**팀 EAMS 프로젝트 설계 계획서**

**주제: 개선된 학사 관리 시스템**

**2조**

**학년/반: 3-A**

**팀명: EAMS(Enhanced Academic Management System)**

**프로젝트 참여자: 2106117 김태연, 2306007 김수혁, 2306008 김명훈**

**목차**

**1 개요 -1-**

* 1. **프로젝트 개요**
  2. **프로젝트 선정 동기**
  3. **프로젝트로 인한 효과**
  4. **프로젝트의 산출물**

**2 업무 분류 차트**

**3 개발 프로세스 및 소스코드 규칙**

**4. 개발 환경**

**4.1 개발 언어**

**4.2 개발 환경**

**5. 요구사항 분석**

**6. 전체 기능 요구사항 분석**

**7. 클래스 설계**

**8. ERD**

**9. DB테이블**

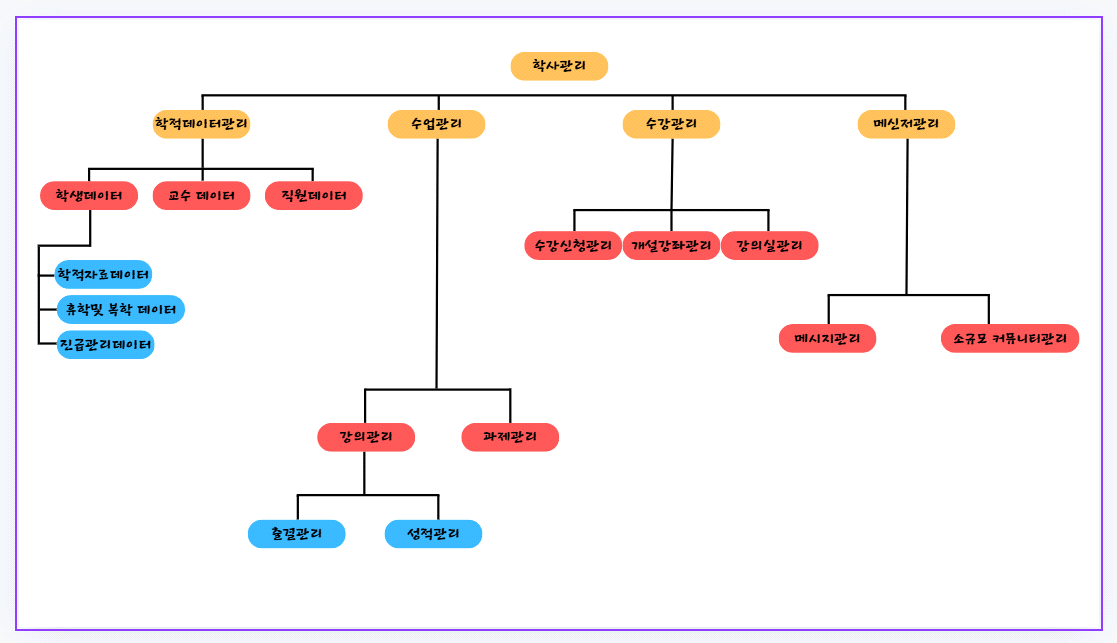
**10. 화면 기능 설명**

**11. 테스트 케이스**

**1. 개요**

|  |
| --- |
| **1.1 프로젝트 개요** |
| **해당 프로젝트는 기존의 학사 관리 시스템에서 발생하는 문제점들을 개선하여 새로운 학사 관리 시스템을 개발하는 팀 프로젝트이다.** |
| **1.2 프로젝트 선정 동기** |
| **현재 사용되는 학사관리 시스템의 기본적으로 두가지 문제점이 존재하며 첫** |
| **번째 문제점은 기능들이 어디에 위치해 있는지 찾는 것이 어렵다는 문제이** |
| **존재한다.** |
| **두 번째 문제점은 메신저의 부재로 학생들이나 교수들이 학사 관리 시스템안** |
| **에서 소통할 창고인 메신저가 굉장히 빈약하다는 문제점이 존재한다.** |
| **이러한 두가지 문제점은 충분히 개선할 수 있는 문제점이라고 봤기에 해당 프로젝트를 계획했다.** |
| **1.3 프로젝트로 인한 효과** |
| **첫 번째 문제점은 디자인 적인 부분에서 많은 개선이 필요한 문제로, 타 학교 포털 사이트와 여러 웹사이트의 디자인을 참고하여 기존 디자인보다 기능들의 위치를 더 직관적으로 배치하며, 또한 검색 기능에 연관 검색어 기능을 추가하여 사용자가 입력한 단어와 관련된 기능을 쉽게 확인할 수 있도록 함으로 원하는 기능의 위치를 보다 쉽게 찾을 수 있을 것을 기대한다.** |
| **현재 학사 관리 시스템에는 메신저 기능이 존재하지만, 사실상 거의 사용되지 않는 상황이다. 이에 따라, 단체 방 기능과 파일 업로드 기능 등 다양한 기능을 갖춘 새로운 메신저를 개발하고, 이를 학생들과 교수들이 자주 방문하는 포털 사이트의 메인 페이지에 배치한다면, 메신저를 더 자주 접하게 되어 사용 빈도가 증가할 것으로 기대된다.** |
| **1.4 프로젝트의 산출물** |
| **이 프로젝트는 웹사이트 기반의 프로젝트로 이 프로젝트 산출물은 웹사이트로만 작동하는 학사 관리 시스템과 그 학사 관리 시스템의 기능들이 어떤 식으로 동작하게 할 지 결정한 회의록이 본 프로젝트의 산출물로 해당 산출물들은 깃허브에 공개된다.** |

**2 업무 분류 차트**



**학사관리를 네 가지 작업으로 세분화 시켰다.**

**학적 데이터관리는 실제로 데이터가 들어가는 작업들을 나타낸다.**

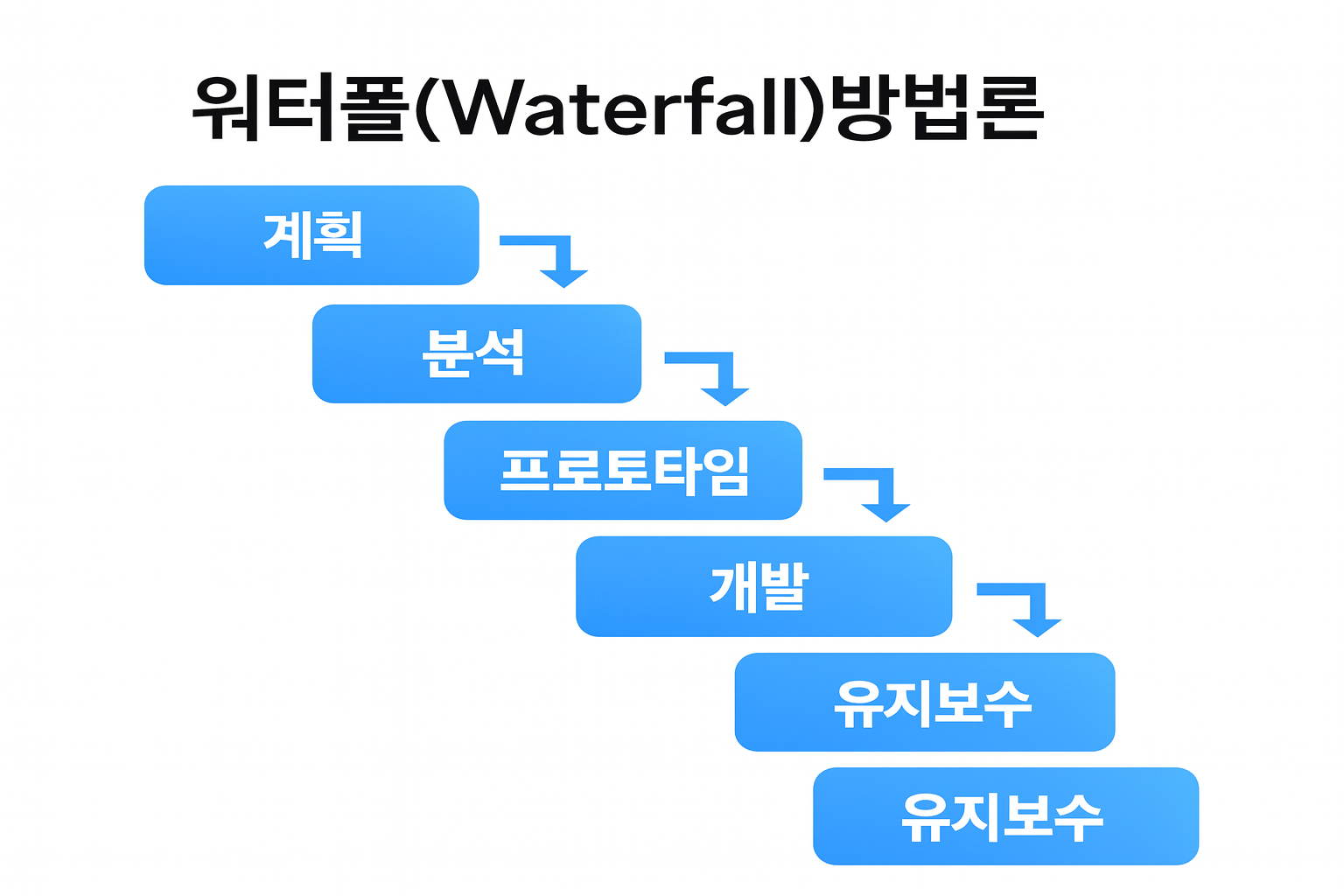
**실제로 관리할 데이터들은 세가지로 학생 데이터, 교수 데이터, 직원 데이터 세가지이기에 학생 데이터 관리, 교수 데이터 관리, 직원 데이터 관리 이렇게 세가지의 기능으로 분리한다.**

**학생데이터 관리에는 학생 인적 사항 그리고 학적 기록이 관리할 수 있도록 하며**

**복학 및 휴학 처리 기능과 특정 조건이 되면 자동으로 진급처리 되는 기능 이렇게 3가지 기능이 존재한다.**

**수업관리는 성적입력 출석관리, 과제 관리 등 교수들이 수업들을 관리할 때 사용하는 기능들을 모아 놓은 것이다.**

**메신저 관리는 직접적인 메시지를 보내는 메시지 기능과 자유 게시판과 같은 소규모 커뮤니티 개설 관리하는 소규모 커뮤니티 관리 두가지로 이루어져 있다.**

**3 개발 프로세스 및 소스코드 규칙 **

**이 프로젝트는 전통적인 폭포수(Waterfall) 모델을 기반으로 하되, 설계 단계에서 프로토타입 개발을 병행하여 설계가 적절한지 판단해 설계의 완성도를 높인다.**

**이는 이미 계획단계에서 요구사항이 어느정도 구체적으로 뽑혔기에 분석과정에서 프로토타입을 진행하여 요구사항을 구체화하는 것 보다 설계단계에서**

**포로토타입 모델을 진행하여 요구사항을 분석하여서 나온 기능들이 정확한지**

**검증함과 동시에 프로토타입은 디자인 설계하는 과정에서도 많은 도움이 되기에 설계단계에서 프로토타이핑을 진행하여 전체적인 완성도를 높인다.**

**또한 소스코드를 작성할 때 표준적인 절차는 클래스명은 파스칼 스타일로 작성하며 함수(또는 메서드)와 변수명은 캐멀 스타일로 작성한다.**

**if문과 for문을 사용할 때는 가능한 주석을 쓰며 클래스와 메서드를 선언할 시 무슨 클래스와 메서드인지 설명하는 주석을 반드시 작성해야 한다.**

**4 개발 환경**

**4,1 개발 언어**

**폰트, 그래픽, 노랑, 상징이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**1. 자바 스크립트**

**자바 스크립트는 웹사이트 개발의 표준 언어로 현재 모든 동적 소스코드는 자바 스크립트로 작성되었다.**

**로고, 그래픽, 디자인, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**2.html**

**웹 브라우저에 표시되는 웹 페이지의 구조를 작성하기 위한 표준 마크업 언어이다.**

**로고, 텍스트, 상징, 디자인이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**3.css**

**HTML 등의 마크업 언어로 작성된 문서가 실제로 웹사이트에 표현되는 방법을 정해주는 스타일시트 언어.**

**4.2 개발 환경**

**상징, 일렉트릭 블루, 블루, 직사각형이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**1.vscode**

**개발한 소스 코드 편집기로, 다양한 프로그래밍 언어를 지원하는 경량 통합 개발 도구이다.**

**폰트, 텍스트, 그래픽, 상징이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**2. next.js**

**Next.js는 React를 기반으로 하는 웹 프레임워크로, 서버 사이드 렌더링(SSR)과 정적 사이트 생성(SSG)을 지원하여 빠르고 효율적인 웹 애플리케이션 개발을 가능하게 한다.**

**5. 요구사항 분석**

**5.1 시스템 소프트웨어 요구사항**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | 분류 | 요구사항 명 | 요구사항 내용 | 우선순위 | 비고 |
| SW-01 | **시스템** | **웹사이트 연결** | **동서울대학교 내부의 다른 관련 웹사이트들과 연계·이동이 가능해야 한다.** | **상** |  |
| SW-02 | **시스템** | **로그인 제약** | **메인 포털 사이트 및 모든 사이트들은 로그인한 학생 및 교직원들만 이용할 수 있어야 한다.** | **상** |  |
| SW-03 | **시스템** | **액터마다 다른 뷰어** | **사용자에 따라 메인 포털사이트 및 통합정보 사이트에서 사용할 수 있는 기능이 달라야 한다.** | **중** | **구현중** |
| SW-04 | **시스템** | **직관적인 인터페이스** | **웹사이트의 모든 기능들이 한눈에 보이며 쉽게 찾을 수 있도록 한다.** | **하** |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | 분류 | 요구사항 명 | 요구사항 내용 | 우선순위 | 비고 |
| SW-05 | **시스템** | **낮은 사양 구동** | **어떤 환경에서도 기능 및 웹사이트를 보는데 지장이 없어야 한다.** | **하** |  |
| SW-06 | **시스템** | **통신 접근** | **웹 브라우저에서 url창에 du. porta.com라고 쳤을 때 바로 포털사이트에 들어갈 수 있어야 한다.** | **중** |  |

**5.2. 기능 관련 요구사항**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | 분류 | 요구사항 명 | 요구사항 내용 | 우선순위 | 비고 |
| R-01 | **기능** | **수업 조회** | **교수는 자신이 강의할 수업을 조회할 수 있어야 한다.** | **상** |  |
| R-02 | **기능** | **수강자 명단 조회** | **교수는 자신이 담당하는 강의에 대해 수강 중인 학생들의 정보를 확인할 수 있어야 한다.** | **중** |  |
| R-03 | **기능** | **성적 관리** | **교수가 담당하는 강의에 대해 수강한 학생들의 성적을 입력하고, 기존 성적을 수정하거나 조회할 수 있어야 한다.** | **상** |  |
| R-04 | **기능** | **출석관리** | **교수는 자신이 담당하는 강의에 대해 수강 중인 학생들의 출석 정보를 입력하고, 수정하거나 조회할 수 있어야한다.** | **상** |  |
| R-05 | **기능** | **시간표** | **학생들은 자신이 수강한 수업들을 요일별로 볼 수 있어야 한다.** | **중** |  |
| R-06 | **기능** | **출석 조회** | **학생은 현재 자신이 신청한 강의에 출석 정보를 볼 수 있으며 직원, 조교의 경우 모든 수업들의 학생 들의 출석 정보들을 조회할 수 있다.** | **상** |  |
| ID | **분류** | **요구사항 명** | **요구사항 내용** | **우선순위** | **비고** |
| R-07 | **기능** | **메시지 보내기** | **다른 사용자에게 메시지 보내기, 삭제를 할 수 있어야 한다.**  **.** | **중** |  |
| R-08 | **기능** | **채팅방** | **여러 사람들에게 메시지를 보낼 수 있게 하는 기능이 있어야 한다.** | **중** |  |
| R-09 | **기능** | **로그인** | **사용자는 로그인 사이트에서 학번 및 비밀번호를 입력해서 사용자를 판별하는 기능이 있어야 한다.** | **상** |  |
| R-10 | **기능** | **로그아웃** | **로그아웃 버튼 클릭 시 포털 사이트에서 나가지며 로그인 사이트로 이동하는 기능이 있어야 한다.** | **중** |  |
| R-11 | **기능** | **비밀번호 변경** | **사용자가 비밀번호를 재설정할 수 있어 한다.** | **하** |  |
| R-12 | **기능** | **개인정보 조회** | **자신의 개인정보 (이름, 나이, 주소, 성별, 학년, 학과, 전화 번호,등)을 조회할 수 있어야 한다.** | **하** |  |
| R-13 | **기능** | **개인정보 수정** | **사용자는 자신의 자주 변경되는 정보(주소, 계좌번호)들은 수정할 수 있어야 한다.** | **하** |  |

**5.3 기타 요구사항**

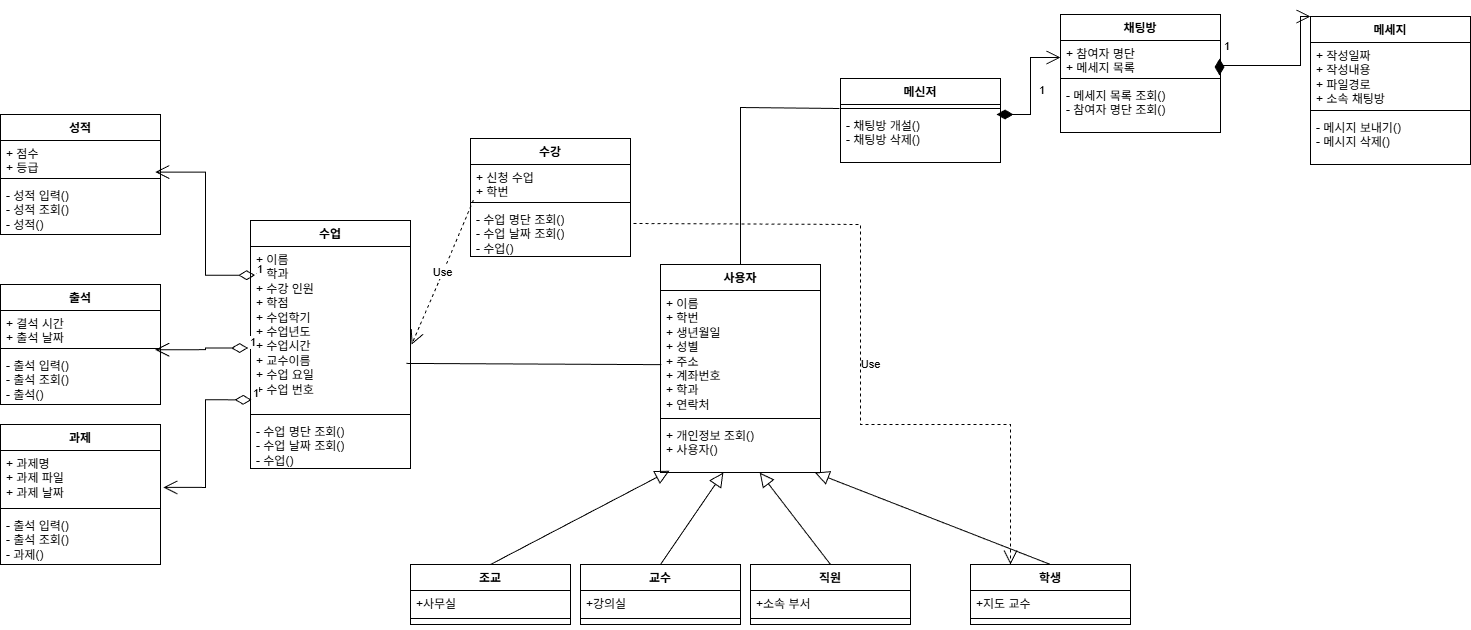
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | 분류 | 요구사항 명 | 요구사항 내용 | 우선순위 | 비고 |
| NF-001 | **성능** | **반응 시간** | **버튼을 클릭하였을 때 웹사이트 가 2초 이내로 반응하여야**  **한다.** | **중** |  |
| NF-002 | **예외조건** | **사용자 로그인 예외사항** | **로그인 입력 시 학번과 비밀번호가 일치하지 않을 시 이에 대한 반응으로 “일치하지 않습니다. 확인해주세요” 라는 메시지가 떠야 한다.** |  |  |

**6. 유스케이스 다이어그램**

**도표, 그림, 스케치, 원이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**7. 클래스 다이어그램**

****

**8. erd**

**스크린샷, 도표, 평면도, 텍스트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**9. DB테이블**

**1. department(학과) 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| department\_id | **INT** | **NOT NULL** |  | **PK** |
| department\_name | **VARCHAR**  **(100)** | **NOT NULL** |  | **UNIQUE** |
| capacity | **INT** | **NOT NULL** |  | **0 이상의 정수만 입력 가능** |
| established\_date | **DATE** | **NOT NULL** |  |  |

**user테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| user\_id | **INT UNSIGNED** | **NOT NULL** | **AUTO\_**  **INCREMENT** | **PK** |
| user\_name | **VARCHAR**  **(50)** | **NOT NULL** |  |  |
| birth\_date | **DATE** | **NOT NULL** |  |  |
| gender | **DATE** | **NOT NULL** |  | **0 또는 1만 가능** |
| phone | **VARCHAR**  **(13)** | **NOT NULL** |  | **000-0000-0000 형식 만 가능** |
| user\_password | **VARCHAR**  **(255 )** | **NOT NULL** |  |  |
| address | **VARCHAR**  **(100)** | **NULL** |  |  |
| account\_number | **CHAR(43)** | **NULL** |  |  |
| department\_id | **INT** | **NOT NULL** |  | **department의 FK** |

**Professor 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| professor\_id | **INT UNSIGNED** | **NOT NULL** |  | **PK user테이블의 fk** |
| lab | **VARCHAR**  **(50)** | **NOT NULL** |  |  |

