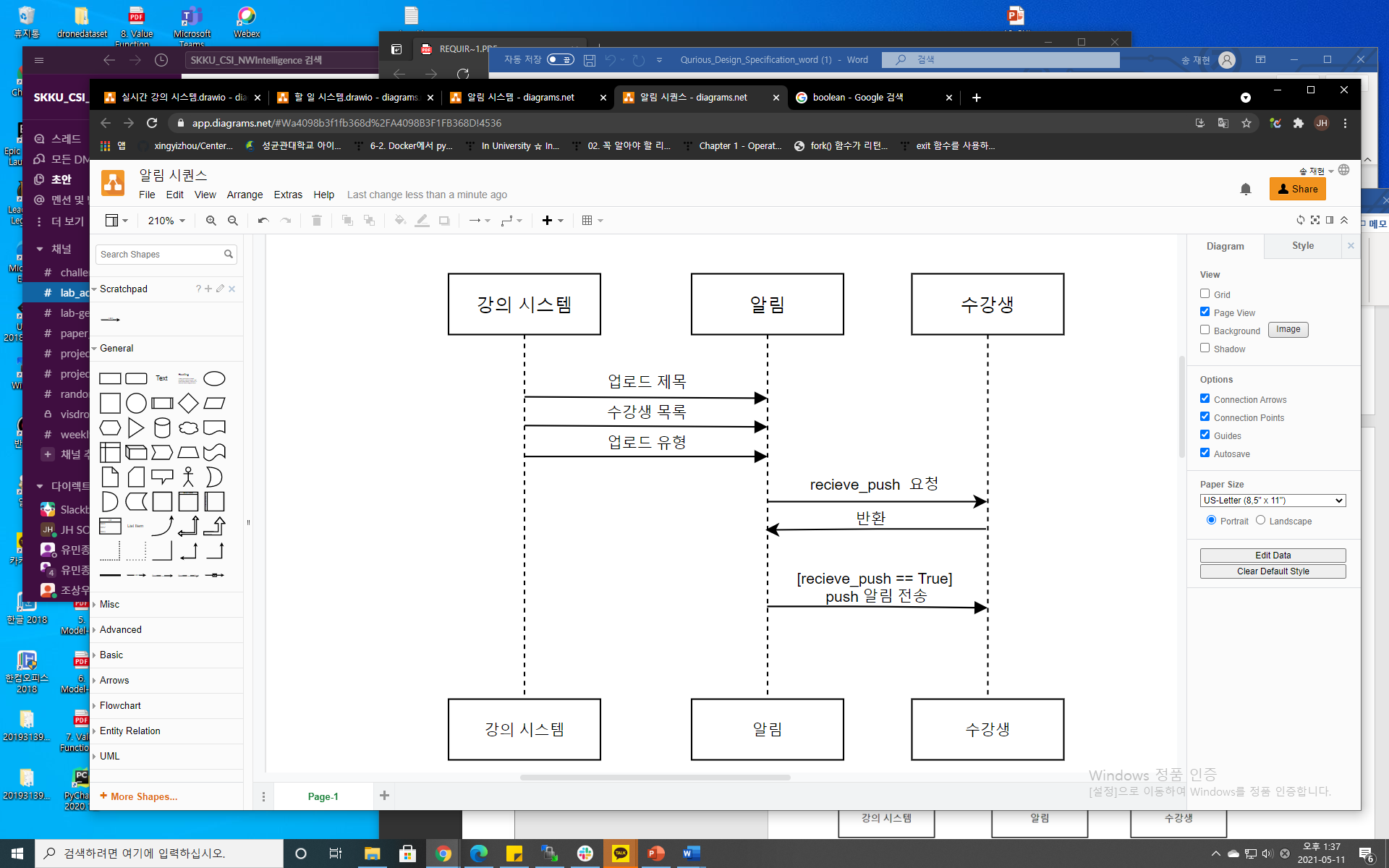
notice\_upload(): 공지가 업로드 된 경우로, 위와 동일하게 작동