**Student 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| Student\_id | **INT UNSIGNED** | **NOT NULL** |  | **PK user테이블의 fk** |
| grade\_class | **CHAR(3)** | **NOT NULL** |  |  |
| professor\_id | **INT UNSIGNED** | **NOT NULL** |  | **PK professor테이블의 fk** |
| enrollment\_status | **CHAR(2)** | **NOT NULL** | **재학** | **재학,휴학,재적 3개만 가능** |

**Employee 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| Employee \_id | **INT UNSIGNED** | **NOT NULL** |  | **PK user테이블의 fk** |

**teaching\_assistant 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| assistant\_id | **INT UNSIGNED** | **NOT NULL** |  | **PK user테이블의 fk** |
| office | **VARCHAR**  **(50)** | **NOT NULL** |  |  |

**Lecture 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| lecture\_id | **INT UNSIGNED** | **NOT NULL** | **AUTO\_**  **INCREMENT** | **PK** |
| lecture\_name | **VARCHAR**  **(50)** | **NULL** |  |  |
| enrollment\_count | **INT** | **NOT NULL** |  |  |
| credit | **TINYINT** | **NOT NULL** |  | **0~9 안에 값만 가능** |
| classroom | **VARCHAR**  **(50)** | **NOT NULL** |  |  |
| professor\_id | **INT UNSIGNED** | **NOT NULL** |  | **professor테이블의 fk** |
| day\_of\_week | **CHAR(1)** | **NULL** |  | **'월','화','수','목','금','토','일'**  **중에 하나 만 가능** |
| lecture\_semester | **TINYINT** | **NULL** |  | **1~2 안에 값만 가능** |
| start\_hours | **TIME** | **NOT NULL** |  |  |
| end\_hours | **TIME** | **NOT NULL** |  | **end\_hours는 start\_hours 보다 값이 커야 한다.**  **(start\_hours < end\_hours)** |

**Enrollment 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| enrollment\_id | **INT UNSIGNED** | **NOT NULL** |  | **PK** |
| lecture\_id | **INT UNSIGNED** | **NOT NULL** |  | **Lecture 테이블의 fk** |
| student\_id | **INT UNSIGNED** | **NOT NULL** |  | **student 테이블의 fk** |
| lecture\_year | **CHAR(2)** | **NOT NULL** |  | **0000년 형식만 가능** |
| lecture\_semester | **TINYINT** | **NULL** |  | **1~2 안에 값만 가능** |
| lecture\_hours | **TINYINT** |  |  | **1~2 안에 값만 가능** |

**Attendance 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| attendance\_date | **DATE** | **NOT NULL** |  | **PK** |
| enrollment\_id | **VARCHAR**  **(50)** | **NOT NULL** |  | **Pk**  **Enrollment 테이블의**  **FK** |
| absent\_hours | **INT** | **NOT NULL** |  | **0~98 안에 가능** |

**Grade 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| enrollment\_id | **VARCHAR**  **(50)** | **NOT NULL** |  | **Pk**  **Enrollment 테이블의**  **FK** |
| grade | **VARCHAR**  **(2)** | **NOT NULL** |  | **'A+','A','A-','B+','B','B-','C+','C','C-','D+','D','D-','F' 해당 값들 만 가능** |

**Assignment 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| assignment\_id | **INT UNSIGNED** | **NOT NULL** | **AUTO\_**  **INCREMENT** | **PK** |
| assignment\_name | **VARCHAR**  **(50)** | **NOT NULL** |  |  |
| assignment\_description | **INT** | **NOT NULL** |  |  |
| lecture\_id | **INT UNSIGNED** | **NOT NULL** |  | **Lecture테이블의 FK** |
| registered\_date | **DATE** | **NULL** |  |  |
| file\_path | **VARCHAR**  **(300)** | **NULL** |  |  |

**assignment\_submit 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| assignment\_id | **INT UNSIGNED** | **NOT NULL** | **AUTO\_**  **INCREMENT** | **PK** |
| enrollment\_id | **INT UNSIGNED** | **NOT NULL** |  | **PK** |
| submission\_name | **VARCHAR**  **(100)** | **NOT NULL** |  |  |
| file\_path | **VARCHAR**  **(300)** | **NULL** |  | **Lecture테이블의 FK** |
| submission\_description | **TEXT** | **NOT NULL** |  |  |

**Chatroom 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| chatroom\_id | **DATE** | **NOT NULL** |  | **PK** |
| chatroom\_name | **VARCHAR**  **(30)** | **NOT NULL** |  |  |
| is\_group | **INT** | **NOT NULL** |  | **t 또는 f만 가능** |
| last\_message\_time | **DATE** | **NULL** |  |  |

**Chatroom\_List 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| chatroom\_id | **INT UNSIGNED** | **NOT NULL** |  | **PK**  **chatroom 테이블의 fk** |
| member\_id | **INT UNSIGNED** | **NOT NULL** |  | **PK**  **user 테이블의 fk** |

**Chatting 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| message\_id | **INT UNSIGNED** | **NOT NULL** | **AUTO\_**  **INCREMENT** | **PK** |
| chatroom\_id | **INT UNSIGNED** | **NOT NULL** |  |  |
| created\_date | **DATE** | **NOT NULL** |  |  |
| create\_hours | **TIME** | **NOT NULL** |  |  |
| content | **TEXT** | **NOT NULL** |  |  |
| file\_path | **VARCHAR**  **(300)** | **NULL** |  |  |
| author\_id | **INT UNSIGNED** | **NOT NULL** |  | **User테이블의**  **fk** |

**Leave\_of\_Absence 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| leave\_id | **INT UNSIGNED** | **NOT NULL** | **AUTO\_**  **INCREMENT** | **PK** |
| student\_id | **INT UNSIGNED** | **NOT NULL** |  |  |
| reason | **TEXT** | **NOT NULL** |  |  |
| applied\_at | **DATETIME** | **NOT NULL** |  |  |
| leave\_year | **SMALLINT UNSIGNED** | **NOT NULL** |  | **YYYY 형식만 가능** |
| leave\_semester | **TINYINT UNSIGNED** | **NULL** |  | **1~2 안에 값만 가능** |
| leave\_type | **VARCHAR(10)** | **NULL** |  | **일반휴학/군휴학/**  **질병휴학/기타 중에 하나만 가능** |
| is\_on\_leave | **CHAR(1)** | **NOT NULL** |  | **T 또는 f 선택** |

**back\_school 테이블**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 컬럼 이름 | 타입 | NULL | 기본값 | 제약사항 |
| return\_id | **INT UNSIGNED** | **NOT NULL** | **AUTO\_**  **INCREMENT** | **PK** |
| leave\_id | **INT UNSIGNED** | **NOT NULL** |  | **Leave\_of\_Absence**  **의 fk** |
| manager | **INT UNSIGNED** | **NULL** |  | **Employee의 fk** |
| return\_reason | **DATETIME** | **NOT NULL** |  |  |
| return\_date | **SMALLINT UNSIGNED** | **NOT NULL** |  | **YYYY 형식만 가능** |
| is\_returned | **TINYINT UNSIGNED** | **NULL** |  | **T 또는 f만 가능** |

**10. 기능 해설**

**1. 메인 페이지**

|  |  |
| --- | --- |
| **화면 구성** | |
|  | |
|  | |
| **화면 기능 설명** | **메인 페이지는 로그인 후 가장 먼저 표시되는 화면으로, 포털 사이트의 주요 기능 및 모든 서비스 페이지로 이동할 수 있는 중심 화면입니다.** |

|  |
| --- |
| **순서도** |
|  |
|  |

|  |
| --- |
| **순차 다이어그램** |
|  |
|  |

**2. 메신저**

|  |  |
| --- | --- |
| **화면 구성** | |
|  | |
| **화면 기능 설명** | **메신저 화면은 자신이 소속된 모든 채팅방을 출력하며 또 새로운 채팅방을 만들 수 있게 해주는 창입니다.** | |

|  |
| --- |
| **순서도** |
|  |
|  |

|  |
| --- |
| **순차 다이어그램** |
|  |
|  |

**3. 채팅방**

|  |  |
| --- | --- |
| **화면 구성** | |
|  | |
| **화면 기능 설명** | **채팅방은 매신저에서 특정 채팅방을 클릭할 시 넘어오는 창으로 해당 채팅방의 모든 메시지를 출력하며 채팅방의 메시지를 보낼 수 있게 해주는 창입니다.** |

|  |
| --- |
| **순서도** |
|  |
|  |

|  |
| --- |
| **순차 다이어그램** |
|  |
|  |

**4.개인정보 관리**

|  |  |
| --- | --- |
| **화면 구성** | |
|  | |
| **화면 기능 설명** | **개인정보 관리는 자신의 정보를 확인하고 수정하는 화면입니다.** |

|  |
| --- |
| **순서도** |
|  |
| **순차 다이어그램** |
|  |
|  |

**5. 출석조회**

|  |  |
| --- | --- |
| **화면 구성** | |
|  | |
| **화면 기능 설명** | **출석 조회는 학생 위주의 화면으로 자신이 현재 수강하고 있는 수업들의 출석 정보를 조회할 수 있는 기능으로 화면처럼 달력 형식으로 볼 수 있습니다.** |

|  |
| --- |
| **순서도** |
|  |

|  |
| --- |
| **순차 다이어그램** |
|  |

**6. 출석입력**

|  |  |
| --- | --- |
| **화면 구성** | |
|  | |
| **화면 기능 설명** | **출석 입력은 교수 위주의 화면으로 해당 학생들의 출석 정보를 주차별로 관리할 수 있다.** |

|  |
| --- |
| **순서도** |
|  |

|  |
| --- |
| **순차 다이어그램** |
|  |

**7. 성적 출력**

|  |  |
| --- | --- |
| **화면 구성** | |
|  | |
| **화면 기능 설명** | **성적 출력 페이지는 교수 또는 학생이 DB에 저장된 성적 데이터를 조회할 수 있는 화면으로, 학기별, 과목별로 정리된 성적 정보를 확인할 수 있다. 해당 페이지에서는 각 학생의 과목명, 점수, 등급 등이 테이블 형태로 표시되며, 사용자는 원하는 학기를 선택하여 필터링된 성적을 조회할 수 있다.** |

|  |
| --- |
| **순서도** |
|  |

|  |
| --- |
| **순차 다이어그램** |
|  |

**8. 성적 입력**

|  |  |
| --- | --- |
| **화면 구성** | |
|  | |
| **화면 기능 설명** | **성적 입력 페이지는 담당 교수가 각 학생의 과목별 점수를 입력하면, 시스템이 자동으로 등급을 산출하여 데이터베이스에 저장하는 기능을 제공한다. 교수는 성적 입력 화면에서 수강 학생 목록과 해당 학생의 점수 입력란을 확인할 수 있으며, 점수 입력 후 *저장 버튼*을 누르면 입력한 점수와 변환된 등급이 즉시 DB에 반영된다.** |

|  |
| --- |
| **순서도** |
|  |
|  |

|  |
| --- |
| **순차 다이어그램** |
|  |

**9. 복학 신청**

|  |  |
| --- | --- |
| **화면 구성** | |
|  | |
| **화면 기능 설명** | **복학 신청은 학생 위주의 화면으로 휴학을 했을 경우에 그 휴학 정보를 바탕으로 복학을 신청하는 기능이다.** |

|  |
| --- |
| **순서도** |
|  |

|  |
| --- |
| **순차 다이어그램** |
|  |
|  |

**10. 휴학 신청**

|  |  |
| --- | --- |
| **화면 구성** | |
|  | |
| **화면 기능 설명** | **학생 위주의 기능으로 자신이 신청하고 싶은 휴학을 신청할 수 있게 해주는 화면이다. 휴학의 종류는 군휴학,일반휴학, 질병휴학,기타 총 4개가 있다.** |

|  |
| --- |
| **순서도** |
|  |

|  |
| --- |
| **순차 다이어그램** |
|  |

**11. 시간표**

|  |  |
| --- | --- |
| **화면 구성** | |
|  | |
| **화면 기능 설명** | **출석 입력은 교수 위주의 화면으로 해당 학생들의 출석 정보를 주차별로 관리할 수 있다.** |

|  |
| --- |
| **순서도** |
|  |

|  |
| --- |
| **순차 다이어그램** |
|  |

**2. 기능 요구사항 테스트 계획**

**2.1 인터페이스 요구사항 테스트**

**TC-001 내부 웹사이트 연동 확인**

* **요구사항 내역:  
  학사관리 시스템 포털사이트에서 동서울 대학교 내부에 다른 웹사이트와 연결되어야 한다.**
* **목적: 학사 관리 시스템에서 동서울대학교 내의 타 웹사이트(포털, 도서관, 메일 등)로의 연동이 정상적으로 되는지 확인한다.**
* **테스트 대상:   
  동서울 대학교 내부 웹사이트 연결**
* **테스트 절차:**
  1. **학생 계정으로 로그인한다.**
  2. **웹사이트 상단 또는 메뉴에서 “도서관” 또는 학사홈페이지 클릭**
  3. **버튼을 클릭 시 페이지로 이동 시 되는지 확인한다.**

**4-1. 도서관 클릭했을 경우 URL** https://dlibrary.du.ac.kr/ **(도서관 URL)인지 확인한다**

**4-2 학과 홈페이지 클릭했을 URL이 https://www.du.ac.kr/main.do 인지 확인한다.**

* **기대 결과:**
  1. **모든 연동 메뉴 클릭 시 대상 웹사이트로 정상적으로 이동**
  2. **연동 오류, 인증 오류, 접근 불가 등의 문제없음**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **입력,출력/테스트 케이스** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **입력: 도서관 버튼 클릭** | **T** | **T** | **T** | **F** | **F** | **F** | **T** | **F** |
| **입력: 학과홈페이지 클릭** | **F** | **F** | **F** | **T** | **T** | **T** | **F** | **T** |
| **출력: 도서관 이동** | **T** | **F** | **F** | **F** | **F** | **F** | **F** | **T** |
| **출력: 학과 홈페이지 이동** | **F** | **F** | **F** | **T** | **F** | **F** | **T** | **F** |
| **출력: 이동 및 통신오류** | **F** | **F** | **T** | **F** | **F** | **T** | **F** | **F** |
| **출력: 학과, 도서관이 아닌 사이트로 이동** | **F** | **T** | **F** | **F** | **T** | **F** | **F** | **F** |

**테스트 케이스**

**각 테스트 케이스 설명:  
케이스 1: 도서관 클릭 시 도서관 사이트로 이동**

**케이스 2: 도서관 클릭 시 학과, 도서관이 아닌 사이트로 이동**

**케이스 3: 도서관 클릭 시 이동 통신 및 이동 오류(404 NOT FOUND)**

**케이스 4: 도서관 클릭 시 학과 홈페이지로 이동**

**케이스 5: 학과 홈페이지 클릭 시 학과 홈페이지로 이동**

**케이스 6 학과 홈페이지 클릭 시 로 학과, 도서관이 아닌 사이트로 이동**

**케이스 7: 학과 홈페이지 클릭 시 이동 통신 및 이동 오류(404 NOT FOUND)**

**케이스 8: 학과 홈페이지 클릭 시 도서관으로 이동**

**테스트 케이스 전제 조건:  
사용자는 한 번에 하나의 버튼만 클릭할 수 있으며,  
각 테스트 케이스는 독립적으로 실행되어야 한다.**

**TC-002 비로그인 사용자 접근 제한 확인**

* **요구사항 내역:  
  메인 포털 사이트 및 모든 사이트들은 로그인한 학생 및 교직원들만 이용할 수 있어야 한다.**
* **목적:  
  로그인을 하지 않은 사용자가 포털사이트 및 내부 웹사이트(도서관, 메일 등)에 접근할 수 없는지 확인한다.**
* **테스트 대상:  
  비로그인 상태의 웹 브라우저**
* **테스트 절차:**
  1. **학사관리 시스템 포털사이트에 로그인하지 않은 상태로 웹사이트에 접속한다.**
  2. **포털 사이트의 내부 내용(메뉴, 데이터등)이 보이지 않는지 확인한다.**
  3. **수강신청사이트, 통합처리메뉴 등 다른 내부 시스템에도 로그인 없이 접근을 시도한다.**
  4. **접근 시도 시 제대로 접근을 차단하는지 확인한다.**
* **기대 결과:**
  1. **로그인하지 않은 사용자가 접근을 시도할 경우, 로그인 페이지로 자동 리다이렉트 되거나 접근 불가 안내 메시지가 표시되어야 한다.**
  2. **로그인하지 않고는 어떠한 내부 시스템(포털, 도서관, 메일 등)도 이용할 수 없어야 한다.**

**TC-003 학생용 포털 사이트 및 통합정보 메뉴 이동 처리 확인**

* **요구사항 내역:  
  사용자 신분에 따라 메인 포털사이트 및 통합정보 사이트에서 사용할 수 있는 기능이 달라야 한다.**
* **목적:  
  학생 계정으로 포털사이트 로그인 시, 학생용 포털 사이트로 정상 이동하고, 통합정보 메뉴에서도 학생 전용 기능(예: 휴학신청 등)이 정상 노출되는지 확인한다.**
* **테스트 대상:  
  포털사이트 로그인 폼, 포털사이트 메인, 통합정보 메뉴**
* **테스트 데이터:**
  + **학생 계정 ID: 11101001 PW: 1234 역할: 학생**
* **테스트 절차:**
  1. **학사관리 시스템 포털사이트에 테스트 학생 계정(11101001/1234)으로 로그인한다.**
  2. **로그인 후 학생용 포털사이트로 이동되는지 확인한다.**
  3. **학생용 포털사이트 중앙에 출석조회, 성적조회 등 학생 관련 메뉴가 노출되는지 확인한다.**
  4. **통합정보 메뉴로 이동하여, 학생 전용 메뉴(예: 휴학신청)가 존재하는지 확인한다.**
* **기대 결과:**
  1. **학생 계정으로 로그인 시, 학생용 포털사이트로 정상적으로 이동해야 한다.**
  2. **포털사이트에서 학생 관련 메뉴가 중앙에 배치되어 있어야 한다.**
  3. **통합정보 메뉴로 이동 시, 학생 전용 메뉴(휴학신청)가 노출되어야 한다.**

**TC-004 교수용 포털 사이트 및 통합정보 메뉴 이동 처리 확인**

* **요구사항 내역:**  
  **사용자 신분에 따라 메인 포털사이트 및 통합정보 사이트에서 사용할 수 있는 기능이 달라야 한다.**
* **목적:  
  교수 계정으로 포털사이트 로그인 시 교수용 포털 사이트로 정상 이동하는 지 확인한다.**
* **테스트 대상:  
  포털사이트 로그인 폼**
* **테스트 데이터**
* **교수 계정  
  ID: 10115 PW: 1234 역할: 교수**
* **테스트 절차:**
  1. **학사관리 시스템 로그인 사이트에서 테스트 교수계정(10115 / 1234)으로 로그인을 진행한다.**
  2. **로그인 후 교수용 포털사이트로 정상적으로 이동되는지 확인한다.**
  3. **교수용 포털사이트 중앙에 강좌개설, 성적입력 등 교수 관련 메뉴가 올바르게 배치되어 있는지 확인한다.**
  4. **통합정보 메뉴로 이동하여, 교수** **전용 메뉴(예: 수업관리)가 존재하는지 확인한다.**
* **기대 결과:**
  1. **교수 계정으로 로그인 시, 교수용 포털사이트로 정상적으로 이동해야 한다.**
  2. **포털사이트에서 교수 관련 메뉴(강좌개설, 성적입력 등)가 중앙에 배치되어 있어야 한다.**
  3. **통합정보 메뉴로 이동 시, 교수 전용 메뉴(수업관리 등)만 노출되어야 한다.**

**TC-005 직원용 포털 사이트 및 통합정보 메뉴 이동 처리 확인**

* **요구사항 내역:  
  사용자 신분에 따라 메인 포털사이트 및 통합정보 사이트에서 사용할 수 있는 기능이 달라야 한다.**
* **목적:  
  직원 계정으로 포털사이트 로그인 시, 직원용 포털 사이트로 정상 이동하고,  
  통합정보 메뉴에서도 직원 전용 기능(예: 휴학등록, 수강신청 마감기간처리 등)이 정상 노출되는지 확인한다.**
* **테스트 대상:  
  포털사이트 로그인 폼**
* **테스트 데이터**
* **직원 계정  
  ID: 01115 PW: 1234 역할: 직원**
* **테스트 절차:**
  1. **학사관리 시스템 포털사이트에 테스트 직원 계정(01115/1234)으로 로그인을 진행한다.**
  2. **로그인 후 직원용 포털사이트로 정상적으로 이동되는지 확인한다.**
  3. **직원용 포털사이트에 들어갈 내용인 수강신청 마감기간처리, 휴학처리 등이 제대로 중앙에 배치되어 있는지 확인한다.**
  4. **통합정보 메뉴로 이동하여 직원 메뉴에 휴학등록 메뉴가 존재하는지 확인한다.**
* **기대 결과:**
  + **직원 계정으로 로그인 시, 직원용 포털사이트로 정상적으로 이동해야 한다.**
  + **포털사이트에서 직원 관련 메뉴가 중앙에 배치되어 있어야 한다.**
  + **통합정보 메뉴로 이동 시, 직원 전용 메뉴(강좌개설, 휴학등록 등)가 노출되어야 한다.**

**TC-006 조교용 포털 사이트 및 통합정보 메뉴 이동 처리 확인**

* **요구사항 내역:  
  사용자 신분에 따라 메인 포털사이트 및 통합정보 사이트에서 사용할 수 있는 기능이 달라야 한다**
* **목적:  
  조교계정으로 포털사이트 로그인 시 조교용 포털 사이트로 이동하는지 확인한다.**
* **테스트 대상:   
  포털사이트 로그인 폼**
* **테스트 데이터**
* **조교 계정  
  ID: 11115 PW: 1234 역할: 조교**
* **테스트 절차:**
  1. **학사관리 시스템 포털사이트에 테스트 조교 계정으로 로그인을 진행한다.**
  2. **조교용 포털사이트에 들어갈 내용인 학생 수강신청 번경이 제대로 중앙에 배치되어 있는지 확인한다.**
* **기대 결과:**
  + **조교 계정으로 로그인 시, 조교용 포털사이트로 정상적으로 이동해야 한다.**
  + **포털사이트에서 조교 관련 메뉴(학생 수강신청 변경 등)가 중앙에 배치되어 있어야 한다.**

**TC-006 뷰어 포함**

* **요구사항 내역:  
  jpf,pdf 등 같은 외부 파일을 열 수 있는 뷰어를 통해 파일들을 볼 수 있어야 한다.**
* **목적:  
  게시판 내부에 pdf,word파일을 볼 수 있어야 한다.**
* **테스트 대상:   
  게시판**
* **테스트 데이터  
  test1.pdf,test2.jpg**
* **학생 계정  
  ID: 1101001 PW: 1234**
* **테스트 절차:**
  1. **포털사이트에서 게시판을 탭으로 들어간다.**
  2. **게시판 파일을 클릭한다.**

**3-1. jpg파일 클릭하여 업로드 한다.**

**3-2. pdf파일을 클릭하여 업로드를 진행한다.**

**4. 등록 버튼을 클릭하여 게시판에 업로드를 한다.**

**5. 등록된 게시글에 업로드한 파일들이 출력상태를 확인한다.**

* **기대 결과:**
  + **게시판 내부 뷰어들이 정상적으로 작동된다는 것을 보여준다.**

**테스트 케이스**

| **입력,출력/테스트 케이스** | **1** | **2** | **3** | **4** | **5** | **6** |
| --- | --- | --- | --- | --- | --- | --- |
| **입력: JPG 파일 업로드** | **T** | **F** | **T** | **T** | **F** | **F** |
| **입력: PDF 파일 업로드** | **F** | **T** | **F** | **F** | **T** | **T** |
| **출력: JPG 파일 출력** | **T** | **F** | **F** | **F** | **F** | **F** |
| **출력: PDF 파일 출력** | **F** | **T** | **F** | **F** | **F** | **F** |
| **출력: 파일 출력 오류** | **F** | **F** | **T** | **F** | **T** | **F** |
| **출력: 파일 업로드 오류** | **F** | **F** | **F** | **T** | **F** | **T** |

**테스트 케이스 설명**

**케이스 1: JPG 업로드 시 JPG출력**

**케이스 2: PDF 업로드 시 PDF출력**

**케이스 3: JPG 업로드 시 파일 출력 오류**

**케이스 4: JPG 업로드 시 파일 업로드 오류**

**케이스 5: PDF 업로드 시 파일 출력 오류**

**케이스 6 PDF 업로드 시 파일 업로드 오류**

**2.2 기능 요구사항 테스트**

**TC-006 강의 관리**

* **요구사항 내역:  
  교수는 자신이 등록한 강의를 등록, 수정, 조회할 수 있어야 한다**
* **목적:  
  교수가 강의 업로드가 정상적으로 가능한지 확인한다.**
* **테스트 대상: 교수 액터 시간표 강좌개설 유스케이스**
* **테스트 데이터**

**테스트 데이터1:  
강의명: c 교수명: 홍길동 학점:3 수업시간 5,6,7교시 수업요일: 수요일**

**테스트 데이터2:  
강의명:c 교수명: 홍길동 학점:4 수업시간 5,6,7,8교시 수업요일: 수요일**

* **교수 계정  
  ID: 10115 PW: 1234 역할: 교수**
* **학생 계정:**

**ID: 1101001 PW: 1234**

* **테스트 절차:**
  1. **교수 계정으로 통합 정보에서 수업 메뉴 클릭**
  2. **강의 개설 탭 클릭**

**3. 강의 등록 클릭**

**4. 테스트 데이터1대로 강의정보를 입력 및 등록**

**5. 수업 탭에 자신 강좌 조회 클릭**

**6. 등록된 강의가 강의 목록에서 정확히 조회되는 지 확인한다.**

**7. 강의 정보(강의명, 교수명, 학점, 수업시간, 요일)가 입력값과 일치하는지**

**확인한다**

**8. 테스트 데이터2 대로 강의 정보 입력**

**9. “학점과 수업시간이 일치하지 않습니다” 출력**

**9. 학생계정으로 url을 이용하여 강의 개설 탭 접근**

**10. “해당 기능을 사용” 메시지 출력”**

* **기대 결과:**

1. **강의가 정상적으로 등록되고, “내 강좌” 목록에서 정확히 조회된다.**
2. **등록된 정보(강의명, 교수명, 학점, 시간, 요일)가 모두 입력한 값과 동일하다.**

**테스트 케이스**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **입력,출력/테스트케이스** | **1** | **2** | **3** | **4** |
| **입력: 교수계정** | **T** | **T** | **F** | **T** |
| **입력: 학생계정** | **F** | **F** | **T** | **F** |
| **입력: 테스트 데이터1** | **T** | **F** | **-** | **F** |
| **입력: 테스트 데이터2** | **F** | **T** | **-** | **F** |
| **출력: 강의 개설 탭 접근 여부** | **T** | **T** | **F** | **T** |
| **출력: 강의 등록 성공** | **T** | **F** | **-** | **F** |
| **출력: 강의 등록 실패** | **F** | **T** | **-** | **T** |

**테스트 케이스 설명:**

**테스트 케이스 1: 교수 계정으로 테스트 데이터1로 강의 등록 시 등록 성공**

**테스트 케이스 2: 교수 계정으로 테스트 데이터2로 강의 등록 시 등록 실패**

**테스트 케이스 3 학생 계정으로 강의 개설 탭 접근 시 실패**

**테스트 케이스 4 교수 계정으로 데이터 없이 강의 동록 시 등록 실패**

**TC-006 수강자 명단 조회**

* **요구사항 내역:  
  교수는 자신이 등록한 강의를 등록, 수정, 조회할 수 있어야 한다**
* **목적:  
  교수가 수강자 명단을 정상적으로 조회 가능하게 한다.**
* **테스트 대상:   
  교수 액터 시간표 수강자 명단 조회 유스케이스**
* **테스트 데이터**

**교수 계정  
ID: 10115 PW: 1234 역할: 교수**

**학생1 정보:  
이름: 홍길동 학번:2010116**

**학생2 정보:  
이름: 유관순 학번:2311117**

**학생4 정보:  
이름: 이순신 학번:2110122**

* **테스트 절차:**
  1. **통합 정보에서 수업 메뉴 클릭**
  2. **수강자 명단 탭 클릭**
  3. **수업 선택**

**3. 수업의 학생 명단이 테스트 데이터와 일치하는지 확인**

* **기대 결과:**
  1. **교수자가 수강자 명단을 조회할 때, 실제로 수강 신청한 학생들의 정보(이름, 학번 등)가 시스템에 정확히 반영되고 일치함을 확인할 수 있다.**
  2. **수강자 명단에서 보여지는 학생1, 학생2, 학생3의 학번 및 이름이**

**테스트데이터의 것과 동일할 것을 기대됨**

**TC-006 성적 관리**

* **요구사항 내역:  
  교수가 담당하는 강의에 대해 수강한 학생들의 성적을 입력하고, 기존 성적을 수정하거나 조회할 수 있어야 한다**
* **목적:  
  교수가 성적을 정상적으로 입력을 가능하게 한다.**
* **테스트 대상:   
  교수 액터 시간표 성적관리 유스케이스**
* **테스트 데이터**

**교수 계정: ID:   
10115 PW: 1234 역할: 교수**

**학생1 정보: 이름:   
홍길동 학번:2010116 입력 성적 ‘F’**

**학생2 정보: 이름:   
유관순 학번:2311117 입력 성적 ‘G’**

**수업 데이터:   
강의명: c 교수명: 홍길동 학점:3 수업시간 5,6,7교시 수업요일: 수요일**

**테스트 절차:**

* 1. **통합 정보에서 수업 메뉴 클릭**
  2. **성적관리 탭 클릭**
  3. **수업 c 선택**

**4. 수업의 테스트 데이터의 학생1의 성적 입력**

**5. “성적입력 완료되었습니다.” 출력**

**5. 수업의 테스트 데이터의 학생2의 성적 입력**

**6. “잘못된 성적입니다.” 메시지 출력 후 입력 취소**

**7. 성적 조회 탭 클릭**

**8. 저장된 학생의 성적이 입력 값과 일치하는지 확인한다.**

* **기대 결과:**
  1. **정상 입력 시(학생1): 입력한 성적이 정확히 저장/조회됨**
  2. **비정상 입력 시(학생2): 오류 메시지가 출력되고, 입력이 저장되지 않음**
  3. **입력 실패한 학생2 외에, 이미 저장된 학생1의 성적은 정상적으로 유지됨**

|  |  |  |
| --- | --- | --- |
| **테스트케이스** | **1** | **2** |
| **입력값** | **F** | **G** |
| **예상 결과** | **승인** | **반려** |
| **실제 결과** | **미정** | **미정** |

**TC-006 출석 관리**

* **요구사항 내역:  
  교수는 자신이 담당하는 강의에 대해 수강 중인 학생들의 출석 정보를 입력하고, 수정하거나 조회할 수 있어야한다.**
* **목적:  
  교수가 출석을 정상적으로 입력을 가능하게 한다.**
* **테스트 대상:  
   교수 액터 시간표 출석관리 유스케이스**
* **테스트 데이터**

**교수 계정  
ID: 10115 PW: 1234 역할: 교수**

**학생1 정보:  
이름: 홍길동 학번:2010116 결석시간 입력 3**

**학생2 정보:  
이름: 유관순 학번:2311117 결석시간 입력 4**

**수업 데이터:   
강의명: c 교수명: 홍길동 학점:3 수업시간 5,6,7교시 수업요일: 수요일**

* **테스트 절차:**
  1. **통합 정보에서 수업 메뉴 클릭**
  2. **출석관리 탭 클릭**
  3. **수업 c 선택**

**4. 학생1의 결석시간 ‘3’ 입력 후 저장 시도**

**5. “출석입력이 완료되었습니다.” 출력**

**5. 학생2의 결석시간 ‘4’ 입력 후 저장 시도**

**6. “잘못된 성적입니다.” 메시지 출력 후 입력 취소**

**7. 출석 조회 탭 클릭**

**8. 학생1의 결석시간이 ‘3’으로 정확히 저장되어 있는지 확인**

**9. 학생2의 결석시간이 저장되지 않았는지 확인**

**기대 결과:**

* 1. **정상 입력 시(학생1): 입력한 출석이 정확히 저장된다.**
  2. **비정상 입력 시(학생2): 오류 메시지가 출력되고, 입력이 저장되지 않음**
  3. **입력 실패한 후에도 학생2 외에, 이미 저장된 학생1의 출석은 정상적으로 유지됨**

|  |  |  |
| --- | --- | --- |
| **출력/테스트 케이스** | **1** | **2** |
| **입력값** | **3** | **4** |
| **예상 결과** | **승인** | **반려** |
| **실제 결과** | **미정** | **미정** |

**TC-006 과제 입력**

* **요구사항 내역:  
  교수는 자신이 담당하는 강의에 대해 과제를 등록하고, 등록된 과제의 내용을 수정하거나 마감 기한 내 제출된 학생 과제를 평가할 수있어야 한다**
* **목적:  
  교수가 과제를 정상적으로 생성할 수 있는 지 확인한다.**
* **테스트 대상:  
   교수 액터 시간표 과제관리 유스케이스**
* **테스트 데이터**

**교수 계정  
ID: 10115 PW: 1234 역할: 교수**

**과제1:  
이름: C에서의 입출력, 내용: 1234, 파일 결과.PNG**

**수업 데이터:   
강의명: c 교수명: 홍길동 학점:3 수업시간 5,6,7교시 수업요일: 수요일**

* **테스트 절차:**
  1. **통합 정보에서 수업 메뉴 클릭**
  2. **과제 생성 탭 클릭**
  3. **수업 c 선택**

**4. 과제 등록 클릭**

**5. 과제 데이터 기반으로 입력 진행**

**6. 등록 버튼 클릭**

**7. 과제가 데이터와 동일하게 생성되었는 지 확인한다.**

**8. 파일이 정상적으로 출력되는 지 확인한다.**

* **기대 결과:**
  1. **과제 데이터와 동일한 과제가 생성된다.**

**2. 첨부 파일로 작성된 사진이 깨짐 없이 정상적으로 출력된다**

**TC-006 과제 제출 확인**

* **요구사항 내역:  
  교수는 자신이 담당하는 강의에 대해 과제를 등록하고, 등록된 과제의 내용을 수정하거나 마감 기한 내 제출된 학생 과제를 평가할 수있어야 한다**
* **목적:  
  교수가 제출된 과제를 정상적으로 확인가능 한 지 테스트한다.**
* **테스트 대상:  
   교수 액터 시간표 과제관리 유스케이스**
* **테스트 데이터**

**교수 계정  
ID: 10115 PW: 1234 역할: 교수**

**과제 제출 데이터:  
이름: 2010116 홍길동 과제 제출합니다. , 내용124 파일 과제결과.PNG**

**수업 데이터:   
강의명: c 교수명: 홍길동 학점:3 수업시간 5,6,7교시 수업요일: 수요일**

* **테스트 절차:**
  1. **통합 정보에서 수업 메뉴 클릭**
  2. **과제 결과 탭 클릭**
  3. **수업 c 선택**

**4. 2010116 홍길동 과제 제출합니다 클릭**

**5. 저장된 데이터와 출력된 데이터가 동일한지 확인한다.**

* **기대 결과:**
  1. **과제제출데이터와 동일한 내용이 출력된다.**
  2. **첨부 파일로 작성된 사진이 깨짐 없이 정상적으로 출력된다.**

**TC-006 수강 변경**

* **요구사항 내역:  
  조교는 특정 학생의 수강 정보를 조회할 수 있다.**

**또한 조교는 학생의 기존 신청 과목을 다른 과목으로 변경할 수 있어야 한다.**

* **목적:  
  조교가 학생의 수강정보를 조회 및 변경 시 데이터가 무결한지 확인한다.**
* **테스트 대상:  
   조교 액터 수강변경 유스 케이스**
* **테스트 데이터**
* **조교 계정  
  ID: 11115 PW: 1234 역할: 조교**
* **학생1 정보:  
  이름: 홍길동 학번:2010116**

**수강 데이터:  
삭제 데이터: 수업명: c 학점:3 수업시간: 3교시~6교시 수업요일: 월요일**

**입력 데이터: 수업명: python 학점:3 수업시간:2교시~5교시 수업요일: 금요일**

* **테스트 절차:**
  1. **통합 정보에서 수강변경 메뉴 클릭**
  2. **테스트 데이터 학생 학번 입력**
  3. **출력된 목록 수강 데이터에서 수업명 c인 수업 삭제 진행**

**4. 새로고침 클릭 후 수강 목록에서 결과 확인**

**5. 수업검색에서 python 조회**

**6. python 강의 신청**

**7.새로고침 후 학생의 수강 목록에 python이 추가되었는지 확인한다**

**8. dbms에서 테스트 학생에 수강 데이터 조회하여 일치하는지 확인**

* **기대 결과:**
  1. **삭제 시 수강목록에서 데이터가 보이지 않고 삭제된다.**

**2. 신청한 수업이 제대로 수강목록에 추가되는지 확인한다.**

**3. 삭제 및 입력이 실제 데이터에 반영된다.**

**TC-006 수강 신청**

* **요구사항 내역:  
  학생들은 수강신청사이트에서 신청가능 한 수업들을 보고 신청 및 신청취소를 할 수 있어야 한다.**

**이 때 수업 요일 및 시간이 겹칠 시 수강이 안되야 한다.**

* **목적:  
  학생이 수강 신청사이트에서 신청 및 신청취소를 진행할 시 결합요소들이 존재하는지 체크한다.**
* **테스트 대상:  
   학생액터 수강신청 유스케이스**
* **테스트 데이터:**

**학생 계정 ID: 11101001 PW: 1234 역할: 학생**

**수강 데이터:  
취소데이터: 수업명: c 학점:3 수업시간: 3교시~6교시 수업요일: 월요일**

**신청데이터: 수업명: java 학점:3 수업시간:2교시~5교시 수업요일: 금요일**

**신청데이터: 수업명: 예술과 미학 수업시간:2교시~4교시 수업요일: 금요일**

* **테스트 절차:**
  1. **통합 정보에서 수강신청 메뉴 클릭**
  2. **수업목록 java 신청 클릭**

**3. 새로고침 클릭 후 신청 목록에서 java수업 존재 확인**

**4. 신청 목록에서 c수업 신청 취소 클릭**

**5. 새로고침 후 신청 목록에서 c가 삭제되었는지 확인**

**6. 수업목록 예술과 미학 수업 클릭 신청**

**7. “수업시간이 겹쳐 신청할 수 없습니다.” 출력확인**

* **기대 결과:**
  1. **신청 시 신청목록에서 이를 반영하는지 확인한다.**

**2. 신청취소 시 신청목록에서 이를 반영하는지 확인한다.**

**3. 수업요일 및 수업시간이 겹치는 수업이 신청이 거절되는지 확인한다.**

**TC-006 시간표**

* **요구사항 내역:  
  학생들은 자신이 수강한 수업들의 시간을 알 수 있어야 한다.**

**이때 수업들의 시간을 시간표 형태로 볼 수 있어야 한다.**

* **목적:  
  학생이 자신이 수강한 수업들의 시간표가 정상적으로 출력되는지 확인한다.**
* **테스트 대상:  
   학생 액터 시간표 유스케이스**
* **테스트 데이터:**

**학생1 계정 ID: 11101001 PW: 1234 역할: 학생**

**학생1 수강 데이터:**

**수업명: java 학점:3 수업시간:2교시~5교시 수업요일: 금요일**

**수업명: c++ 수업시간5교시~7교시 수업요일: 금요일**

**학생2 계정 ID:112010002 PW:1234 역할: 학생**

**학생2 수강 데이터:**

**없음**

**테스트 절차:**

* 1. **학생1으로 통합 정보에서 시간표 메뉴 클릭**
  2. **시간표 조회 클릭**

**3. 현재 수강데이터와 맞게 시간표과 출력되었는지 확인 진행한다.**

**4. 학생2으로 통합 정보에서 시간표 메뉴 클릭**

**5. 시간표 조회 클릭**

**6. “현재 수강한 수업이 없습니다. 출력되는 지 확인”**

* **기대 결과:**
  1. **수강 데이터가 존재할 경우 정상적으로 수강 데이터에 맞춰 시간표가 출력된다.**

**2. 수강한 수업이 없을 경우 시간표 출력이 거절된다.**

**TC-006 휴학 신청**

* **요구사항 내역:  
  학생들은 수강 포털에서 휴학신청을 진행할 수 있어야 한다.**

**이때 휴학 기록을 조회하여 일반 휴학기록이 있을 시 자동으로 거절되어야 한다.**

* **목적:  
  학생이 많이 신청하는 일반 휴학유형으로 휴학신청 진행할 시 정상적으로 신청된 데이터가 조회되는지 확인한다.**
* **테스트 대상:  
   학생 액터 휴학신청 유스케이스**
* **테스트 데이터:**

**학생1 계정 ID: 11101001 PW: 1234 역할: 학생**

**학생1 휴학기록 데이터:**

**데이터 없음**

**학생1 휴학신청 데이터:**

**휴학유형: 일반 휴학 휴학사유: 진로 관련 휴학일자:2002-11-11**

**학생2 계정 ID:112010002 PW:1234 역할: 학생**

**학생2 휴학 데이터:**

**학번:112011002 휴학유형: 일반 휴학 휴학사유:기타 휴학일자:2001-11-11**

**테스트 절차:**

* 1. **학생1으로 통합 정보에서 휴학신청 메뉴 클릭**
  2. **휴학 신청서 작성 클릭**

**3. 휴학 신청 데이터에 맞게 신청서 입력 후 등록 버튼 클릭**

**4. 휴학 신청 기록 클릭하여 신청서가 있는지 확인**

**5. 학생2로 휴학신청 메뉴 클릭**

**6. 신청 유형 일반 휴학으로 신청서 입력 후 등록 버튼 클릭**

**7. “일반 휴학을 이미 했었기에 일반 휴학을 진행할 수 없습니다.” 출력되는지 확인**

* **기대 결과:**
  1. **일반 휴학 기록이 없을 경우 정상적으로 일반 휴학이 진행된다.**

**2. 일반 휴학 기록이 있을 경우 일반 휴학이 거절된다.**

**TC-006 휴학처리**

* **요구사항 내역:  
  직원들은 학생들이 신청한 휴학 신청을 반려하거나 승인할 수 있어야 한다.**
* **목적:  
  휴학처리가 정상적으로 되는 지 확인한다.**
* **테스트 대상:  
  직원 액터 유스케이스 휴학처리**

**테스트 데이터:  
학생1 휴학신청 데이터:**

**휴학유형: 일반 휴학 휴학사유: 진로 관련 휴학일자:2002-11-11**

**학생2 휴학신청 데이터:**

**휴학유형: 일반 휴학 휴학사유: 사생활 휴학일자:2002-12-31**

**직원 계정**

**ID: 01115 PW: 1234 역할: 직원**

**학생1 계정:**

**ID: 1101001 PW: 1234**

**학생2 계정:**

**ID: 1102002 PW: 456777**

* **테스트 절차:**
  1. **직원 계정으로 통합 정보에서 휴학처리 클릭**

**2. 테스트 데이터에 학생1 휴학신청을 승인 버튼 클릭**

**3. 학생1의 휴학 신청의 승인 여부가 승인으로 바뀌었는지 확인**

**4. 테스트 데이터에 학생2 휴학신청을 반려 버튼 클릭**

**5. 학생1의 휴학 신청의 승인 여부가 반려로 바뀌었는지 확인**

* **기대 효과**

**1. 휴학처리에서 승인이 정상적으로 작동된다.**

**2. 휴학처리에서 반려가 정상적으로 작동된다**

**TC-006 자신정보 조회**

* **요구사항 내역:  
  자신의 개인정보(이름, 나이, 주소, 성별, 학년, 학과, 전화번호, 등)을 조회할 수 있으며 자주 변경되는 정보(주소, 계좌번호)들은 수정할 수 있도록 해야 한다.**
* **목적:  
  사용자가 자신 정보를 조회가 원하는 하게 이루어지는 확인하고 수정사항이 제대로 반영되는지 확인한다.**
* **테스트 대상:  
   사용자 액터 정보 조회 수정 유스케이스**
* **테스트 데이터:**