assignment\_upload(): 과제가 업로드 된 경우로, 위와 동일하게 작동

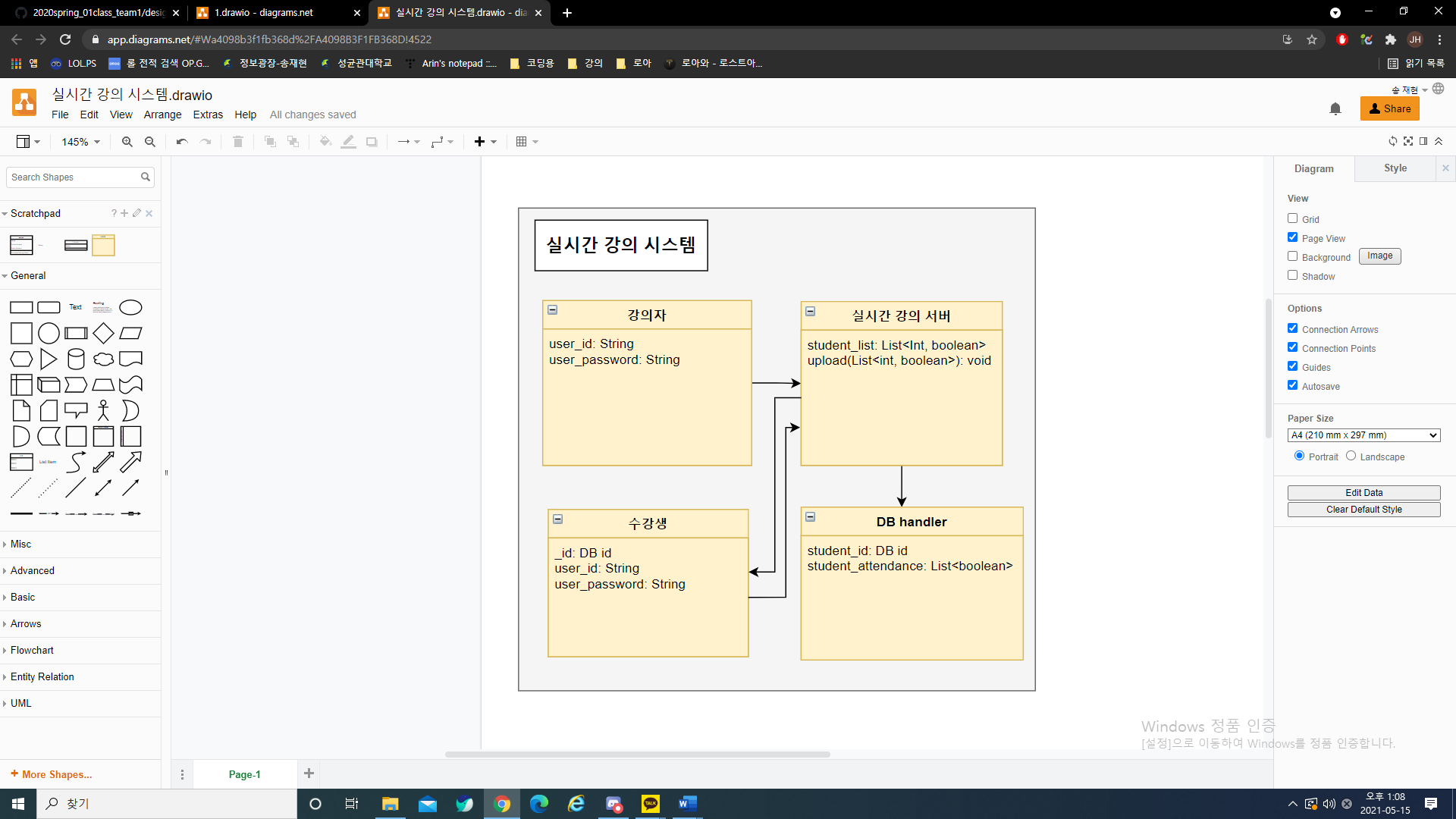
**알림**

alarm(String, Int, boolean): 수강생이 푸시 알림을 수신할 경우, 시스템에서 전달한 제목(String) 및 유형 식별(Int) 인자를 전달받고 알림을 학생에게 전송