**학생1 계정  
ID: 11101001 PW: 1234 역할: 학생**

**이름: 홍길동 학번: 11101001 학과: 컴퓨터 소프트웨어학과 학년:3**

**수정 시 입력 값: 주소: 가나시 다라구 마바동**

**계좌번호: 0000-0000-0000-0000로 입력**

**교수1 계정  
ID: 10115 PW: 1234 역할: 교수**

**이름: 이상 학번: 10115 학과: 컴퓨터 소프트웨어학과 신분: 교수**

**수정 시 입력 값: 주소: 가시 다구 바동**

**계좌번호: 1111-1111-0000-0000로 입력**

**조교1 계정  
ID: 11115 PW: 1234 역할: 조교**

**이름: 유관순 학번: 10115 학과: 컴퓨터 소프트웨어학과 신분: 조교**

**수정 시 입력 값: 주소: 사시 아구 차동**

**계좌번호: 1111-1111-0000-0000로 입력**

* **직원1 계정  
  ID: 01115 PW: 1234 역할: 직원**

**이름: 이순신 학번: 01115 학과: 컴퓨터 소프트웨어학과 신분: 직원**

**수정 시 입력 값: 주소: 나시 구구 마마동**

**계좌번호: 1111-1111-1111-0000로 입력**

**테스트 절차:**

* 1. **학생1으로 통합 정보에서 자신정보 메뉴 클릭**
  2. **학생1의 테스트 데이터 값과 조회창에서 출력된 값이 일치하는지 확인진행**

**3. 수정 버튼을 클릭해 주소 및 계좌번호를 입력 값으로 입력 후 등록**

**4. 조회하여 입력값이 제대로 출력되었는지 확인**

**5. 교수1로 통합 정보에서 자신정보 메뉴 클릭**

**6. 교수1의 테스트 데이터에 값과 조회창에서 출력된 값이 일치하는 확**

**인진행**

**7. 수정 버튼을 클릭해 주소 및 계좌번호를 입력 값으로 입력 후 등록**

**8. 조회하여 입력값이 제대로 출력되었는지 확인한다.**

**9. 조교 계정으로 통합정보 메뉴에서 자신정보 조회를 클릭**

**10.조교1의 테스트 데이터 값과 조회창에서 출력된 값이 일치하는지 확인진행**

**11.수정 버튼을 클릭해 주소 및 계좌번호를 입력 값으로 입력 후 등록**

**12.조회하여 입력값이 제대로 출력되었는지 확인**

**13.직원1 계정으로 통합정보 메뉴에서 자신정보 조회를 클릭**

**10.직원1의 테스트 데이터 값과 조회창에서 출력된 값이 일치하는지 확인진행**

**11.수정 버튼을 클릭해 주소 및 계좌번호를 입력 값으로 입력 후 등록**

**12.조회하여 입력값이 제대로 출력되었는지 확인**

* **기대 결과:**
  1. **학생의 정보를 조회할 시 테스트 데이터와 일치하는 값이 조회된다.**

**2. 학생 데이터를 수정할 시 입력 값이 정상적으로 수정된다.**

**3. 교수의 정보를 조회할 시 테스트 데이터와 일치하는 값이 조회된다.**

**4. 교수 데이터를 수정할 시 입력 값이 정상적으로 출력된다.**

**5. 조교의 정보를 조회할 시 테스트 데이터와 일치하는 값이 조회된다.**

**6. 조교 데이터를 수정할 시 입력 값이 정상적으로 수정된다.**

**7. 직원의 정보를 조회할 시 테스트 데이터와 일치하는 값이 조회된다.**

**8. 직원 데이터를 수정할 시 입력 값이 정상적으로 출력된다.**

**.**

**TC-006 게시글 작성, 조회**

**요구사항 내역:  
메인 포털 사이트와 연결된 글을 작성할 수 있는 게시판사이트가 만들어져야 한다.**

**사용자들은 게시판에서 글 작성,조회를 할 수 있다.**

* **목적:  
  사용자가 개시글 작성에 결합요소가 존재 확인하고 이를 수정한다.**
* **테스트 대상:  
   사용자 액터 게시글 조회 유스케이스**
* **테스트 데이터:**

**학생1 계정 ID: 11101001 PW: 1234 역할: 학생**

**게시글 작성 값:  
 제목: 테스트 게시글입니다. 내용: 내용이 출력되는지 확인합니다.**

**사진: 고양이1.jpg**

**테스트 절차:**

* 1. **학생1으로 포털사이트에서 게시판 메뉴 클릭**
  2. **게시판에서 “테스트 게시판” 클릭**

**3. 게시판에서 게시글 작성 클릭**

**4. 게시글 작성에서 게시글 작성 값을 입력한 후 게시글 등록 클릭**

**5. 게시글 목록에서 입력한 게시글이 있는지 확인한다.**

**6. 입력한 게시글을 클릭하여 게시글에서 입력한 값과 조회된 값이 일치하는 지 확인한다.**

* **기대 결과:**
  1. **게시글 작성이 정상적으로 이루어진다.**

**2. 작성된 게시글이 정상적으로 조회된다.**

**TC-006 비밀번호 변경**

* **요구사항 내역:  
  사용자가 비밀번호를 잃어버렸을 시 비밀번호 찾기를 통해 비밀번호를 재설정할 수 있어야 한다.  
  목적:  
  사용자가 비밀번호를 분실했을 때, 비밀번호 찾기 기능을 통해 안전하고 정상적으로 비밀번호를 재설정할 수 있는지를 확인한다.**
* **테스트 대상:  
   사용자 액터 비밀번호 변경 유스케이스**
* **테스트 데이터:**

**학생1 계정**

**ID: 11101001 전화번호: 010-1111-1111 이름: 홍길동**

**교수1 계정**

**ID: 10115 전화번호: 010-2222-2222 이름: 이상**

**조교1 계정**

**ID: 11115 전화번호: 010-3333-3333 이름: 유관순**

**직원1 계정**

**ID: 01115 전화번호: 010-4444-4444 이름: 이순신**

* **테스트 절차:  
  1. 학사관리시스템을 실행하고 로그인 사이트에 접속한다.  
  2. 로그인 화면에서 "비밀번호 찾기" 또는 "비밀번호 재설정" 버튼을 클릭한다.  
  3. 사용자 식별 정보(예: 학번, 이름, 이메일 등)를 입력한다.  
  4. "인증 요청" 버튼을 클릭하여 본인 확인 절차를 수행한다.  
  5. 수신한 인증 코드를 입력하고 "확인"을 클릭한다.  
  6. 새로운 비밀번호와 비밀번호 확인란에 동일한 값을 입력한다.(예:4321)  
  7. "비밀번호 변경" 또는 "재설정" 버튼을 클릭  
  8. 변경한 새 비밀번호(예:4321)로 다시 로그인 한다.**
* **기대 결과:  
  1. 로그인 페이지가 정상적 으로 표시된다.  
  2. 비밀번호 재설정 화면으로 이동된다.  
  3. 입력란에 값이 정상적으로 입력되고, 다음 단계로 이동할 수 있는 버튼이 활성화된다.  
  4. 인증 코드가 전송되고, 인증 입력란이 표시된다.  
  5. 인증이 성공하며 비밀번호 재설정 화면이 표시된다.  
  6. 두 입력란의 값이 일치하면 "변경" 버튼이 활성화된다.  
  7. 비밀번호가 정상적으로 변경되었다는 메시지가 표시되고, 로그인 화면으로 이동된다.  
  8. 로그인이 성공하며 포털 메인 화면으로 이동된다.**

**TC-006 로그아웃**

* **요구사항 내역:  
  사용자는 로그인 사이트에서 학번 및 비밀번호를 입력하여 포털 사이트로 이동할 수 있으며 포털사이트에서 "로그아웃" 버튼 클릭 시 포털  
  사이트에서 나가지며 로그인 사이트로 이동한다.**
* **목적:  
  사용자가 로그아웃이 정상적으로 실행되는지 확인한다**
* **테스트 대상:  
   사용자액터 로그아웃 유스케이스**
* **테스트 데이터:**

**학생1 계정**

**ID: 11101001 PW: 1234 역할: 학생**

**교수1 계정**

**ID: 10115 PW: 1234 역할: 교수**

**조교1 계정**

**ID: 11115 PW: 1234 역할: 조교**

**•직원1 계정**

**ID: 01115 PW: 1234 역할: 직원**

* **테스트 절차:  
  학생1 계정  
  1. 학사관리시스템을 실행하고 로그인 사이트에 접속한다.  
  2. 학번 입력란에 11101001을 입력한다.  
  3. 비밀번호 입력란에 1234를 입력한다.  
  4. "로그인" 버튼을 클릭한다.  
  5. 포털 화면 상단 또는 메뉴에서 "로그아웃" 버튼을 클릭한다.  
  6. 다시 브라우저에서 "뒤로 가기" 버튼을 클릭한다.  
  7. 로그인 페이지에서 로그인 버튼 외 다른 포털 기능을 클릭하려 시도한다.**

**8.직원,교수,조교계정을 대상으로 이를 반복하여 수행한다.**

* **기대 결과:  
  모든 사용자에 대해서  
  1. 로그인 화면이 정상적으로 출력된다.  
  2. 학번 입력란에 입력값이 정상적으로 표시된다.  
  3. 비밀번호 입력란에 입력값이 ●●●● 형태로 표시된다.  
  4. 포털 사이트의 메인 화면으로 정상적으로 이동된다.  
  5. 사용자 세션이 종료되고 로그인 화면으로 이동된다.  
  6. 포털 화면이 보이지 않고, 로그인 페이지가 유지된다. 다음을 만족한다**

**TC-006 로그인**

* **요구사항 내역:  
  사용자는 로그인 사이트에서 학번 및 비밀번호를 입력하여 포털 사이트로 이동할 수 있으며 포털사이트에서 "로그아웃" 버튼 클릭 시 포털  
  사이트에서 나가지며 로그인 사이트로 이동한다.**
* **목적:  
  사용자가 로그인이 정상적으로 실행되는지 확인한다**
* **테스트 대상:  
   사용자액터 로그인 유스케이스**
* **테스트 데이터:**

**학생1 계정**

**ID: 11101001 PW: 1234 역할: 학생**

**교수1 계정**

**ID: 10115 PW: 1234 역할: 교수**

**조교1 계정**

**ID: 11115 PW: 1234 역할: 조교**

**•직원1 계정**

**ID: 01115 PW: 1234 역할: 직원**

* **테스트 절차:  
  학생1 계정  
  1. 학사관리시스템을 실행하고 로그인 사이트에 접속한다.  
  2. 학번 입력란에 11101001을 입력한다.  
  3. 비밀번호 입력란에 1234를 입력한다.  
  4. "로그인" 버튼을 클릭한다.  
  5. 포털 사이트에서 사용자 이름 또는 학번이 화면에 표시되는지 확인한다.  
  교수1 계정  
  1. 학사관리시스템을 실행하고 로그인 사이트에 접속한다.  
  2. 학번 입력란에 10115을 입력한다.  
  3. 비밀번호 입력란에 1234를 입력한다.  
  4. "로그인" 버튼을 클릭한다.  
  5. 포털 사이트에서 사용자 이름 또는 학번이 화면에 표시되는지 확인한다.  
  조교1 계정  
  1. 학사관리시스템을 실행하고 로그인 사이트에 접속한다.  
  2. 학번 입력란에 11115을 입력한다.  
  3. 비밀번호 입력란에 1234를 입력한다.  
  4. "로그인" 버튼을 클릭한다.  
  5. 포털 사이트에서 사용자 이름 또는 학번이 화면에 표시되는지 확인한다.  
  직원1 계정  
  1. 학사관리시스템을 실행하고 로그인 사이트에 접속한다.  
  2. 학번 입력란에 01115을 입력한다.  
  3. 비밀번호 입력란에 1234를 입력한다.  
  4. "로그인" 버튼을 클릭한다.  
  5. 포털 사이트에서 사용자 이름 또는 학번이 화면에 표시되는지 확인한다.**
* **기대 결과:**

**모든 사용자에 대해서  
1. 로그인 화면이 정상적으로 출력된다.  
2. 학번 입력란에 입력값이 정상적으로 표시된다.  
3. 비밀번호 입력란에 입력값이 ●●●● 형태로 표시된다.  
4. 포털 사이트의 메인 화면으로 정상적으로 이동된다.  
5. 로그인된 사용자 정보가 화면에 표시된다.**

**다음을 만족한다.**

**TC-006 출석조회**

**요구사항 내역:  
학생은 현재 자신이 신청한 강의에 출석 정보를 볼 수 있으며 직원, 조교의 경우 모든 수업들의 학생들의 출석 정보들을 조회할 수 있다.**

* **목적:  
  사용자가 개시글 작성에 결합요소가 존재 확인하고 이를 수정한다.**
* **테스트 대상:  
   사용자 액터 출석조회 유스케이스**
* **테스트 데이터:**

**학생1 계정  
ID: 11101001 PW: 1234 역할: 학생 이름: 홍길동 학번:11101001**

**조교1 계정  
ID: 11115 PW: 1234 역할: 조교**

**직원1 계정  
ID: 01115 PW: 1234 역할: 직원**

**출석 데이터**

**수업: java 이름: 홍길동 학번:2010116 출석날짜: 2025-06-26일짜**

**결석시간 입력 3**

**테스트 절차:**

* 1. **학생1 계정으로 통합정보에서 출석조회 클릭**
  2. **수업에서 java클릭 출석데이터의 결석 날짜가 출석조회창에 출력값과 일치하는 지 확인한다.**

**3. 조교1 계정으로 통합정보에서 출석조회 클릭**

**4. 수업에서 java클릭 후 출석데이터의 출석날짜일치 하는 날짜 클릭 후 학생1의 결석시간이 출석데이터값 과 일치하는 지 확인한다.**

**5. 직원1 계정으로 통합정보에서 출석조회 클릭**

**6. 수업에서 java클릭 후 출석데이터의 출석날짜일치 하는 날짜 클릭 후 학생1의 결석시간이 출석데이터값 과 일치하는 지 확인한다.**

**기대 결과:**

**1. 학생이 출석조회가 출석데이터가 정상적으로 출력되는 것을 확인한다.**

**2. 조교,직원이 출석조회가 출석데이터가 정상적으로 출력되는 것을 확인한다.**

**TC-006 게시판 작성**

**요구사항 내역:  
메인 포털 사이트와 연결된 글을 작성할 수 있는 게시판사이트가 만들어져야 한다.**

**교직원들은 게시판에서 게시판 창설을 할 수 있어야한다.**

* **목적:  
  교직원이 게시글 작성을 진행할 때 결합요소가 존재 확인하고 이를 수정한다.**
* **테스트 대상:  
   교직원 액터 게시판 작성 유스케이스**
* **테스트 데이터:**

**조교1 계정  
ID: 11115 PW: 1234 역할: 조교**

**직원1 계정  
ID: 01115 PW: 1234 역할: 직원**

**교수1 계정  
ID: 10115 PW: 1234 역할: 교수 이름: 이순신**

**게시판 데이터:  
이름 테스트 게시판 게시판정보: 테스트용 게시판입니다. 관리자: 이순신**

**게시글 데이터:  
제목: 테스트 게시글입니다. 내용: 테스트 내용입니다.**

**테스트 절차:**

* 1. **교수1으로 포털사이트에서 게시판 메뉴 클릭**
  2. **게시판에서 게시판 창설 클릭**

**3. 게시판 창설에서 게시판 데이터대로 입력 진행 후 등록 버튼 클릭**

**4. 게시판 탭에서 입력한 게시판이 존재하는 지 확인**

**5. 테스트 게시판 클릭 후 게시판정보가 입력값과 동일한지 확인**

**6. 테스트 게시판에서 게시글 작성 클릭 게시글 데이터 입력한다.**

**7. 게시글 조회하여 제대로 일치하는 확인한다.**

* **기대 결과:**
  1. **게시판 창설이 정상적으로 이루어진다**

**2. 게시판에서 게시글을 작성이 제대로 이뤄지는 확인한다.**

**3. 비기능 요구사항 테스트 계획 및 시나리오**

**TC-006 반응시간**

* **요구사항 내역:  
  버튼을 클릭하였을 때 웹사이트가 2초 이내로 반응하여야 하며 길어도 4초 이내로 반응이 와야 한다.**
* **목적:  
  포털사이트에 주요 버튼(통합정보 버튼,수강신청 버튼, 성적조회 버튼) 클릭하였을때 반응 시간을 확인한다.**
* **테스트 대상:  
   성능 요구사항 반응**
* **테스트 데이터**

**학생1 정보:  
ID: 11101001 PW: 1234 역할: 학생**

* **테스트 절차:**
  1. **포털사이트에서 통합정보버튼 클릭**
  2. **lighthouse에서 First Contentful Paint시간 측정**
  3. **포털사이트에서 수강신청버튼 클릭**
  4. **lighthouse에서 First Contentful Paint시간 측정**
  5. **포털사이트에서 성적조회 클릭**
  6. **lighthouse에서 First Contentful Paint시간 측정**
* **기대 결과:**
  1. **수강신청, 통합정보, 성적조회 버튼을 클릭 시간에 평균이 2초이내이다.**

**TC-006 처리율**

* **요구사항 내역:  
  시스템은 피크 타임(Peak Time)에서 1초당 10건 요청을 안정적으로 처리할 수 있어야 한다.**
* **목적:  
  JMeter를 통해 1초당 10건을 처리할 수 있는지 확인하여 향우 방향성을 결정한다.**
* **테스트 대상:  
   성능 요구사항 반응**
* **테스트 데이터**

**학생1 정보:  
ID: 11101001 PW: 1234 역할: 학생**

* **테스트 절차:**
  1. **JMeter를 실행하여 서버 및 네트워크 환경을 점검한다.**
  2. **Thread Group(스레드 그룹) 또는 Throughput Controller를 이용해 초당 10건의 요청이 발생하도록 설정한다.**
  3. **du.potal.com으로 url을 설정한다.**
  4. **결과 기록을 위한 리스너(그래프, 요약 리포트 등)를 추가한다.**
  5. **적은 부하(예: 초당 10~20건)로 사전 테스트를 실행하여 시스템 및 스크립트 동작을 확인한다.**
  6. **정상 응답 및 오류 발생 여부를 확인한다.**
* **기대 결과:**
  1. **1초당에 10건에 요청에 대해 안정적으로 수용한다.**

**TC-006 하드디스크**

* **요구사항 내역:  
  학사관리 시스템 운영에 필요한 최소 30GB (테라바이트) 이상의 사용 가능한 저장 공간을 확보해야 한다. 이는 시스템 데이터, 로그, 백업 등을 포함하는 용량이다.**
* **목적:  
  클라우드를 통한 30GB이 원할하게 이루어지는 확인한다.**
* **테스트 대상:  
   성능 요구사항 반응**
* **테스트 데이터**

**학생1 정보:  
ID: 11101001 PW: 1234 역할: 학생**

* **테스트 절차:**
  1. **클라우드 관리 콘솔접속**
  2. **서버/인스턴스/스토리지 서비스의 ‘할당 용량’ 정보가 30GB로 세팅되어 있는지 확인한다.**
  3. **임의 파일을 생성하여 30GB에 근접할 때까지 저장 시도하여 업로드중 성능하락이 있는지 확인한다.**
  4. **30GB를 초과하여 용량 업로드 진행**
  5. **임의 파일을 생성하여 30GB에 근접할 때까지 저장 시도 시**
  6. **“용량 초과” 안내 또는 오류 메시지가 적절히 출력되는지 확인한다.**
  7. **콘솔에서 실제로 용량이 30GB가 되는지 확인한다.**
* **기대 결과:**
  1. **클라우드에서 30GB에 해당하는 용량이 제공된다.**

**TC-006 안정성**

* **요구사항 내역:  
  서버가 점검시간 오전 4시부터 5시까지를 제외한 나머지 시간에는 서버가 안정인 상태를 유지해야 한다**
* **목적:  
  실제 24시간 모니터링이 어려우므로,  
  점검 시간(오전 4~5시) 전후, 일반 운영 시간(예: 오전 3:55, 4:05, 5:01, 6:00 등)에 테스트를 여러 번 반복하여 안정성을 확인한다.**
* **테스트 대상:  
   비기능 요구사항 안정성 부분**
* **테스트 데이터**

**학생1 정보:  
ID: 11101001 PW: 1234 역할: 학생**

* **테스트 절차:**
  1. **접속시간 전인 3:55분에 웹사이트에 직접 접속한다.**
  2. **이후 오전 4:00, 4:30, 5:00 등 점검 시간대에 접속을 시도한다.**
  3. **“점검중” 메시지를 확인한다.**
  4. **오전 5:01에 웹사이트 또는 서비스에 다시 접속한다.**
  5. **정상적으로 서버에 접속되는지 확인한다.**
* **기대 결과:**
  1. **점검시간일때 접속이 차단된다.**
  2. **점검시간이후에 접속이 되는 지 확인한다.**

**TC-006 안정성**

* **요구사항 내역:  
  동서울 대학교 정원인 8051명이기에 이에 1/4인 2000명 정도는 수용할 수 있을 정도의 서버가 구축되어야 한다.**
* **목적:  
  서버가 약, 2000명 이상의 동시 접속 사용자를 안정적으로 수용할 수 있는지를 확인한다.**
* **테스트 대상 :  
  비 기능 요구사항 트래픽 수용도 부분**
* **테스트 데이터 :  
  1.비용 문제로 인해 실제 수용도는 20명 정도의 동시 접속 사용자   
  2.사용자 행위 시나리오 : 로그인, 강의조회, 수강신청 페이지 이동, 로그아웃 등 시나리오별 요청 흐름  
  3. 테스트 시간 : 각 부하 단계마다 5~10분 지속  
  4. 응답 시간 기준값 : 2초 이내 정상, 3초 이상 경고, 5초 이상 오류로 판단**
* **테스트 절차 :   
  1. 테스트를 수행할 시스템과 서버 환경을 점검한다.  
  2. 테스트에 사용할 시나리오를 설정한다.  
  (예: 로그인 -> 주요 페이지 접속 -> 로그아웃)  
  3. 동시 접속 사용자 수를 1명부터 점진적으로 증가시킨다  
  (예: 1 -> 5 -> 10 -> 15 -> 20)  
  4. 각 접속 단계별로 서버의 응답 상태 및 시스템 부하를 관찰한다.  
  (예: 화면 응답속도, 페이지 전환 속도, 오류 발생 여부 등)  
  5. 최대 동시 접속자 수를 기준으로 최종 테스트를 진행한다.  
  6. 테스트 결과를 기록하고, 응답 시간 및 오류 발생률 등을 분석한다.  
  7. 테스트 종료 후 시스템을 초기화하고 정리한다.**
* **기대 결과:**