* + - 1. **Sequence diagram**



* + 1. **실시간 강의 시스템**
       1. **Class diagram**



**강의자**

user\_id: 강의자가 설정한 자신의 id

user\_password: 강의자가 설정한 자신의 password

**수강생**

\_id: DB에 저장된 수강생 id

user\_id: 수강생의 학번

user\_password: 수강생이 설정한 자신의 password

**실시간 강의 서버**

student\_list: [학번, 출석여부]로 이루어진 list

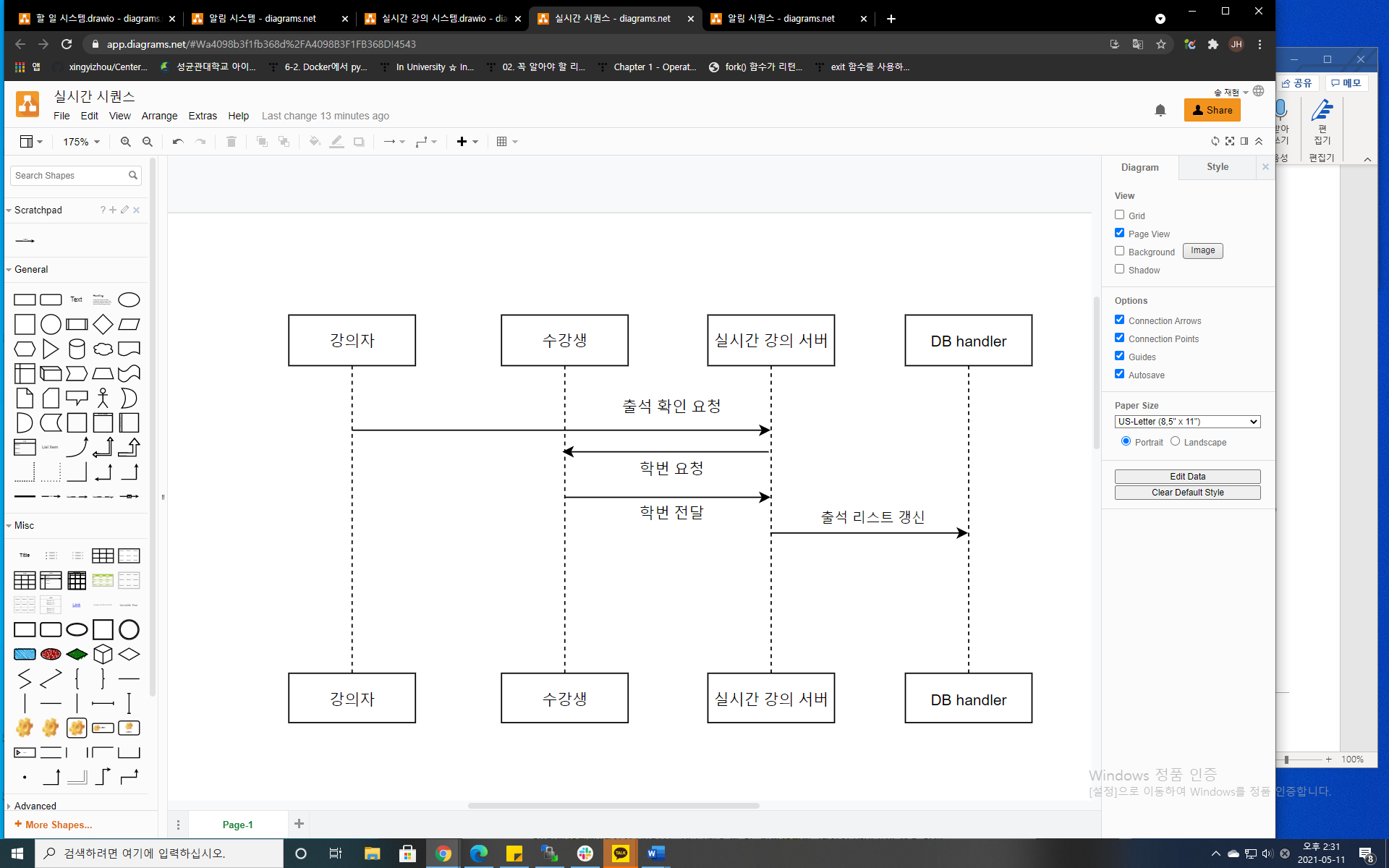
upload(<Int, boolean>): 수강생들의 출석 여부 리스트 갱신