**1. 시스템이 정상적으로 작동 가능하며, 테스트 준비가 완료된다.  
2. 시나리오가 명확하게 정의되어 테스트 흐름을 따를 수 있어야한다.  
3. 각 단계에서 시스템이 안정적으로 반응해야 한다.  
4. 모든 사용자가 정상적으로 서비스를 이용할 수 있어야한다.  
5. 서버가 다운되자 않고, 서비스 지연이나 오류 없이 작동해야 한다.  
6. 시스템이 요구되는 트래픽 수준을 수용 가능한지 판단할 수 있어야한다.  
7. 서버가 정상 상태로 복구되고, 테스트 환경이 정리된다.**

**12. 소스코드**

**MainContent.js 파일 소스코드**

**파일 설명: 해당 파일은 메인 페이지의 소스코드로 해더와 메인페이지의 요소들을 임포트 하며 메시지를 임포트 받아서 메인 페이지의 출력하는 로직 두개의 파트로 구성되어 있다.**

|  |
| --- |
| **"use client";**  **import { useEffect, useState } from "react";**  **import Messenger from "../messeage\_component/Messenger\_home/MessengerContainer";**  **import LoginBox from "./MainComponent/profile";**  **import QuickMenu from "./MainComponent/QuickMenu";**  **import LectureSchedule from "./MainComponent/lecture\_schedule";**  **import Shortcuts from "./MainComponent/Shortcuts";**  **import News from "./MainComponent/News";**  **import Calendar from "./MainComponent/Calendar";**  **import Community from "./MainComponent/Community";**  **import Timetable from "../timetable/timetable\_content";**  **import MessengerNotificationPanel from "./MainComponent/MessengerNotificationPanel";**  **// 메신저 모달 컴포넌트 (x\_to\_messenger)**  **import MessengerModal from "../messeage\_component/Messenger\_chatwindow/x\_to\_messenger";**  **// ✅ 새 채팅방 생성 모달(1번 창)**  **// 경로를 실제 파일 위치에 맞게! (Messenger\_home 이 아니라 Messenger\_chatwindow 라면 이렇게)**  **import CreateChatWindowContainer from "../messeage\_component/Messenger\_home/create\_chatwindow\_container";**  **export default function MainContent() {**  **const [showMessenger, setShowMessenger] = useState(false);**  **// ✅ 1번 창(새 채팅방 만들기) 모달 표시 여부**  **const [showCreateChatWindow, setShowCreateChatWindow] = useState(false);**  **const [profileData, setProfileData] = useState(null);**  **const [loading, setLoading] = useState(true);**  **const [fetchError, setFetchError] = useState(null);**  **const [lastLoginTime, setLastLoginTime] = useState("");**  **const v = (x) =>**  **x === null || x === undefined || x === "" ? "-" : String(x);**  **const formatDate = (iso) => {**  **if (!iso) return "-";**  **const d = new Date(iso);**  **if (isNaN(d)) return v(iso);**  **const y = d.getFullYear();**  **const m = String(d.getMonth() + 1).padStart(2, "0");**  **const day = String(d.getDate()).padStart(2, "0");**  **return `${y}.${m}.${day}`;**  **};**  **const formatGender = (g) => {**  **if (!g) return "-";**  **const s = String(g).toUpperCase();**  **if (s === "M") return "남";**  **if (s === "F") return "여";**  **return s;**  **};**  **const formatPhone = (p) => {**  **if (!p) return "-";**  **const d = String(p).replace(/\D/g, "");**  **if (d.length === 11 && d.startsWith("010")) {**  **return `010-${d.slice(3, 7)}-${d.slice(7)}`;**  **}**  **return p;**  **};**  **// 마지막 로그인 시간 표시용**  **useEffect(() => {**  **const d = new Date();**  **const pad = (n) => String(n).padStart(2, "0");**  **setLastLoginTime(**  **`${d.getFullYear()}.${pad(d.getMonth() + 1)}.${pad(**  **d.getDate()**  **)} ${pad(d.getHours())}:${pad(d.getMinutes())}`**  **);**  **}, []);**  **// 프로필 데이터 fetch**  **useEffect(() => {**  **let ignore = false;**  **(async () => {**  **setLoading(true);**  **setFetchError(null);**  **try {**  **const res = await fetch(**  **"http://localhost:3000/api/univer\_city/user\_route",**  **{ cache: "no-store" }**  **);**  **const data = await res.json().catch(() => ({}));**  **if (!res.ok || data?.ok === false) {**  **if (!ignore) {**  **setFetchError(**  **data?.reason || data?.error || `HTTP\_${res.status}`**  **);**  **}**  **return;**  **}**  **const result = data?.result;**  **let row = null;**  **if (Array.isArray(result) && result.length) row = result[0];**  **else if (Array.isArray(result?.rows) && result.rows.length)**  **row = result.rows[0];**  **else if (Array.isArray(result?.data) && result.data.length)**  **row = result.data[0];**  **else if (result && typeof result === "object") row = result;**  **if (!row) {**  **if (!ignore) setProfileData(null);**  **return;**  **}**  **const cleaned = {**  **user\_id: v(row.user\_id),**  **user\_name: v(row.user\_name),**  **department\_id: v(row.department\_id),**  **birth\_date: formatDate(row.birth\_date),**  **gender: formatGender(row.gender),**  **phone: formatPhone(row.phone),**  **address: v(row.address),**  **account\_number: v(row.account\_number),**  **};**  **const typeVal = data?.user\_type ?? "unknown";**  **const typeSrc = data?.user\_type\_source ?? null;**  **const raw = typeSrc?.raw ?? null;**  **cleaned.user\_type = typeVal;**  **cleaned.user\_type\_foundIn = typeSrc?.foundIn ?? null;**  **let subtypeRecord = null;**  **if (raw && typeof raw === "object") {**  **const tableName = raw.user\_type;**  **if (**  **tableName &&**  **raw[tableName] &&**  **typeof raw[tableName] === "object"**  **) {**  **subtypeRecord = raw[tableName];**  **}**  **}**  **cleaned.user\_type\_record = subtypeRecord;**  **if (!ignore) setProfileData(cleaned);**  **} catch (e) {**  **if (!ignore) setFetchError(e?.message || "UNKNOWN\_ERROR");**  **} finally {**  **if (!ignore) setLoading(false);**  **}**  **})();**  **return () => {**  **ignore = true;**  **};**  **}, []);**  **// 채팅방 뷰 → 메신저 패널로 돌아올 때 쓰는 이벤트**  **useEffect(() => {**  **if (typeof window === "undefined") return;**  **const handler = () => {**  **setShowMessenger(true);**  **};**  **window.addEventListener("OPEN\_MESSENGER\_FROM\_CHAT", handler);**  **return () => {**  **window.removeEventListener("OPEN\_MESSENGER\_FROM\_CHAT", handler);**  **};**  **}, []);**  **// ✅ 어디서든 `OPEN\_CREATE\_CHAT\_WINDOW` 이벤트만 쏘면**  **// 여기서 1번 창(새 채팅방 만들기) 모달을 띄움**  **useEffect(() => {**  **if (typeof window === "undefined") return;**  **const handler = () => {**  **setShowCreateChatWindow(true);**  **};**  **window.addEventListener("OPEN\_CREATE\_CHAT\_WINDOW", handler);**  **return () => {**  **window.removeEventListener("OPEN\_CREATE\_CHAT\_WINDOW", handler);**  **};**  **}, []);**  **const currentUserId =**  **profileData && profileData.user\_id && profileData.user\_id !== "-"**  **? profileData.user\_id**  **: null;**  **return (**  **<>**  **<LoginBox**  **data={profileData}**  **loading={loading}**  **fetchError={fetchError}**  **onOpenMessenger={() => setShowMessenger(true)}**  **showDetails={false}**  **lastLoginTime={lastLoginTime}**  **userType={profileData?.user\_type ?? null}**  **/>**  **<Shortcuts />**  **<QuickMenu />**  **<Calendar />**  **<div className="card">**  **<Timetable />**  **</div>**  **<MessengerNotificationPanel />**  **{showMessenger && (**  **<MessengerModal onClose={() => setShowMessenger(false)} />**  **)}**  **{/\* ✅ 새 채팅방 만들기 모달(1번 창) \*/}**  **{showCreateChatWindow && (**  **<CreateChatWindowContainer**  **currentUserId={currentUserId}**  **onClose={() => setShowCreateChatWindow(false)}**  **onCreated={() => {**  **setShowCreateChatWindow(false);**  **setShowMessenger(true);**  **}}**  **/>**  **)}**  **</>**  **);**  **}** |

|  |
| --- |
| */\*attendance\_input.js \*/*  'use client'  import { useState, useEffect } from 'react';  function AttendanceContent() {    const [selectedClass, setSelectedClass] = useState('컴소과 2-1 자료구조');    const [todayDate, setTodayDate] = useState('');    useEffect(() => {      const today = new Date();      const y = today.getFullYear();      const m = String(today.getMonth() + 1).padStart(2, '0');      const d = String(today.getDate()).padStart(2, '0');      setTodayDate(`오늘 ${y}-${m}-${d}`);    }, []);    const toast = (msg) => alert(msg);    const WEEK\_START = '2025-03-01';    const WEEK\_COUNT = 16;    const addDays = (date, days) => {      const d = new Date(date);      d.setDate(d.getDate() + days);      return `${d.getFullYear()}-${String(d.getMonth() + 1).padStart(2, '0')}-${String(        d.getDate()      ).padStart(2, '0')}`;    };    const weeks = Array.from({ length: WEEK\_COUNT }, (\_, i) => {      const dateStr = addDays(WEEK\_START, i \* 7);      return (        <div key={i} className={styles.weekItem}>          <span>{i + 1}주차</span>          <span className={styles.date}>({dateStr})</span>        </div>      );    });    return (      <div className={styles.layout}>        <main className={styles.main}>          <header className={styles.topbar}>            <div className={styles.left}>              <div className={styles.crumbs}>                출석관리 &gt; <strong>출석 입력</strong>              </div>              <div className={styles.titleRow}>                <h1 className={styles.title}>출석 입력</h1>                <select                  className={styles.classSelect}                  onChange={(e) => setSelectedClass(e.target.value)}                  value={selectedClass}                >                  <option>컴소과 2-1 자료구조</option>                  <option>컴소과 1-2 웹프로그래밍</option>                  <option>컴소과 3-1 운영체제</option>                </select>              </div>            </div>            <div className={styles.right}>              <div className={styles.todayDate}>{todayDate}</div>            </div>          </header>          <section className={styles.grid}>            <aside className={styles.toolCol}>              <div className={styles.card}>                <h3>필터</h3>                <label className={styles.field}>                  <span>상태</span>                  <select className={styles.formInput}>                    {*/\* formInput 클래스 적용 \*/*}                    <option>전체</option>                    <option>출석</option>                    <option>지각</option>                    <option>조퇴</option>                    <option>결석</option>                  </select>                </label>                <label className={styles.field}>                  <span>이름/학번 검색</span>                  <input type="text" placeholder="예: 20231234 / 홍길동" />                </label>              </div>              <div className={styles.card}>                <h3>메모</h3>                <textarea rows="6" placeholder="수업 특이사항, 과제 공지 등"></textarea>                <div className={styles.rightRow}>                  <button                    className={`${styles.btn} ${styles.ghost}`}                    type="button"                    onClick={() => toast('메모가 임시저장되었습니다.')}                  >                    임시저장                  </button>                </div>              </div>            </aside>            <div className={styles.centerPlaceholder}>              <div className={styles.placeholderBox}>중앙 출석표 컴포넌트 영역</div>            </div>            <aside className={styles.infoCol}>              <div className={styles.card}>                <h3>주차 선택</h3>                <div className={styles.weeks}>{weeks}</div>              </div>              <div className={styles.card}>                <h3>오늘 수업정보</h3>                <dl className={styles.desc}>                  <div>                    <dt>강의명</dt>                    <dd>{selectedClass}</dd>                  </div>                  <div>                    <dt>담당</dt>                    <dd>김교수</dd>                  </div>                  <div>                    <dt>정원/수강</dt>                    <dd>40 / 36</dd>                  </div>                  <div>                    <dt>평균 출석률</dt>                    <dd>92%</dd>                  </div>                </dl>                <div className={styles.legend}>                  <span className={`${styles.pill} ${styles.ok}`}>출석</span>                  <span className={`${styles.pill} ${styles.late}`}>지각</span>                  <span className={`${styles.pill} ${styles.leave}`}>조퇴</span>                  <span className={`${styles.pill} ${styles.absent}`}>결석</span>                </div>              </div>              <div className={styles.card}>                <h3>최근 공지</h3>                <ul className={styles.list}>                  <li>9/10: 팀프로젝트 조 편성 완료</li>                  <li>9/12: 실습실 변경 B-205</li>                  <li>9/15: 퀴즈 #2 공지</li>                </ul>              </div>            </aside>          </section>          <footer className={styles.bottomBar}>            <div className={styles.leftRow}>              <span className={styles.muted}>자동저장 간격 30초</span>            </div>            <div className={styles.rightRow}>              <button                className={`${styles.btn} ${styles.ghost}`}                type="button"                onClick={() => toast('마지막 저장 상태로 원복했습니다.')}              >                원복              </button>              <button                className={styles.btn}                type="button"                onClick={() => toast('임시저장 완료')}              >                임시저장              </button>              <button                className={`${styles.btn} ${styles.primary}`}                type="button"                onClick={() => toast('제출/저장되었습니다.')}              >                제출/저장              </button>            </div>          </footer>        </main>      </div>    );  } |

|  |
| --- |
| *// app/main\_content/components/personal\_information/Personal\_Information\_Form.js*  "use client";  import { useEffect, useState } from "react";  function normalizeUserFromApi(data) {    if (!data || !data.result) return null;    const r = data.result;  *// 1) result가 배열인 경우*    if (Array.isArray(r) && r.length > 0) {      return r[0];    }  *// 2) result.rows[0] 형태인 경우*    if (r && Array.isArray(r.rows) && r.rows.length > 0) {      return r.rows[0];    }  *// 3) 그 외: 그냥 result 자체가 row인 경우라고 가정*    return r;  }  export default function Personal\_Information\_Form() {    const [loading, setLoading] = useState(true);    const [loadError, setLoadError] = useState(null);    const [user, setUser] = useState(null);    const [phone, setPhone] = useState("");    const [address, setAddress] = useState("");    const [saving, setSaving] = useState(false);    const [saveError, setSaveError] = useState(null);    const [saveSuccess, setSaveSuccess] = useState(false);  *// ---------- 1) 최초 로딩: 현재 사용자 정보 가져오기 ----------*    useEffect(() => {      const fetchUser = async () => {        try {          setLoading(true);          setLoadError(null);  *// 현재 세션 기반 사용자 정보 라우터*          const res = await fetch("/api/univer\_city/user\_route", {            method: "GET",            cache: "no-store",          });          const data = await res.json();          if (!res.*ok* || data.ok === false) {            throw new Error(              data?.reason || "사용자 정보를 불러오는데 실패했습니다."            );          }          const row = normalizeUserFromApi(data);          if (!row) {            throw new Error("사용자 정보가 비어 있습니다.");          }          setUser(row);          setPhone(row.phone || "");          setAddress(row.address || "");        } catch (e) {          console.error(e);          setLoadError(e.message || String(e));        } finally {          setLoading(false);        }      };      fetchUser();    }, []);  *// ---------- 2) 전화번호 간단 검증 ----------*    const phoneRegex = /^[0-9]{3}-[0-9]{4}-[0-9]{4}$/;    const isPhoneValid = phoneRegex.test(phone);  *// ---------- 3) 저장 버튼 클릭 시 ----------*    const handleSubmit = async (e) => {      e.preventDefault();      setSaveSuccess(false);      setSaveError(null);      if (!user) {        setSaveError("사용자 정보가 없습니다.");        return;      }      if (!isPhoneValid) {        setSaveError("전화번호 형식이 올바르지 않습니다. 예: 010-1234-5678");        return;      }      try {        setSaving(true);        const updateBody = {          table: "user",          set: {            phone,            address,          },          where: `user\_id = ${user.user\_id}`,        };        const res = await fetch("/api/univer\_city/update\_where\_route", {          method: "POST",          headers: {            "Content-Type": "application/json",          },          body: JSON.stringify(updateBody),        });        const data = await res.json().catch(() => ({}));        if (!res.*ok* || data.ok === false) {          throw new Error(            data?.reason || "개인정보를 수정하는 데 실패했습니다."          );        }        setSaveSuccess(true);  *// 로컬 user 상태도 업데이트*        setUser((prev) =>          prev            ? {                ...prev,                phone,                address,              }            : prev        );      } catch (e) {        console.error(e);        setSaveError(e.message || String(e));      } finally {        setSaving(false);      }    };  *// ---------- 4) 렌더링 ----------*    if (loading) {      return <div>개인정보를 불러오는 중입니다...</div>;    }    if (loadError) {      return <div style={{ color: "red" }}>오류: {loadError}</div>;    }    if (!user) {      return <div>사용자 정보를 찾을 수 없습니다.</div>;    }    return (      <div        style={{          maxWidth: "600px",          margin: "0 auto",          padding: "16px",          border: "1px solid #ddd",          borderRadius: "8px",          backgroundColor: "#fff",        }}      >        <h2 style={{ marginBottom: "16px" }}>개인정보 수정</h2>        <form onSubmit={handleSubmit}>          {*/\* 읽기 전용 기본 정보들 \*/*}          <div style={{ marginBottom: "12px" }}>            <label              style={{ display: "block", fontWeight: "bold", marginBottom: 4 }}            >              사용자 ID            </label>            <input              type="text"              value={user.user\_id ?? ""}              disabled              style={{ width: "100%", padding: "8px" }}            />          </div>          <div style={{ marginBottom: "12px" }}>            <label              style={{ display: "block", fontWeight: "bold", marginBottom: 4 }}            >              이름            </label>            <input              type="text"              value={user.user\_name ?? ""}              disabled              style={{ width: "100%", padding: "8px" }}            />          </div>          <div style={{ marginBottom: "12px" }}>            <label              style={{ display: "block", fontWeight: "bold", marginBottom: 4 }}            >              생년월일            </label>            <input              type="text"              value={user.birth\_date ?? ""}              disabled              style={{ width: "100%", padding: "8px" }}            />          </div>          <div style={{ marginBottom: "12px" }}>            <label              style={{ display: "block", fontWeight: "bold", marginBottom: 4 }}            >              성별            </label>            <input              type="text"              value={user.gender ?? ""}              disabled              style={{ width: "100%", padding: "8px" }}            />          </div>          <div style={{ marginBottom: "12px" }}>            <label              style={{ display: "block", fontWeight: "bold", marginBottom: 4 }}            >              전화번호            </label>            <input              type="text"              value={phone}              onChange={(e) => setPhone(e.target.value)}              placeholder="010-1234-5678"              style={{ width: "100%", padding: "8px" }}            />            {!isPhoneValid && phone.*length* > 0 && (              <div style={{ color: "red", fontSize: "12px", marginTop: "4px" }}>                전화번호 형식이 올바르지 않습니다. 예: 010-1234-5678              </div>            )}          </div>          {*/\* 수정 가능: 주소 \*/*}          <div style={{ marginBottom: "12px" }}>            <label              style={{ display: "block", fontWeight: "bold", marginBottom: 4 }}            >              주소            </label>            <input              type="text"              value={address}              onChange={(e) => setAddress(e.target.value)}              placeholder="주소를 입력하세요"              style={{ width: "100%", padding: "8px" }}            />          </div>          <div style={{ marginBottom: "12px" }}>            <label              style={{ display: "block", fontWeight: "bold", marginBottom: 4 }}            >              계좌번호            </label>            <input              type="text"              value={user.account\_number ?? ""}              disabled              style={{ width: "100%", padding: "8px" }}            />          </div>          {*/\* 저장 결과 메시지 \*/*}          {saveError && (            <div style={{ color: "red", marginBottom: "8px" }}>{saveError}</div>          )}          {saveSuccess && (            <div style={{ color: "green", marginBottom: "8px" }}>              개인정보가 수정되었습니다.            </div>          )}          <button            type="submit"            disabled={saving}            style={{              padding: "8px 16px",              fontWeight: "bold",              borderRadius: "4px",              border: "none",              backgroundColor: saving ? "#999" : "#1e40af",              color: "#fff",              cursor: saving ? "default" : "pointer",            }}          >            {saving ? "저장 중..." : "저장"}          </button>        </form>      </div>    );  } |

|  |
| --- |
| 'use client';  import { useState, useEffect, useMemo } from 'react';  import styles from '../css/page.module.css';  *// 🔻 default export 로 변경*  export default function GradesExportContent() {    const [students, setStudents] = useState([]);    const [today, setToday] = useState('');    const [error, setError] = useState(null);    const [loading, setLoading] = useState(true);    const [selectedSemester, setSelectedSemester] = useState('');    const [selectedSubject, setSelectedSubject] = useState('');    useEffect(() => {      setToday(new Date().toLocaleDateString('ko-KR'));    }, []);    useEffect(() => {      const fetchGrades = async () => {        try {          setLoading(true);          const query = selectedSemester            ? `/api/univer\_city/select\_grade\_route?semester=${                selectedSemester.includes('2학기') ? 2 : 1              }`            : `/api/univer\_city/select\_grade\_route`;          const res = await fetch(query);          const data = await res.json();          if (!res.*ok*) throw new Error(data.error || 'DB 불러오기 실패');          const formatted = data.map((row, idx) => ({            id: idx + 1,            studentId: row.student\_id,            name: row.student\_name,            department: row.department\_name,            subject: row.lecture\_name,            professor: row.professor\_name,            grade: row.grade || '-',            year: row.lecture\_year || '2025',            semester: row.lecture\_semester === 1 ? '1학기' : '2학기',          }));          setStudents(formatted);        } catch (err) {          console.error('❌ 성적 데이터 불러오기 오류:', err);          setError('DB에서 데이터를 불러오지 못했습니다.');        } finally {          setLoading(false);        }      };      fetchGrades();    }, [selectedSemester]);    const semesterOptions = useMemo(() => {      const years = new Set(students.map((s) => s.year || '2025'));      const options = [];      years.forEach((year) => {        options.push(`${year}-1학기`);        options.push(`${year}-2학기`);      });      if (options.length === 0) {        options.push('2025-1학기');        options.push('2025-2학기');      }      return Array.from(new Set(options)).sort();    }, [students]);    const filteredBySemester = useMemo(() => {      if (!selectedSemester) return students;      const [year, semText] = selectedSemester.split('-');      const sem = semText.includes('2') ? '2학기' : '1학기';      return students.filter(        (s) => s.year.toString() === year && s.semester === sem      );    }, [selectedSemester, students]);    const filteredStudents = useMemo(() => {      if (!selectedSubject) return filteredBySemester;      return filteredBySemester.filter((s) => s.subject === selectedSubject);    }, [filteredBySemester, selectedSubject]);    const subjects = useMemo(() => {      const unique = [...new Set(filteredBySemester.map((s) => s.subject))];      return unique;    }, [filteredBySemester]);    if (loading)      return <p style={{ textAlign: 'center' }}>📦 데이터를 불러오는 중...</p>;    if (error)      return <p style={{ color: 'red', textAlign: 'center' }}>{error}</p>;    return (      <div className={styles.gradeContainer}>        <div className={styles.header}>          <h2>성적 출력</h2>          <div className={styles.date}>{today}</div>        </div>        <div className={styles.filters}>          <label style={{ display: 'flex', alignItems: 'center', gap: '4px' }}>            학기 선택:            <select              value={selectedSemester}              onChange={(e) => setSelectedSemester(e.target.value)}            >              <option value="">전체 학기</option>              {semesterOptions.map((sem, i) => (                <option key={i} value={sem}>                  {sem}                </option>              ))}            </select>          </label>          <label style={{ display: 'flex', alignItems: 'center', gap: '4px' }}>            과목 선택:            <select              value={selectedSubject}              onChange={(e) => setSelectedSubject(e.target.value)}            >              <option value="">전체 과목</option>              {subjects.map((subject, i) => (                <option key={i} value={subject}>                  {subject}                </option>              ))}            </select>          </label>          <input            type="text"            placeholder="학번 또는 이름 검색"            className={styles.searchInput}          />        </div>        <table className={styles.gradeTable}>          <thead>            <tr>              <th>학번</th>              <th>이름</th>              <th>학과</th>              <th>과목명</th>              <th>교수</th>              <th>등급</th>              <th>학기</th>            </tr>          </thead>          <tbody>            {filteredStudents.length === 0 ? (              <tr>                <td colSpan="7" style={{ textAlign: 'center', padding: '20px' }}>                  선택된 학기 또는 과목에 해당하는 성적이 없습니다.                </td>              </tr>            ) : (              filteredStudents.map((student) => (                <tr key={student.id}>                  <td>{student.studentId}</td>                  <td>{student.name}</td>                  <td>{student.department}</td>                  <td>{student.subject}</td>                  <td>{student.professor}</td>                  <td>{student.grade}</td>                  <td>{`${student.year} ${student.semester}`}</td>                </tr>              ))            )}          </tbody>        </table>        <div className={styles.toolbar}>          <button onClick={() => window.location.reload()}>새로고침</button>          <button onClick={() => window.print()}>인쇄</button>        </div>      </div>    );  } |

|  |
| --- |
| 'use client';  import { useState, useEffect } from 'react';  import styles from '../Integrated/attendance-input/style.module.css';  import AttendanceCard from '../Integrated/attendance-input/AttendanceCard';  import AttendanceCheck from '../Integrated/attendance-check/AttendanceCheck';  export default function AttendanceContent() {    const [selectedClass, setSelectedClass] = useState('컴소과 2-1 자료구조');    const [todayDate, setTodayDate] = useState('');    const [students, setStudents] = useState([]);    const [allAttendanceData, setAllAttendanceData] = useState({});    const [filterMode, setFilterMode] = useState('전체');    const handleStudentStatusChange = (studentId, status, memo, lateReason) => {      setAllAttendanceData(prevData => ({          ...prevData,          [studentId]: {              status,              memo,              lateReason,          }      }));  };    useEffect(() => {      const today = new Date();      const y = today.getFullYear();      const m = String(today.getMonth() + 1).padStart(2, '0');      const d = String(today.getDate()).padStart(2, '0');      setTodayDate(`오늘 ${y}-${m}-${d}`);      const LECTURE\_ID = 1; *// 💡 [필수 수정] 실제 강의 ID로 대체하세요.*      const ATTENDANCE\_DATE = `${y}-${m}-${d}`;      const fetchAttendanceData = async () => {          try {              const API\_URL = `/api/univer\_city/attendance\_data?lectureId=${LECTURE\_ID}&date=${ATTENDANCE\_DATE}`;              const response = await fetch(API\_URL);              if (!response.*ok*) {                    throw new Error('출석 데이터를 불러오는 데 실패했습니다.');              }              const fetchedStudents = await response.json();                const initialData = {};              fetchedStudents.forEach(s => {                  const uniqueStudentId = s.studentId || s.user\_id;                  initialData[uniqueStudentId] = {                      status: s.attendance\_status || '미처리',                      memo: s.memo || '',                      lateReason: s.late\_reason || '',                  };              });                setStudents(fetchedStudents);              setAllAttendanceData(initialData);            } catch (error) {              console.error('출석 데이터 로드 오류:', error);  *// 사용자에게 오류 메시지 표시*              toast('출석 데이터를 불러오는 중 오류가 발생했습니다.');          }      };        fetchAttendanceData();    }, []);    const toast = (msg) => alert(msg);    const WEEK\_START = '2025-03-01';    const WEEK\_COUNT = 16;    const addDays = (date, days) => {      const d = new Date(date);      d.setDate(d.getDate() + days);      return `${d.getFullYear()}-${String(d.getMonth() + 1).padStart(2, '0')}-${String(        d.getDate()      ).padStart(2, '0')}`;    };    const weeks = Array.from({ length: WEEK\_COUNT }, (\_, i) => {      const dateStr = addDays(WEEK\_START, i \* 7);      return (        <div key={i} className={styles.weekItem}>          <span>{i + 1}주차</span>          <span className={styles.date}>({dateStr})</span>        </div>      );    });    const handleSubmit = async () => {      const LECTURE\_ID = 1;      const ATTENDANCE\_DATE = todayDate.split(' ')[1];      const attendanceRecords = Object.keys(allAttendanceData).map(studentId => ({          studentId: studentId,          ...allAttendanceData[studentId]      }));      const payload = {          lectureId: LECTURE\_ID,          attendanceDate: ATTENDANCE\_DATE,          attendanceData: attendanceRecords      };        try {          const response = await fetch('/api/univer\_city/insert\_attendance', {              method: 'POST',              headers: { 'Content-Type': 'application/json' },              body: JSON.stringify(payload),          });          if (response.*ok*) {              toast('제출/저장되었습니다.');          } else {              const errorData = await response.json();              throw new Error(errorData.message || '데이터 저장 실패');          }      } catch (error) {          console.error('출석 데이터 저장 오류:', error);          toast(`저장에 실패했습니다: ${error.message}`);      }  };      return (      <div className={styles.layout}>        <main className={styles.main}>          <header className={styles.topbar}>            <div className={styles.left}>              <div className={styles.crumbs}>                출석관리 &gt; <strong>출석 입력</strong>              </div>              <div className={styles.titleRow}>                <h1 className={styles.title}>출석 입력</h1>                <select                  className={styles.classSelect}                  onChange={(e) => setSelectedClass(e.target.value)}                  value={selectedClass}                >                  <option>컴소과 2-1 자료구조</option>                  <option>컴소과 1-2 웹프로그래밍</option>                  <option>컴소과 3-1 운영체제</option>                </select>              </div>            </div>            <div className={styles.right}>              <div className={styles.todayDate}>{todayDate}</div>            </div>          </header>          <section className={styles.grid}>            <aside className={styles.toolCol}>              <div className={styles.card}>                <h3>출석상태</h3>                <label className={styles.field}>                  <span>상태</span>                  <select className={styles.formInput}>                    <option>전체</option>                    <option>출석</option>                    <option>지각</option>                    <option>조퇴</option>                    <option>결석</option>                  </select>                </label>                <label className={styles.field}>                  <span>이름/학번 검색</span>                  <input type="text" placeholder="예: 20231234 / 홍길동" />                </label>              </div>              <div className={styles.card}>                <h3>메모</h3>                <textarea rows="6" placeholder="수업 특이사항, 과제 공지 등"></textarea>                <div className={styles.rightRow}>                  <button                    className={`${styles.btn} ${styles.ghost}`}                    type="button"                    onClick={() => toast('메모가 임시저장되었습니다.')}                  >                    임시저장                  </button>                </div>              </div>            </aside>            <div className={styles.cards}>                {students.length > 0 ? (                    students.map((student) => (                      <AttendanceCard                          key={student.id}                          student={student}                          onStatusChange={handleStudentStatusChange}                      />                  ))              ) : (                  <div className={styles.placeholderBox}>                      {filterMode === '결석자'                       ? '결석 또는 미처리 상태인 학생이 없습니다.'                       : '학생 데이터가 없습니다.'}                  </div>                )}              </div>            <aside className={styles.infoCol}>              <div className={styles.card}>                <h3>주차 선택</h3>                <div className={styles.weeks}>{weeks}</div>              </div>              <div className={styles.card}>                <h3>오늘 수업정보</h3>                <dl className={styles.desc}>                  <div>                    <dt>강의명</dt>                    <dd>{selectedClass}</dd>                  </div>                  <div>                    <dt>담당</dt>                    <dd>김교수</dd>                  </div>                  <div>                    <dt>정원/수강</dt>                    <dd>36 / 40</dd>                  </div>                  <div>                    <dt>평균 출석률</dt>                    <dd>92%</dd>                  </div>                </dl>                <div className={styles.legend}>                  <span className={`${styles.pill} ${styles.ok}`}>출석</span>                  <span className={`${styles.pill} ${styles.late}`}>지각</span>                  <span className={`${styles.pill} ${styles.leave}`}>조퇴</span>                  <span className={`${styles.pill} ${styles.absent}`}>결석</span>                </div>              </div>              <div className={styles.card}>                <h3>최근 공지</h3>                <ul className={styles.list}>                  <li>9/10: 팀프로젝트 조 편성 완료</li>                  <li>9/12: 실습실 변경 B-205</li>                  <li>9/15: 퀴즈 #2 공지</li>                </ul>              </div>            </aside>            </section>          <footer className={styles.bottomBar}>            <div className={styles.leftRow}>              <span className={styles.muted}>자동저장 간격 30초</span>            </div>            <div className={styles.rightRow}>              <button                className={`${styles.btn} ${styles.primary}`}                type="button"                onClick={() => toast('마지막 저장 상태로 원복했습니다.')}              >                원복              </button>              <button                className={styles.btn}                type="button"                onClick={() => toast('임시저장 완료')}              >                임시저장              </button>              <button              className={`${styles.btn} ${styles.primary}`}              type="button"              onClick={async () => {                  const ATTENDANCE\_DATE = new Date().toISOString().split('T')[0];                  const LECTURE\_ID = 1;                  const PROFESSOR\_ID = '1';                  const attendanceRecordsArray = Object.keys(allAttendanceData)                      .map(studentId => ({                          studentId: studentId,                          status: allAttendanceData[studentId].status,                          memo: allAttendanceData[studentId].memo,                          lateReason: allAttendanceData[studentId].lateReason,                      }));                    const dataToSave = {                      lectureId: LECTURE\_ID,                      attendanceDate: ATTENDANCE\_DATE,                      professorId: PROFESSOR\_ID,                      records: attendanceRecordsArray                  };                  try {                      const response = await fetch('/api/univer\_city/insert\_attendance', { *// ⭐️ 저장 API 경로*                          method: 'POST',                          headers: {                              'Content-Type': 'application/json',                          },                          body: JSON.stringify(dataToSave),                      });                      if (response.*ok*) {                          alert('✅ 출석 정보가 최종 저장/수정되었습니다.');                      } else {                          const errorData = await response.json();                          alert(`❌ 저장 실패: ${errorData.message}`);                      }                  } catch (error) {                      console.error('저장 중 오류:', error);                      alert('❌ 네트워크 오류 또는 서버 오류로 저장에 실패했습니다.');                  }              }}            >              제출/저장            </button>            </div>          </footer>        </main>      </div>    );  } |

|  |
| --- |
| *// IntegratedMenu.js*  "use client";  import { useEffect, useMemo, useState } from "react";  import { useRouter } from "next/navigation";  import AccordionSidebar from "./AccordionSidebar";  import "./css/UnvierFuction\_menu.css";  export default function IntegratedMenu() {    const router = useRouter();    const [userType, setUserType] = useState(null); *// "student" | "professor" | "employee" | "teaching\_assistant" | null*    const [loading, setLoading] = useState(true);    const sectionsStudent = useMemo(      () => [        { title: "학적", items: ["개인정보 관리"] } ,        { title: "수 업", items: ["수업 시간표"] },        { title: "수 강", items: ["수강관리", "수강출력"] },        {          title: "성 적",          items: [            { title: "성적처리", items: ["기이수성적조회"] },          ],        },        { title: "출석", items: ["출석조회"] }, *// 학생은 조회 중심*        { title: "휴학", items: ["휴학 신청", "복학 신청"] },      ],      []    );    const sectionsStaff = useMemo(      () => [        { title: "학적", items: ["개인정보 관리"] },        { title: "수 업", items: ["수업 시간표"] },        { title: "수 강", items: ["수강관리", "수강출력"] },        {          title: "성 적",          items: [            { title: "성적입력", items: ["성적입력", "성적출력"] }, *// 교원/직원용*          ],        },        { title: "출석", items: ["출석입력", "출석조회"] },        { title: "휴학", items: ["휴학 신청", "복학 신청"] },      ],      []    );    const routeMapStudent = useMemo(      () => ({        "학적 > 개인정보 관리":"/main\_content/Integrated/student\_info",              "수 업 > 수업 시간표" :"/main\_content/Integrated/timetable",        "휴학 > 휴학 신청": "/main\_content/Integrated/leave\_school",        "휴학 > 복학 신청": "/main\_content/Integrated/back\_school",        "출석 > 출석조회": "/main\_content/Integrated/attendance-check",        "성 적 > 성적처리 > 금학기 성적 조회(미관점)":          "/main\_content/Integrated/grades/current",        "성 적 > 성적처리 > 기이수성적조회":          "/main\_content/Integrated/student\_grades\_export",      }),      []    );    const routeMapStaff = useMemo(      () => ({        "학적 > 학적 관리 > 개인정보 관리":"/main\_content/Integrated/student\_info",        "수 업 > 수업 시간표" :"/main\_content/Integrated/timetable",        "휴학 > 휴학 신청": "/main\_content/Integrated/leave\_school",        "휴학 > 복학 신청": "/main\_content/Integrated/back\_school",        "출석 > 출석입력": "/main\_content/Integrated/attendance-input",        "출석 > 출석조회": "/main\_content/Integrated/attendance-check",        "성 적 > 성적처리 > 기이수성적조회":          "/main\_content/Integrated/grades/history",        "성 적 > 성적입력 > 성적입력": "/main\_content/Integrated/grades-input",        "성 적 > 성적입력 > 성적출력": "/main\_content/Integrated/grade\_export",      }),      []    );    const normalizeUserType = (raw) => {      if (!raw || typeof raw !== "string") return null;      const t = raw.trim().toLowerCase();      if (t === "employ") return "employee";      return t;    };    useEffect(() => {      let alive = true;      (async () => {        try {          setLoading(true);          const res = await fetch("http://localhost:3000/api/univer\_city/user\_route", {            cache: "no-store",            credentials: "same-origin",          });          const data = await res.json().catch(() => ({}));          const normalized = normalizeUserType(data?.user\_type);          if (alive) setUserType(normalized);        } catch {          if (alive) setUserType(null);        } finally {          if (alive) setLoading(false);        }      })();      return () => {        alive = false;      };    }, []);    const selectedSections = useMemo(() => {      if (userType === "student") return sectionsStudent;      if (userType === "employee" || userType === "professor" || userType === "teaching\_assistant")        return sectionsStaff;      return sectionsStudent;    }, [userType, sectionsStudent, sectionsStaff]);    const routeMap = useMemo(() => {      if (userType === "student") return routeMapStudent;      if (userType === "employee" || userType === "professor" || userType === "teaching\_assistant")        return routeMapStaff;      return routeMapStudent;    }, [userType, routeMapStudent, routeMapStaff]);    const handleSelect = (key) => {      const path = routeMap[key];      if (path) {        router.push(path);      } else {        console.warn(`[IntegratedMenu] routeMap에 매핑이 없습니다: ${key}`);      }    };    return (      <div className="route-integrated">        <div className="integrated-container">          <div className="integrated-body">            {loading ? (              <div style={{ padding: 16 }}>메뉴 로딩 중…</div>            ) : (              <AccordionSidebar sections={selectedSections} onSelect={handleSelect} />            )}          </div>        </div>      </div>    );  } |

|  |
| --- |
| */\*AttendanceCheck.css\*/*  body {      font-family: *Arial*, *sans-serif*;      padding: 20px;      background-color: #f4f4f9;  }  header {      margin-bottom: 20px;      display: *flex*;      justify-content: *space-between*;      align-items: *center*;  }  .controls select, .controls button {      padding: 8px;      margin-left: 10px;  }  .calendar-nav {      display: *flex*;      justify-content: *space-between*;      align-items: *center*;      margin-bottom: 15px;  }  .calendar-nav button {      padding: 5px 10px;  }  .calendar {      display: *grid*;      grid-template-columns: repeat(7, 1fr);      width: 100%;      max-width: 900px;      border: 1px *solid* #ccc;      background-color: #fff;  }  .day-name, .day {      padding: 10px;      border: 1px *solid* #eee;      text-align: *center*;      min-height: 80px;  }  .day-name {      background-color: #e0e0e0;      font-weight: *bold*;  }  */\* 출결 상태 색상 \*/*  .present { background-color: #d4edda; color: #155724; } */\* 녹색 \*/*  .late { background-color: #ffeeba; color: #856404; }    */\* 노란색 \*/*  .absent { background-color: #f8d7da; color: #721c24; }  */\* 빨간색 \*/*  .unknown { background-color: #f8f8f8; color: #6c757d; } */\* 회색 \*/*  .no-class { background-color: #f8f8f8; color: #ccc; }  */\* 범례 스타일 \*/*  .legend {      margin-top: 20px;      padding: 10px;      border-top: 1px *solid* #ccc;  }  .status-box {      display: *inline-block*;      padding: 3px 8px;      margin-right: 10px;      border-radius: 4px;      font-size: 0.9em;  }  */\* Modal (팝업) 스타일 \*/*  .modal {      display: *none*;      position: *fixed*;      z-index: 1;      left: 0;      top: 0;      width: 100%;      height: 100%;      overflow: *auto*;      background-color: rgba(0,0,0,0.4);  }  .modal-content {      background-color: #fefefe;      margin: 15% *auto*;      padding: 20px;      border: 1px *solid* #888;      width: 80%;      max-width: 600px;      position: *relative*;  }  .close {      color: #aaa;      float: *right*;      font-size: 28px;      font-weight: *bold*;  }  .close:hover, .close:focus {      color: *black*;      text-decoration: *none*;      cursor: *pointer*;  } |

|  |
| --- |
| /\*AttendanceCheck.html\*/  <!DOCTYPE html>  <html lang="ko">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>출석 조회 및 관리</title>      <link rel="stylesheet" href="style.css">  </head>  <body>      <header>          <h1>📅 출석 조회 및 관리 시스템</h1>          <div class="controls">              <select id="course-select">                  <option value="data-structure">컴퓨터와 2-1 자료구조</option>                  <option value="algorithms">알고리즘</option>              </select>              <select id="user-role">                  <option value="professor">교수</option>                  <option value="student">학생</option>              </select>              <button onclick="renderCalendar()">조회</button>          </div>      </header>      <div class="calendar-container">          <div class="calendar-nav">              <button onclick="prevMonth()"> &lt; </button>              <h2 id="current-month-year">2025년 11월</h2>              <button onclick="nextMonth()"> &gt; </button>          </div>            <div class="calendar" id="calendar">              <div class="day-name">일</div>              <div class="day-name">월</div>              <div class="day-name">화</div>              <div class="day-name">수</div>              <div class="day-name">목</div>              <div class="day-name">금</div>              <div class="day-name">토</div>              </div>      </div>        <div class="legend">          \*\*범례:\*\* <span class="status-box present">출석</span>          <span class="status-box late">지각</span>          <span class="status-box absent">결석</span>          <span class="status-box unknown">미처리/수업없음</span>      </div>      <div id="modal" class="modal">          <div class="modal-content">              <span class="close" onclick="closeModal()">&times;</span>              <h3 id="modal-title">출결 상세/관리</h3>              <div id="modal-body">                  </div>          </div>      </div>      <script src="script.js"></script>  </body>  </html> |