**DB handler**

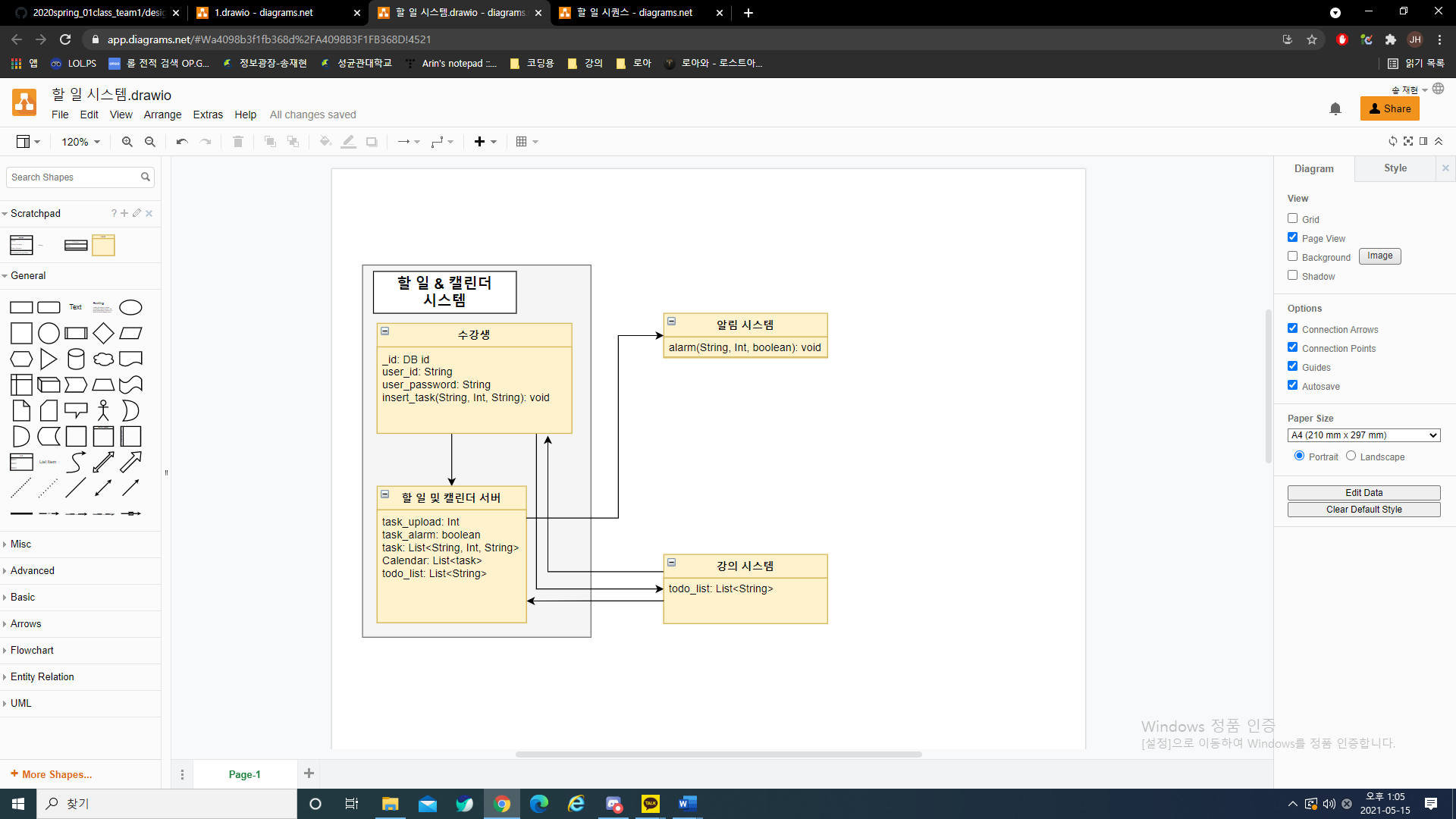
student\_id: DB에 저장된 수강생 id

student\_attandence: 해당 id 수강생의 출석 여부 기록

* + - 1. **Sequence diagram**



* + 1. **할 일 & 캘린더 시스템**
       1. **Class diagram**



**수강생**

\_id: DB에 저장된 수강생 id

user\_id: 수강생의 학번

user\_password: 수강생이 설정한 자신의 password

insert\_task(String, Int, String): 할 일의 제목, 날짜, 세부 설명을 전달