|  |
| --- |
| */\*AttendanceCheck.js \*/*  'use client';  import { useState, useEffect, useCallback } from 'react';  import styles from '../attendance-check/style.module.css';  const toast = (msg) => alert(msg);  const TODAY = new Date();  const transformAttendanceData = (rawData) => {      const transformed = {};      rawData.forEach(item => {          const date = item.date || item.attendance\_date;          const studentId = item.studentId;          if (!transformed[date]) {              transformed[date] = {};          }          transformed[date][studentId] = {              status: item.attendance\_status,              memo: item.memo || '',          };      });      return transformed;  };  const getKoreanStatus = (status) => {      switch (status) {          case 'present':          case '출석':              return '출석';          case 'late':          case '지각':              return '지각';          case 'absent':          case '결석':              return '결석';          case 'leave':          case '조퇴':              return '조퇴';          default:              return '미처리';      }  };  const getStatusClass = (status) => {      switch (status) {          case 'present':          case '출석':              return styles.present;          case 'late':          case '지각':              return styles.late;          case 'absent':          case '결석':              return styles.absent;          case 'leave':          case '조퇴':              return styles.leave;          default:              return styles.noData;      }  };  const fetchAttendanceData = async (monthStr, lectureId) => {      const res = await fetch(          `/api/univer\_city/attendance\_check?courseId=${lectureId}&month=${monthStr}`      );      if (!res.*ok*) return [];      return await res.json();  };  const fetchDailyStudentAttendance = async (date, lectureId) => {      const res = await fetch(          `/api/univer\_city/attendance\_data?lectureId=${lectureId}&date=${date}`      );      if (!res.*ok*) return [];      return await res.json();  };  const fetchLectureList = async () => {      const res = await fetch("/api/univer\_city/lecture\_list");      if (!res.*ok*) return [];      return await res.json();  };  const fetchLectureDetail = async (lectureId) => {      const res = await fetch(`/api/univer\_city/lecture\_detail?lectureId=${lectureId}`);      if (!res.*ok*) return null;      return await res.json();  };  export default function AttendanceCheck({ initialUserRole }) {      const [currentDate, setCurrentDate] = useState(new Date());      const [userRole, setUserRole] = useState(initialUserRole || 'professor');      const [lectureList, setLectureList] = useState([]);      const [selectedLecture, setSelectedLecture] = useState(null);      const [lectureDetail, setLectureDetail] = useState(null);      const [modalData, setModalData] = useState(null);      const [attendanceData, setAttendanceData] = useState({});      const [isLoading, setIsLoading] = useState(true);      const USER\_ID = 2;      useEffect(() => {          const loadLectures = async () => {              const list = await fetchLectureList();              setLectureList(list);              if (list.length > 0) {                  setSelectedLecture(list[0].lecture\_id);              }          };          loadLectures();      }, []);      useEffect(() => {          if (!selectedLecture) return;          const loadDetail = async () => {              const detail = await fetchLectureDetail(selectedLecture);              setLectureDetail(detail);          };          loadDetail();      }, [selectedLecture]);      useEffect(() => {          if (!selectedLecture) return;          const fetchMonth = async () => {              setIsLoading(true);              const y = currentDate.getFullYear();              const m = String(currentDate.getMonth() + 1).padStart(2, '0');              const monthStr = `${y}-${m}`;              const rawData = await fetchAttendanceData(monthStr, selectedLecture);              const transformed = transformAttendanceData(rawData);              setAttendanceData(transformed);              setIsLoading(false);          };          fetchMonth();      }, [currentDate, selectedLecture]);      const handleDayClick = useCallback(          async (date) => {              if (userRole !== 'professor') return;              if (!lectureDetail) return;              const formattedDate = date.toISOString().split('T')[0];              setModalData({                  title: `${formattedDate} 출석 상세`,                  body: <p>로딩중...</p>,              });              const data = await fetchDailyStudentAttendance(                  formattedDate,                  selectedLecture              );              const table = (                  <table className={styles.dataTable}>                      <thead>                          <tr>                              <th>학번</th>                              <th>이름</th>                              <th>출결</th>                              <th>시간</th>                              <th>메모</th>                          </tr>                      </thead>                      <tbody>                          {data.map((st) => (                              <tr key={st.studentId}>                                  <td>{st.student\_number}</td>                                  <td>{st.name}</td>                                  <td>                                      <span                                          className={`${styles.statusPill} ${getStatusClass(                                              st.status                                          )}`}                                      >                                          {getKoreanStatus(st.status)}                                      </span>                                  </td>                                  <td>{st.hours > 0 ? `${st.hours}시간` : '-'}</td>                                  <td>{st.memo || st.late\_reason || '-'}</td>                              </tr>                          ))}                      </tbody>                  </table>              );              setModalData({                  title: `${formattedDate} 출석 상세`,                  body: table,              });          },          [userRole, selectedLecture, lectureDetail]      );      const renderCalendar = useCallback(() => {          if (!lectureDetail) return null;          const year = currentDate.getFullYear();          const month = currentDate.getMonth();          const first = new Date(year, month, 1);          const startIndex = first.getDay();          const daysInMonth = new Date(year, month + 1, 0).getDate();          const grid = [];          const korMap = {              일: 0,              월: 1,              화: 2,              수: 3,              목: 4,              금: 5,              토: 6,          };          const lectureDayIndex = korMap[lectureDetail.day\_of\_week];          for (let i = 0; i < startIndex; i++) {              grid.push({ current: false });          }          for (let day = 1; day <= daysInMonth; day++) {              const dateObj = new Date(year, month, day);              const dateStr = dateObj.toISOString().split('T')[0];              const isLectureDay = dateObj.getDay() === lectureDayIndex;              const daily = attendanceData[dateStr]?.[USER\_ID];              const status = daily                  ? daily.status                  : isLectureDay                  ? '미처리'                  : 'noData';              grid.push({                  current: true,                  date: dateObj,                  isLectureDay,                  status,              });          }          return grid.map((d, idx) => {              if (!d.current) {                  return (                      <div                          key={idx}                          className={`${styles.day} ${styles.otherMonth}`}                      ></div>                  );              }              let className = `${styles.day}`;              if (d.isLectureDay) {                  className += ` ${styles.lectureDay}`;                  className += ` ${styles.lectureDayBorder}`;              }              return (                  <div                      key={idx}                      className={className}                      onClick={() =>                          d.isLectureDay && handleDayClick(d.date)                      }                  >                      <span className={styles.dateNumber}>                          {d.date.getDate()}                      </span>                      {d.isLectureDay && (                          <span                              className={`${styles.statusPill} ${getStatusClass(                                  d.status                              )}`}                          >                              {getKoreanStatus(d.status)}                          </span>                      )}                  </div>              );          });      }, [attendanceData, currentDate, selectedLecture, lectureDetail]);      return (          <div className={styles.layout}>              <div className={styles.topbar}>                  <div className={styles.titleRow}>                      <h1 className={styles.title}>출석 조회</h1>                      <select                          className={styles.classSelect}                          value={selectedLecture || ''}                          onChange={(e) =>                              setSelectedLecture(Number(e.target.value))                          }                      >                          {lectureList.map((lec) => (                              <option                                  key={lec.lecture\_id}                                  value={lec.lecture\_id}                              >                                  {lec.lecture\_name} ({lec.day\_of\_week})                              </option>                          ))}                      </select>                  </div>              </div>              <div className={styles.calendarContainer}>                  <div className={styles.calendarNav}>                      <button                          onClick={() =>                              setCurrentDate(                                  (prev) =>                                      new Date(                                          prev.getFullYear(),                                          prev.getMonth() - 1,                                          1                                      )                              )                          }                      >                          &lt; 이전달                      </button>                      <h2>{`${currentDate.getFullYear()}년 ${                          currentDate.getMonth() + 1                      }월`}</h2>                      <button                          onClick={() =>                              setCurrentDate(                                  (prev) =>                                      new Date(                                          prev.getFullYear(),                                          prev.getMonth() + 1,                                          1                                      )                              )                          }                      >                          다음달 &gt;                      </button>                  </div>                  <div className={styles.calendar}>                      <div className={styles.dayName}>일</div>                      <div className={styles.dayName}>월</div>                      <div className={styles.dayName}>화</div>                      <div className={styles.dayName}>수</div>                      <div className={styles.dayName}>목</div>                      <div className={styles.dayName}>금</div>                      <div className={styles.dayName}>토</div>                      {isLoading ? (                          <div                              style={{                                  gridColumn: '1 / span 7',                                  textAlign: 'center',                                  padding: '20px',                              }}                          >                              로딩중...                          </div>                      ) : (                          renderCalendar()                      )}                  </div>              </div>              {modalData && (                  <div className={styles.modalBackdrop}>                      <div className={styles.modalContent}>                          <span                              className={styles.close}                              onClick={() => setModalData(null)}                          >                              &times;                          </span>                          <h3>{modalData.title}</h3>                          <div className={styles.modalBody}>                              {modalData.body}                          </div>                      </div>                  </div>              )}          </div>      );  } |

|  |
| --- |
| 'use client'; *// AttendanceCheck.js가 'use client' 이므로 페이지 파일에도 추가합니다.*  import IntegratedMenu from '../../../../app/main\_content/components/IntegratedMenu';  import AttendanceCheck from '../attendance-check/AttendanceCheck';  export default function AttendanceCheckPage() {     return (        <div style={{ display: "flex", alignItems: "flex-start" }}>        <IntegratedMenu/>          <div style={{ flex: 1 }}>        <AttendanceCheck initialUserRole="professor" />      </div>    </div>      );  } |

|  |
| --- |
| */\* style.module.css \*/*  .layout {      padding: 20px;      background-color: #f4f4f9;  }  .topbar {      margin-bottom: 20px;      padding-bottom: 10px;      border-bottom: 1px *solid* #ccc;  }  .crumbs {      font-size: 0.9em;      color: #666;  }  .title {      font-size: 1.8em;      margin: 5px 0 15px;  }  .controls {      display: *flex*;      gap: 10px;  }  .classSelect, .roleSelect {      padding: 8px;      border-radius: 4px;  }  .classDay {      background-color: #f0f7ff;      border-color: #a8d9ff;      font-weight: 600;  }  .noData {      background-color: #f8f8f8;      color: #6c757d;  }  .day {      padding: 10px;      border: 1px *solid* #eee;      text-align: *center*;      min-height: 80px;  }  .calendarContainer {      max-width: 900px;      margin: 0 *auto*;      background-color: #fff;      padding: 20px;      border-radius: 8px;      box-shadow: 0 2px 5px rgba(0,0,0,0.1);  }  .calendarNav {      display: *flex*;      justify-content: *space-between*;      align-items: *center*;      margin-bottom: 15px;      padding: 0 5px;  }  .calendar {      display: *grid*;      grid-template-columns: repeat(7, 1fr);      width: 100%;      border: 1px *solid* var(--line, #ccc);      border-radius: 4px;      overflow: *hidden*;  }  .dayName {      padding: 10px;      border: 1px *solid* var(--line, #eee);      text-align: *center*;      background-color: #e0e0e0;      font-weight: *bold*;      font-size: 0.9em;  }  .day {      padding: 5px;      border: 1px *solid* var(--line, #eee);      text-align: *left*;      min-height: 90px;      position: *relative*;      cursor: *default*;      transition: background-color 0.1s;  }  .day:hover {      box-shadow: *inset* 0 0 0 1px var(--accent, #3b82f6);  }  .otherMonth {      background-color: var(--bg, #f4f4f9);      color: var(--sub, #aaa);  }  .today {      border: 2px *solid* var(--accent, #3b82f6) !important;      background-color: var(--accent-light, #eaf2ff);  }  .dateNumber {      display: *block*;      font-size: 1.1em;      font-weight: 600;      margin-bottom: 5px;  }  .pillsContainer {      display: *flex*;      flex-direction: *column*;      gap: 2px;      margin-top: 5px;      line-height: 1;  }  .statusPill {      display: *inline-block*;      padding: 2px 5px;      margin-right: 3px;      border-radius: 9999px;      font-size: 0.7em;      font-weight: 500;      white-space: *nowrap*;      text-align: *center*;  }  .present { background-color: var(--color-green-light, #d4edda); color: var(--color-green-dark, #155724); }  .late { background-color: var(--color-yellow-light, #ffeeba); color: var(--color-yellow-dark, #856404); }  .leave { background-color: var(--color-blue-light, #cfe2ff); color: var(--color-blue-dark, #084298); }  .absent { background-color: var(--color-red-light, #f8d7da); color: var(--color-red-dark, #721c24); }  .noData { background-color: var(--color-gray-light, #f8f9fa); color: var(--color-gray-dark, #6c757d); border: 1px *solid* #dee2e6; }  .legend {      margin-top: 20px;      padding: 10px;      border-top: 1px *solid* #ccc;      display: *flex*;      gap: 15px;      align-items: *center*;      font-size: 0.9em;  }  .statusBox {      display: *inline-block*;      padding: 5px 10px;      border-radius: 4px;      font-size: 0.9em;      font-weight: 600;  }  .calendarNav button {      padding: 8px 15px;      font-size: 1em;      cursor: *pointer*;      border: 1px *solid* #ddd;      background: #f0f0f0;      border-radius: 4px;  }  .calendarNav h2 {      font-size: 1.5em;      margin: 0;  }  .calendarGrid {      display: *grid*;      grid-template-columns: repeat(7, 1fr);      border-top: 1px *solid* #eee;      border-left: 1px *solid* #eee;  }  .dayName {      background-color: #e0e0e0;      font-weight: *bold*;      padding: 10px;      text-align: *center*;      border-right: 1px *solid* #eee;      border-bottom: 1px *solid* #eee;  }  .day {      padding: 10px;      min-height: 80px;      border-right: 1px *solid* #eee;      border-bottom: 1px *solid* #eee;      text-align: *right*;      cursor: *pointer*;      position: *relative*;      transition: background-color 0.2s;  }  .day:hover {      background-color: #fafafa;  }  .day strong {      display: *block*;      font-size: 1.2em;      margin-bottom: 5px;  }  */\* 출결 상태 색상 \*/*  .present { background-color: #d4edda; color: #155724; } */\* 녹색 \*/*  .late { background-color: #ffeeba; color: #856404; }    */\* 노란색 \*/*  .absent { background-color: #f8d7da; color: #721c24; }  */\* 빨간색 \*/*  .unknown { background-color: #f8f8f8; color: #6c757d; } */\* 회색 \*/*  .noClass { background-color: #f8f8f8; color: #ccc; cursor: *default*; }  .summary, .myStatus {      display: *block*;      font-size: 0.85em;      text-align: *left*;      margin-top: 5px;      padding: 2px 0;      font-weight: *normal*;  }  */\* 범례 \*/*  .legend {      margin-top: 20px;      padding: 10px;      text-align: *center*;  }  .statusBox {      display: *inline-block*;      padding: 5px 10px;      margin-left: 10px;      border-radius: 4px;      font-size: 0.9em;  }  */\* Modal \*/*  .modalBackdrop {      position: *fixed*;      z-index: 1000;      left: 0;      top: 0;      width: 100%;      height: 100%;      background-color: rgba(0,0,0,0.5);      display: *flex*;      justify-content: *center*;      align-items: *center*;  }  .modalContent {      background-color: #fff;      padding: 25px;      border-radius: 8px;      width: 90%;      max-width: 900px;      position: *relative*;      box-shadow: 0 5px 15px rgba(0,0,0,0.3);      max-height: 90vh;      overflow-y: *auto*;  }  .close {      color: #aaa;      float: *right*;      font-size: 28px;      font-weight: *bold*;      cursor: *pointer*;  }  .modalBody {      padding-top: 15px;  }  .modalFooter {      margin-top: 20px;      display: *flex*;      justify-content: *flex-end*;      gap: 10px;  }  .dataTable {      width: 100%;      border-collapse: *collapse*;      margin-top: 15px;      font-size: 0.9em;      table-layout: *fixed*;  }  .dataTable th, .dataTable td {      border: 1px *solid* #ddd;      padding: 8px;      text-align: *left*;  }  .dataTable th, .dataTable td {      border: 1px *solid* var(--line, #ddd);      padding: 10px;      text-align: *center*;  }  .dataTable th {      background-color: var(--bg, #f4f4f9);      font-weight: 600;  }  .btn {      padding: 10px 15px;      border: 1px *solid* *transparent*;      border-radius: 8px;      font-weight: 600;      cursor: *pointer*;      transition: background-color 0.2s;  }  .btn.primary {      background-color: #4c51bf;      color: #fff;  }  .btn.primary:hover {      background-color: #3f45a0;  }  .primaryBtn {      background-color: #007bff;      color: *white*;      border: *none*;      padding: 10px 15px;      border-radius: 4px;      cursor: *pointer*;      margin-top: 15px;  }  .ghostBtn {      background-color: *transparent*;      color: #007bff;      border: 1px *solid* #007bff;      padding: 10px 15px;      border-radius: 4px;      cursor: *pointer*;      margin-top: 15px;  }  .formInput {      width: 100%;      padding: 8px;      border: 1px *solid* #ccc;      border-radius: 4px;      box-sizing: *border-box*;      margin-top: 5px;  }  .lectureDay {      background-color: #e4f2ff;      border-radius: 6px;  }  .lectureDayBorder {      border: 2px *solid* #7bb9ff;  } |

|  |
| --- |
| <!doctype html>  <html lang="ko">  <head>  <meta charset="utf-8" />  <meta name="viewport" content="width=device-width, initial-scale=1" />  <title>출석 카드(선택적 box-sizing)</title>  <style>    :root {      --bg:#f6f7fb;      --card:#fff;      --ink:#0f172a;      --sub:#64748b;      --line:#e5e7eb;      --radius:16px;    }    body {      margin:0;      padding:24px;      background:var(--bg);      color:var(--ink);      font-family: *ui-sans-serif*, *system-ui*, *-apple-system*, "Noto Sans KR", *sans-serif*;    }    .cards, .card, .photo, .photo .placeholder, .right, .panel,    .info-grid, .chips, .chip, .notes textarea, .footer, .btn {      box-sizing: *border-box*;    }    .cards {      display: *grid*;      grid-auto-rows: *auto*;      gap: 18px;      justify-content: *start*;    }    .card {      display: *grid*;      grid-auto-flow: *column*;      grid-template-columns: *auto* *auto*;      gap: 14px;      background: var(--card);      border: 1px *solid* var(--line);      border-radius: var(--radius);      box-shadow: 0 6px 16px rgba(0,0,0,.05);      padding: 14px;      width: *fit-content*;    }    .photo {      width: 160px;      aspect-ratio: 3 / 4;      border: 2px *dashed* #cbd5e1;      border-radius: var(--radius);      background: #f1f5f9 *center*/*cover* *no-repeat*;      display: *grid*;      place-items: *center*;    }    .photo .placeholder {      color: var(--sub);      font-size: 14px;      text-align: *center*;      display: *grid*;      gap: 6px;    }    .photo .placeholder svg {      width: 32px;      height: 32px;      opacity:.55;    }    .right {      display: *grid*;      gap: 12px;      width: *fit-content*;    }    .panel {      border: 1px *solid* var(--line);      border-radius: var(--radius);      padding: 12px;      background: #fff;    }    .panel h2 {      margin: 0 0 8px;      font-size: 15px;    }    .info-grid {      display: *grid*;      grid-template-columns: 70px *auto*;      gap: 6px 10px;    }    .label { color: var(--sub); font-size: 13px }    .value { font-weight: 600; font-size: 14px }    .chips { display:*flex*; gap:8px; flex-wrap:*wrap*; margin:2px 0 8px }    .chip {      padding:6px 10px;      border-radius:999px;      border:1px *solid* var(--line);      background:#fff;      font-weight:600;      font-size:13px;      cursor:*pointer*;    }    .notes label { display:*block*; font-size:12px; color:var(--sub); margin-bottom:4px }    .notes textarea {      width:100%;      min-height:70px;      resize:*vertical*;      border:1px *solid* var(--line);      border-radius:10px;      padding:8px;      font:*inherit*;      font-size:13px;    }    .footer {      display:*flex*;      justify-content:*space-between*;      align-items:*center*;      margin-top:6px;    }    .stamp { font-size:12px; color:var(--sub) }    .btn {      background:var(--ink);      color:#fff;      border:1px *solid* var(--ink);      border-radius:8px;      padding:6px 12px;      font-weight:700;      font-size:13px;      cursor:*pointer*;    }  </style>  </head>  <body>  <div class="cards">    <article class="card">      <div class="photo" aria-label="학생 사진">        <div class="placeholder">          <svg viewBox="0 0 24 24" fill="currentColor">            <path d="M9 3a1 1 0 0 0-.894.553L7.382 5H5a3 3 0 0 0-3 3v9a3 3 0 0 0 3 3h14a3 3 0 0 0 3-3V8a3 3 0 0 0-3-3h-2.382l-.724-1.447A1 1 0 0 0 14 3H9zm3 5a5 5 0 1 1 0 10 5 5 0 0 1 0-10zm0 2a3 3 0 1 0 0 6 3 3 0 0 0 0-6z"/>          </svg>          <strong>사진 없음</strong>        </div>      </div>      <div class="right">        <section class="panel info">          <h2>학생 정보</h2>          <div class="info-grid">            <div class="label">이름</div><div class="value">홍길동</div>            <div class="label">학번</div><div class="value">20241234</div>          </div>        </section>        <section class="panel">          <h2>출석 처리</h2>          <div class="chips">            <button class="chip" type="button">출석</button>            <button class="chip" type="button">지각</button>            <button class="chip" type="button">결석</button>          </div>          <div class="notes">            <label for="memo">메모(선택)</label>            <textarea id="memo" placeholder="특이사항을 기록하세요."></textarea>          </div>          <div class="footer">            <span class="stamp">처리 전</span>            <button class="btn" type="button">저장</button>          </div>        </section>      </div>    </article>  </div>  </body>  </html> |