**할 일 및 캘린더 서버**

task\_upload: 알림 기능 사용 시, 알림 시스템으로 전달하는 캘린더임을 알리는 인자

task\_alarm: 알림 기능의 사용 여부

task: insert\_task를 통해 받은 정보를 저장

Calendar: 캘린더를 구성하는 위의 task로 이루어진 리스트

todo\_list: 강의 시스템에서 받아온 리스트

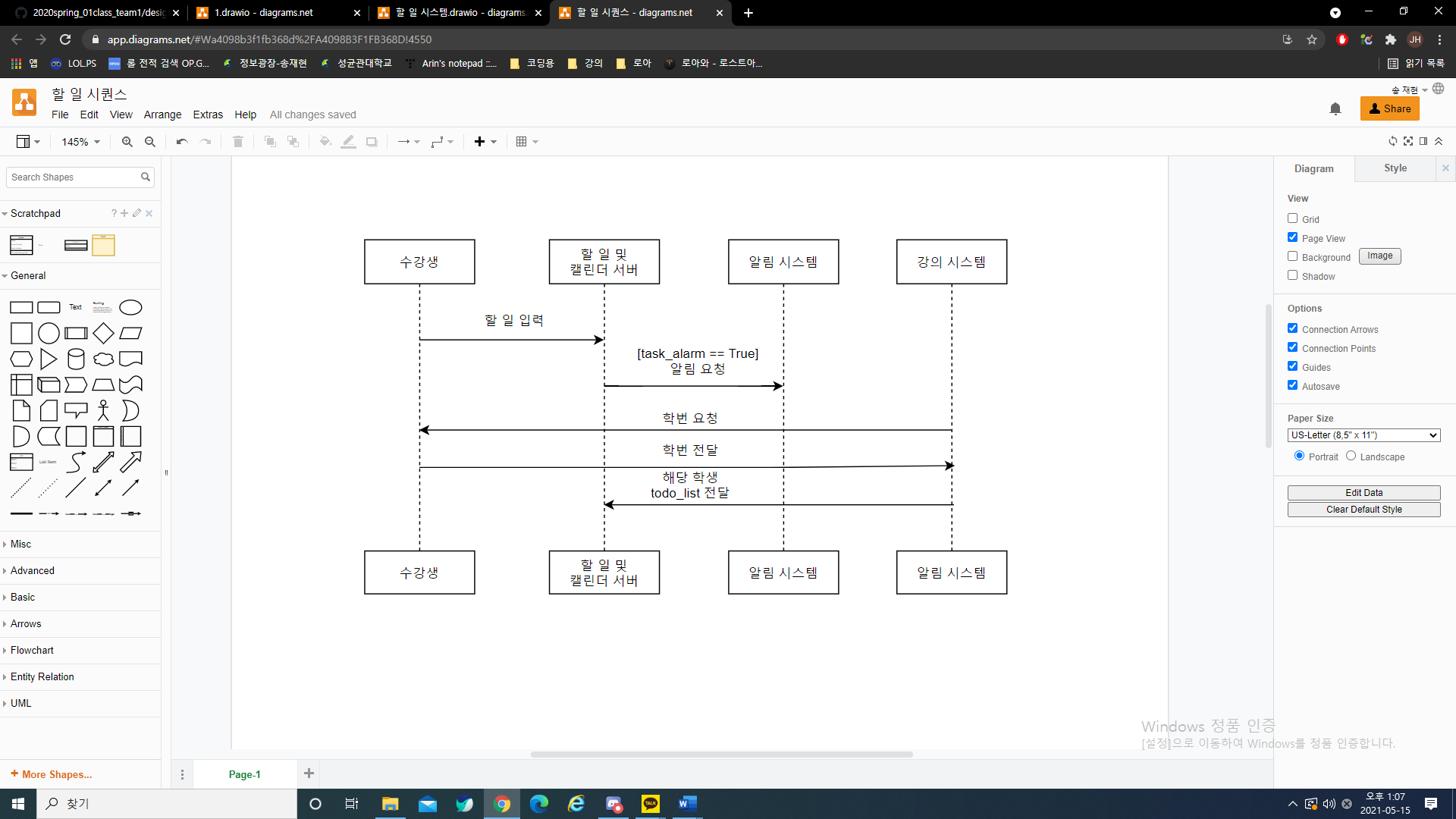
**알림 시스템**

alarm(String, Int, boolean): 수강생이 푸시 알림을 수신할 경우, 시스템에서 전달한 제목(String) 및 유형 식별(Int) 인자를 전달받고 알림을 학생에게 전송

**강의 시스템**

todo\_list: 수강생이 아직 완료하지 않은 강의, 과제 등으로 이루어진 리스트

* + - 1. **Sequence diagram**



1. **Testing Plan**
   1. **Objectives**

Testing Plan 장은 시스템이 의도대로 잘 작동하고 있는지, 고객의 요구사항을 잘 만족하고 있는지, 시스템이 안전한지 등을 검사할 수 있는 Test를 기획하기 위한 장이다. Test는 총 세가지, Development Testing, Release Testing, User Testing가 있다. 이 Test들에 대한 정책(Policy)과 Test Case를 설명한다.

* 1. **Testing Policy**
     1. **Development Testing**

Development Testing은 소프트웨어 엔지니어에 의해 시스템을 개발하는 단계에서 진행되는 Testing이다. 이 Testing은 다른 테스트를 하기 전에, 각각의 요소들이 의도대로 기능하는지 확인하고, 요소들을 통합할 때, 요소들이 충돌하며 생기는 문제는 없는지 찾기 위함이다. 이 방법으로 결과물의 질을 향상시키고 더 효율적인 개발을 진행할 수 있다.

* + - 1. **Performance**

통합된 시스템에서 각각의 요소들에 접근하는 데에 걸리는 시간은 너무 길어서는 안된다. 따라서 응답 시간에 4초의 제한을 둘 것이고, 4초가 넘는지를 Testing 할 것이다.

* + - 1. **Reliability**

킹고 러닝 시스템은 사용자에 성적에 영향을 주는 만큼, Reliability에서 특별히 엄밀한 Testing을 시행해야 할 필요가 있다. 각각의 요소들이 제대로 작동하는지도 Testing을 해보아야 하고, 통합된 시스템에서 요소들 끼리 충돌하며 뭔가를 누락시키는 등의 오류는 없는지 Testing을 거쳐야 한다.

* + - 1. **Security**

온라인 캠퍼스 시스템인 킹고 러닝은 자세한 개인 정보를 담고 있는 요소들이 많다. 우선 이 요소들 자체적으로 불필요하게 정보를 흘리고 다니지 않는지 확인을 해 보아야 한다. 또한 통합된 시스템에서 개인 정보를 접근할 필요가 없는 요소들이 개인 정보를 접근하고자 하는지 확인해 보아야 한다.

* + 1. **Release Testing**

Release Testing은 개발이 완료된 시스템을 배포하기 전에 시행하는 Testing이다. 개발을 맡은 팀 말고 다른 팀이 테스트를 진행하게 되는데, 그 목적은 이 정도면 출시할 만한지, 문제는 없는지 확인을 하는게 목적이다. 킹고 러닝 프로젝트의 경우 이 테스트를 위해 팀원 중 한 명을 미리 선발한다.

* + 1. **User Testing**

User Testing은 실제 사용자들과 함께 Testing을 진행하며 그들에게서 의견과 조언을 받는 단계이다. Alpha Testing, Beta Testing 등으로 구성할 수 있다. 킹고 러닝 프로젝트는 Alpha Testing을 진행해 20명의 사용자들에게 의견을 수집할 것이다.

* + 1. **Testing Case**

테스트 케이스는, 앞서 언급한 performance, reliability, security를 확인함과 더불어 기능적으로 작동을 잘 하는지 검사하도록 제작한다. 예시는 다음과 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Action | Test Case | Result |
| 응답 시간 | 1. 게시판에 글을 작성한다.  2. 글이 업로드 되기까지 시간을 측정한다. | 4초 이내이다. | SUCCESS |
| 4초 이상이다. | FAIL |
| 응답 시간 | 1. 강의를 수강한다.  2. 버퍼링 등 재생되지 않는 시간을 측정한다.  3. 강의의 길이에 대한 비율을 구한다. | 1% 이내이다. | SUCCESS |
| 1% 이상이다. | FAIL |
| 과제 제출 | 1. 과제를 제출한다.  2. 정상적으로 제출되는지 확인한다. | 제출된다. | SUCCESS |
| 누락된다. | FAIL |
| 녹화 강의 출석 체크 | 1. 녹화 강의를 끝까지 수강한다.  2. 출석으로 체크가 되었는지 확인한다. | 출석으로 뜬다. | SUCCESS |
| 누락된다. | FAIL |
| 실시간 강의 출석 체크 | 1. 수업 시작 시간 10분 이내로 접속한다.  2. 실시간 강의를 끝까지 수강한다.  3. 출석으로 체크가 되었는지 확인한다. | 출석으로 뜬다. | SUCCESS |
| 누락된다. | FAIL |
| 개인정보보안 | 1. 사용자로 로그인을 한다.  2. 시스템에 해당 사용자의 정보가 암호화되어 있는지 확인한다. | 암호화된다. | SUCCESS |
| 그렇지 않다. | FAIL |

1. **Development Plan**
   1. **Objectives**

이 장에서는 개발에 필요한 환경들과 기술들에 대해서 기술한다.

* 1. **Frontend Environment** 
     1. **Android Studio**



안드로이드 스튜디오는 안드로이드 OS를 환경으로 하는 앱을 개발하기 위해 사용하는 툴이다. 이것은 세 가지 종류의 컴퓨터 언어(Java, Kotlin, and C++)를 지원하고 우리는 Java를 이용한다. 또한, 이것은 눈에 띄고 알아보기 쉬운 코드 편집기를 지원하여 개발자들이 코드를 개발하기 더 쉽고 정확하게 만들어 준다. Frontend 개발에서 우리는 이것을 사용해서 UI 레이아웃들과 구조를 만든다.

* + 1. **Adobe Photoshop**

아이콘과 사진 편집을 위한 앱이다. Adobe 회사가 만들었고 Windows와 macOS에서 사용이 가능하다. 우리는 이걸 사용함으로써 눈에 띄고 앱과 어울리는 레이아웃들과 아이콘을 만들어서 앱 접근성을 향상시킬 것이다.

* + 1. **Xcode**



Xcode는 Android Studio와 다르게 IOS operating system을 개발 환경으로 하는 개발 툴이다. 모든 성균관대학교 학생들이 사용할 수 있도록 해야 하므로 Android OS뿐만 아니라 IOS도 지원하기 위해서 Xcode로 이것을 해결한다.

* 1. **Backend Environment** 
     1. **Github**

Github는 우리의 팀 프로젝트를 진행할 때 필요하다. 이것은 component들의 통합을 쉽게 하도록 도와준다. 또한 즉각적인 피드백을 서로에게 해줄 수가 있어서 소프트웨어의 질을 향상시켜준다.

* + 1. **GLS학생 인트라넷**

우리는 성균관대학교에서 제공하고 학생에 대한 정보를 담고 있는 데이터베이스인 GLS학생 인트라넷을 사용한다. 이것을 통해서 학생들의 수강중인 강의나 시간표 등을 가져오고 앱에서 사용할 수 있도록 한다.

* 1. **Constraints**

우리가 소프트웨어를 디자인할 때 여기에 적혀 있는 규칙들을 준수할 것이다. 또 다른 자세한 규칙들은 개발자가 개발하려는 방향과 관련되어 있지만 아래 사항들은 변하지 않을 것이다.

* 이미 증명된 기술들을 사용한다.
* 비용이 드는 기술들을 사용하는 것을 지양한다.
* 유저가 편하게 사용할 수 있는 방향으로 개발한다.
* 개발과 유지 비용을 디자인하면서 고려한다.
* 최대한 오픈 소스를 활용한다.
* 소프트웨어 성능의 향상을 도모하는 방향으로 디자인을 결정한다.
* 소프트웨어 코드를 최적화해야 한다.
* 후에 변화를 수용하기 위해서 소스 코드를 잘 설명하는 코멘트들을 적어 놓아야 한다.
  1. **Assumptions and Dependencies**

앞서 말했듯이 우리는 IOS와 Android OS, 두 운영체제에 지원되는 앱을 개발할 것이다. 그 의외에 운영체제에는 지원이 불가능하다는 것을 명시한다. 또한 Android OS는 6.0버전 이상이여야 지원이 되고 IOS는 13.0 버전 이상일 시 지원이 된다.