|  |
| --- |
| :root { --bg:#f6f7fb; --ink:#111827; --muted:#6b7280; --line:#e5e7eb; --accent:#2563eb; }  \* { box-sizing: *border-box*; }  body { margin:0; font-family: *system-ui*,*-apple-system*,Segoe UI,Roboto,Apple SD Gothic Neo,Malgun Gothic,*Helvetica*,*Arial*,*sans-serif*; color:var(--ink); background:var(--bg); }  .layout {    display:*grid*;    grid-template-columns: 1fr;    min-height:100dvh;    max-width: 100%;    margin: 0 *auto*;  }  .main{ display:*grid*; grid-template-rows:*auto* 1fr *auto*; min-height:100dvh; }  .topbar{    display:*grid*; align-items:*center*; justify-content:*space-between*;    padding:30px clamp(16px,3vw,28px); background:#fff; border-bottom:1px *solid* var(--line); width:100%;  }  .crumbs{ font-size:12px; color:var(--muted); }  .titleRow{ display:*flex*; align-items:*center*; gap:10px; margin-top:4px; }  .title{ margin:0; font-size:clamp(20px,2.4vw,28px); }  .classSelect{ border:1px *solid* #d1d5db; border-radius:10px; padding:8px 10px; background:#fff; }  .todayDate{ font-size:14px; font-weight:600; color:var(--ink); }  .grid{ display:*grid*; grid-template-columns:280px 1fr 320px; gap:16px; padding:16px clamp(16px,3vw,28px); }  .card{ background:#fff; border:1px *solid* var(--line); border-radius:16px; padding:14px; display:*grid*; gap:12px; }  .card h3{ margin:0; font-size:16px; }  .btn{ border:1px *solid* #d1d5db; background:#fff; color:var(--ink); padding:10px 12px; border-radius:10px; font-weight:600; cursor:*pointer*; }  .btn:hover{ background:#f3f4f6; }  .btn.primary{ background:var(--accent); border-color:var(--accent); color:#fff; }  .btn.ghost{ background:#fff; }  .field{ display:*grid*; gap:6px; }  .field span{ font-size:13px; color:#374151; }  .field input, .field select, textarea{ border:1px *solid* #d1d5db; border-radius:10px; padding:10px 12px; background:#fff; font-size:14px; }  .list{ margin:0; padding-left:18px; font-size:13px; }  .desc{ display:*grid*; gap:6px; }  .desc div{ display:*grid*; grid-template-columns:80px 1fr; gap:8px; }  .desc dt{ color:#6b7280; }  .desc dd{ margin:0; }  .legend{ display:*flex*; flex-wrap:*wrap*; gap:6px; }  .pill{ padding:4px 8px; border-radius:999px; font-size:12px; border:1px *solid* var(--line); background:#fff; }  .pill.ok{ border-color:#10b981; color:#064e3b; }  .pill.late{ border-color:#f59e0b; color:#7c2d12; }  .pill.leave{ border-color:#06b6d4; color:#0e7490; }  .pill.absent{ border-color:#ef4444; color:#7f1d1d; }  .centerPlaceholder{ display:*grid*; }  .placeholderBox{ border:2px *dashed* var(--line); border-radius:16px; min-height:420px; display:*grid*; place-items:*center*; color:#6b7280; background:#fafafa; }  .weeks{ display:*grid*; gap:8px; max-height:320px; overflow:*auto*; padding-right:2px; }  .weekItem{ display:*flex*; align-items:*center*; justify-content:*space-between*; gap:10px; padding:8px 10px; border:1px *solid* var(--line); border-radius:10px; background:#fff; cursor:*pointer*; }  .weekItem:hover{ background:#f9fafb; }  .weekItem.active{ outline:2px *solid* var(--accent); border-color:var(--accent); }  .weekItem .date{ font-size:12px; color:#6b7280; }  .bottomBar{ position:*sticky*; bottom:0; display:*flex*; align-items:*center*; justify-content:*space-between*; padding:12px clamp(16px,3vw,28px); background:rgba(255,255,255,.9); border-top:1px *solid* var(--line); backdrop-filter:blur(6px); }  .leftRow,.rightRow{ display:*flex*; gap:8px; align-items:*center*; }  .muted{ color:#6b7280; font-size:13px; }  @media (max-width:1200px){ .grid{ grid-template-columns:240px 1fr 300px; } }  @media (max-width:980px){    .layout{ grid-template-columns: 1fr; }    .grid{ grid-template-columns:1fr; }    .centerPlaceholder{ order:2; }    .toolCol,.infoCol{ order:1; }  } |

|  |
| --- |
| <!doctype html>  <html lang="ko">  <head>    <meta charset="utf-8" />    <meta name="viewport" content="width=device-width,initial-scale=1" />    <title>출석 입력 | 동서울 학사</title>  *<!-- CSS 분리 -->*    <link rel="stylesheet" href="attendance-input.css" />  </head>  <body>    <div class="layout">      <main class="main">  *<!-- 상단 헤더 -->*        <header class="topbar">          <div class="left">            <div class="crumbs">출석관리 &gt; <strong>출석 입력</strong></div>            <div class="titleRow">              <h1 class="title">출석 입력</h1>              <select class="classSelect" id="classSelect">                <option>컴소과 2-1 자료구조</option>                <option>컴소과 1-2 웹프로그래밍</option>                <option>컴소과 3-1 운영체제</option>              </select>            </div>          </div>          <div class="right">            <div class="todayDate" id="todayDate"></div>          </div>        </header>        <section class="grid">          <aside class="toolCol">            <div class="card">              <h3>필터</h3>              <label class="field">                <span>상태</span>                <select>                  <option>전체</option>                  <option>출석</option>                  <option>지각</option>                  <option>조퇴</option>                  <option>결석</option>                </select>              </label>              <label class="field">                <span>이름/학번 검색</span>                <input type="text" placeholder="예: 홍길동 / 20231234  " />              </label>            </div>            <div class="card">              <h3>메모</h3>              <textarea id="memo" rows="6" placeholder="수업 특이사항, 과제 공지 등"></textarea>              <div class="rightRow">                <button class="btn ghost" type="button" id="saveMemo">임시저장</button>              </div>            </div>            <div class="card">              <h3>주차 선택</h3>              <div class="weeks" id="weeks"></div>            </div>            <div class="card">              <h3>오늘 수업정보</h3>              <dl class="desc">                <div><dt>강의명</dt><dd id="className">컴소과                   2-1 자료구조</dd></div>                <div><dt>담당</dt><dd>김교수</dd></div>                <div><dt>정원/수강</dt><dd> 36 / 40 </dd></div>                <div><dt>평균 출석률</dt><dd>92%</dd></div>              </dl>              <div class="legend">                <span class="pill ok">출석</span>                <span class="pill late">지각</span>                <span class="pill leave">조퇴</span>                <span class="pill absent">결석</span>              </div>            </div>            <div class="card">              <h3>최근 공지</h3>              <ul class="list">                <li>9/10: 팀프로젝트 조 편성 완료</li>                <li>9/12: 실습실 변경 B-205</li>                <li>9/15: 퀴즈 #2 공지</li>              </ul>            </div>          </aside>          <div class="centerPlaceholder">            <div class="placeholderBox">중앙 출석표 컴포넌트 영역</div>          </div>        </section>        <footer class="bottomBar">          <div class="leftRow"><span class="muted">자동저장 간격 30초</span></div>          <div class="rightRow">            <button class="btn ghost" type="button" id="revertBtn">원복</button>            <button class="btn" type="button" id="tempSaveBtn">임시저장</button>            <button class="btn primary" type="button" id="submitBtn">제출/저장</button>          </div>        </footer>      </main>    </div>    <script>      const todayDateEl = document.getElementById("todayDate");      const today = new Date();      const y = today.getFullYear();      const m = String(today.getMonth()+1).padStart(2,"0");      const d = String(today.getDate()).padStart(2,"0");      todayDateEl.textContent = `오늘 ${y}-${m}-${d}`;      const classSelect = document.getElementById('classSelect');      const className = document.getElementById('className');      classSelect.addEventListener('change',()=>{ className.textContent = classSelect.value; });      const toast = (msg)=>alert(msg);      document.getElementById('saveMemo').addEventListener('click',()=>toast('메모가 임시저장되었습니다.'));      document.getElementById('revertBtn').addEventListener('click',()=>toast('마지막 저장 상태로 원복했습니다.'));      document.getElementById('tempSaveBtn').addEventListener('click',()=>toast('임시저장 완료'));      document.getElementById('submitBtn').addEventListener('click',()=>toast('제출/저장되었습니다.'));      const WEEK\_START = "2025-03-01";      const WEEK\_COUNT = 16;      function addDays(date, days){        const d = new Date(date);        d.setDate(d.getDate() + days);        return `${d.getFullYear()}-${String(d.getMonth()+1).padStart(2,"0")}-${String(d.getDate()).padStart(2,"0")}`;      }      const weeksEl = document.getElementById('weeks');      for(let i=1;i<=WEEK\_COUNT;i++){        const dateStr = addDays(WEEK\_START, (i-1)\*7);        const btn = document.createElement('div');        btn.className = 'weekItem';        btn.innerHTML = `<span>${i}주차</span><span class="date">(${dateStr})</span>`;        weeksEl.appendChild(btn);      }    </script>  </body>  </html> |

|  |
| --- |
| :root {    --bg:#f6f7fb;    --card:#fff;    --ink:#0f172a;    --sub:#64748b;    --line:#e5e7eb;    --radius:16px;  }  .cards, .card, .photo, .photo .placeholder, .right, .panel,  .info-grid, .chips, .chip, .notes textarea, .footer, .btn {    box-sizing: *border-box*;  }  .cards {    gap: 16px;  }  .cards .card {      max-width: 400px;      width: 100%;  }  .card {      display: *flex* ;      flex-direction: *column* ;      align-items: *center* ;      gap: 5px ;      padding: 20px ;      max-width: 300px ;      min-width: 200px ;      width: 100% ;    }  .photo {  width: 120px;      aspect-ratio: 1 / 1;      border: 2px *dashed* var(--line);      border-radius: 50%;      background: #f1f5f9 *center*/*cover* *no-repeat*;      display: *grid*;      place-items: *center*;      margin: 0 *auto*;    }    .photo .placeholder {      border-radius: 0%;  }    .footer {      display: *flex*;      justify-content: *flex-end*;      align-items: *center*;      margin-top: 10px;      width: *fit-content*;  .chips .chip.active {      font-weight: *bold* !important;      color: *white* !important; */\* 글자색을 흰색으로 변경 \*/*      border: 1px *solid* *transparent* !important; */\* 경계선 제거 \*/*  }  .chips .chip.active.출석 {      background-color: #10b981 !important; */\* 녹색 \*/*  }  .chips .chip.active.지각 {      background-color: #f59e0b !important; */\* 주황색 \*/*  }  .chips .chip.active.결석 {      background-color: #ef4444 !important; */\* 빨간색 \*/*  }  } |

|  |
| --- |
| *// AttendanceCard.js*  'use client';  import { useState } from 'react';  import './AttendanceCard.css';  const toast = (msg) => alert(msg);  export default function AttendanceCard({ student, onStatusChange }) {    const studentInfo = student || {      name: '데이터 로드 오류',      id: 'N/A',    };    const [attendanceStatus, setAttendanceStatus] = useState('미처리');    const [memo, setMemo] = useState(studentInfo.memo);    const handleStatusChange = (status) => {      setAttendanceStatus(status);  *// 부모 컴포넌트에 변경된 상태 전달*      if (onStatusChange) {        onStatusChange(studentInfo.id, status, memo);      }    };    const handleSave = async () => {      if (!studentInfo.id || studentInfo.id === 'N/A') {          return toast("학생 정보가 유효하지 않아 저장할 수 없습니다.");      }      const LECTURE\_ID\_MOCK = 1;      const ATTENDANCE\_DATE\_MOCK = new Date().toISOString().split('T')[0]; *// 오늘 날짜 YYYY-MM-DD*        const postData = {          lectureId: LECTURE\_ID\_MOCK,          attendanceDate: ATTENDANCE\_DATE\_MOCK,          attendanceData: [              {                  studentId: studentInfo.id,                  status: attendanceStatus,                  memo: memo,                  lateReason: null,              }          ]      };        const API\_ENDPOINT = '/api/univer\_city/insert\_attendance';        try {          const response = await fetch(API\_ENDPOINT, {              method: 'POST',              headers: {                  'Content-Type': 'application/json',              },              body: JSON.stringify(postData),          });            const data = await response.json();            if (response.*ok* && data.success) {              toast(`[저장 성공] ${studentInfo.name} (${attendanceStatus}) 상태가 DB에 저장되었습니다.`);          } else {              toast(`[저장 실패] 오류: ${data.message || '알 수 없는 서버 오류'}. 상세: ${data.detail || ''}`);          }      } catch (error) {          console.error('API 호출 중 오류 발생:', error);          toast('네트워크 오류 또는 서버 접속 실패.');      }  };    const handleEdit = () => {      toast(`[수정] ${studentInfo.name} 정보 수정 모드 활성화`);    };    return (      <div className="cards">        <article className="card" style={{            display: 'flex',            flexDirection: 'column',            alignItems: 'center',            gap: '15px',            padding: '20px',            maxWidth: '300px'        }}>            <div className="photo" aria-label="학생 사진" style={{              width: '120px',              aspectRatio: '1/1',              borderRadius: '0%',              margin: '0 0 10px 0'          }}>              <div className="placeholder" style={{ borderRadius: '0%', width: '100%', height: '20%' }}>                <strong>사진 없음</strong>              </div>          </div>          <section className="panel attendance" style={{ textAlign: 'center', width: '100%', padding: '0 10px' }}>              <div className="info-grid" style={{                  display: 'grid',                  gridTemplateColumns: '1fr 1fr',                  textAlign: 'center',                  width: 'fit-content',                  margin: '0 auto'              }}>                <div className="label" style={{ fontWeight: 'bold' }}>이름</div><div className="value">{studentInfo.name}</div>                <div className="label" style={{ fontWeight: 'bold' }}>학번</div><div className="value">{studentInfo.id}</div>              </div>          </section>          <hr style={{ width: '90%', border: 'none', borderTop: '1px solid var(--line, #e5e7eb)' }} />          <section className="panel attendance" style={{ width: '100%', textAlign: 'center', padding: '0 10px' }}>              <div style={{ display: 'flex', justifyContent: 'center', gap: '20px', marginTop: '10px' }}>                  {['출석', '지각', '결석'].map(status => (                      <label key={status} style={{ display: 'flex', alignItems: 'center', cursor: 'pointer' }}>                          <input                              type="radio"                              name={`attendance-${studentInfo.id}`}                              value={status}                              checked={attendanceStatus === status}                              onChange={() => handleStatusChange(status)}                              style={{ marginRight: '5px' }}                          />                          {status}                          {status === '지각' && (                              <input                                  type="text"                                  placeholder=" "                                  style={{                                      marginLeft: '10px',                                      padding: '4px 6px',                                      width: '10px',                                      border: '1px solid #ccc',                                      borderRadius: '4px',                                      opacity: attendanceStatus === '지각' ? 1 : 0.6,                                      pointerEvents: attendanceStatus === '지각' ? 'auto' : 'none',                                  }}                              />                          )}                      </label>                  ))}              </div>          </section>            <hr style={{ width: '90%', border: 'none', borderTop: '1px solid var(--line, #e5e7eb)' }} />          <div className="notes" style={{ width: '100%', padding: '0 10px' }}>              <label htmlFor={`memo-${studentInfo.id}`} style={{ display: 'block', marginBottom: '5px', textAlign: 'center' }}>                  메모(선택)              </label>              <textarea                  id={`memo-${studentInfo.id}`}                  placeholder="특이사항을 기록하세요."                  value={memo}                  onChange={(e) => setMemo(e.target.value)}                  rows="3"                  style={{                      width: '100%',                      padding: '8px',                      border: '1px solid var(--line, #ccc)',                      borderRadius: '4px',                      boxSizing: 'border-box'                  }}              ></textarea>          </div>          <div className="footer" style={{              width: '100%',              display: 'flex',              justifyContent: 'flex-end',              gap: '10px',              marginTop: '10px',              padding: '0 10px',              backgroundColor: 'transparent'          }}>              <button className="btn ghost" type="button" onClick={handleEdit}>                  수정              </button>              <button className="btn primary" type="button" onClick={handleSave}>                  저장              </button>          </div>        </article>      </div>    );  } |

|  |
| --- |
| *// app/attendance-input/page.js*  'use client';  import IntegratedMenu from '../../components/IntegratedMenu'; *// ✅ 꼭 임포트*  import AttendanceContent from '../../components/AttendanceContent';  import "./style.module.css"  export default function AttendanceInputPage() {    return (      <div style={{ display: "flex", alignItems: "flex-start" }}>      <IntegratedMenu/>        <div style={{ flex: 1 }}>      <AttendanceContent />    </div>  </div>    );  } |

|  |
| --- |
| */\* app/attendance-input/style.module.css \*/*  .layout {    min-height: calc(100vh - [헤더높이]);    display: *flex*;    flex-direction: *column*;    width: 100%;  }  .main {    display: *grid*;    grid-template-rows: *auto* 1fr *auto*;    min-height: 100dvh;  }  .topbar {    display: *flex*;    align-items: *center*;    justify-content: *space-between*;    padding: 18px clamp(16px, 3vw, 28px);    background: #fff;    border-bottom: 1px *solid* var(--line);  }  .crumbs {    font-size: 12px;    color: var(--muted);  }  .titleRow {    display: *flex*;    align-items: *center*;    gap: 10px;    margin-top: 4px;  }  .title {    margin: 0;    font-size: clamp(20px, 2.4vw, 28px);  }  .classSelect {    border: 1px *solid* #d1d5db;    border-radius: 10px;    padding: 8px 10px;    background: #fff;  }  .todayDate {    font-size: 14px;    font-weight: 600;    color: var(--ink);  }  */\* 본문 \*/*  .grid {    display: *grid*;    grid-template-columns: 300px 1fr 300px;    gap: 24px;    padding: 16px;    align-items: *start*;  }  .toolCol,  .infoCol {    height: *auto*;    align-self: *start*;    display: *flex*;    flex-direction: *column*;    gap: 12px;  }  .card {    background: #fff;    border: 3px *solid* var(--line);    border-radius: 16px;    padding: 14px;    display: *grid*;    gap: 12px;  }  .cards {    display: *flex*;    flex-direction: *row*;    flex-wrap: *wrap*;    gap: 16px;    justify-content: *center*;    width: 100%;  }  .card h3 {    margin: 0;    font-size: 20px;  }  .btn {    border: 1px *solid* #d1d5db;    background: #fff;    color: var(--ink);    padding: 10px 12px;    border-radius: 10px;    font-weight: 600;    cursor: *pointer*;  }  .btn:hover {    background: #f3f4f6;  }  .btn.primary {    background: var(--accent);    border-color: var(--accent);    color: #fff;  }  .btn.ghost {    background: #fff;  }  .field {    display: *grid*;    gap: 6px;  }  .field span {    font-size: 15px;    color: #374151;  }  .field input,  .field select,  .list {    list-style: *none*;    padding: 0;    margin: 0;  }  .list li {    border-bottom: 1px *solid* var(--line, #e5e7eb);    padding: 10px 0;    font-size: 15px;    color: var(--ink);  }  .list li:last-child {    border-bottom: *none*;  }  .desc {    display: *grid*;    gap: 6px;  }  .desc div {    display: *grid*;    grid-template-columns: 80px 1fr;    gap: 8px;  }  .desc dt {    color: #6b7280;  }  .desc dd {    margin: 0;  }  .legend {    display: *flex*;    flex-wrap: *wrap*;    gap: 6px;  }  .pill {    padding: 4px 8px;    border-radius: 999px;    font-size: 12px;    border: 1px *solid* var(--line);    background: #fff;  }  .pill.ok {    border-color: #10b981;    color: #064e3b;  }  .pill.late {    border-color: #f59e0b;    color: #7c2d12;  }  .pill.leave {    border-color: #06b6d4;    color: #0e7490;  }  .pill.absent {    border-color: #ef4444;    color: #7f1d1d;  }  .centerPlaceholder {    display: *grid*;  }  .centerArea {    grid-column: 2 / 3;    overflow-y: *auto*;  }  .placeholderBox {    border: 2px *dashed* var(--line);    border-radius: 16px;    min-height: 420px;    display: *grid*;    place-items: *center*;    color: #6b7280;    background: #fafafa;  }  .weeks {    display: *grid*;    gap: 8px;    max-height: 320px;    overflow: *auto*;    padding-right: 2px;  }  .weekItem {    display: *flex*;    align-items: *center*;    justify-content: *space-between*;    gap: 10px;    padding: 8px 10px;    border: 1px *solid* var(--line);    border-radius: 10px;    background: #fff;    cursor: *pointer*;  }  .weekItem:hover {    background: #f9fafb;  }  .weekItem.active {    outline: 2px *solid* var(--accent);    border-color: var(--accent);  }  .weekItem .date {    font-size: 12px;    color: #6b7280;  }  .formInput {    border: 1px *solid* #d1d5db;    border-radius: 10px;    padding: 10px 12px;    background: #fff;    font-size: 20px;  }  .bottomBar {    position: *sticky*;    bottom: 0;    display: *flex*;    align-items: *center*;    justify-content: *space-between*;    padding: 12px clamp(16px, 3vw, 28px);    background: rgba(255, 255, 255, 0.9);    border-top: 1px *solid* var(--line);    backdrop-filter: blur(6px);  }  .leftRow,  .rightRow {    display: *flex*;    gap: 8px;    align-items: *center*;  }  .muted {    color: #6b7280;    font-size: 13px;  }  @media (max-width: 1200px) {    .grid {      grid-template-columns: 240px 1fr 300px;    }  }  @media (max-width: 980px) {    .layout {      grid-template-columns: 1fr;    }    .grid {      grid-template-columns: 1fr;    }    .toolCol {    display: *flex*;    flex-direction: *column*;    gap: 20px;      position: *sticky*;    top: 24px;    max-height: calc(100vh - 48px);    overflow-y: *auto*;  }  } |

|  |
| --- |
| *// /src/app/api/univer\_city/attendance\_check/route.js*  import { db } from "@/lib/db";  import { NextResponse } from 'next/server';  function mapAttendanceStatus(hours) {      if (hours === 0) return { status: '출석', minutes: 0 };      if (hours === 1) return { status: '지각', minutes: 10 };      if (hours >= 2) return { status: '결석', minutes: 0 };      return { status: '미처리', minutes: 0 };  }  export async function GET(request) {      const { searchParams } = new URL(request.url);      const courseId = searchParams.get('courseId');      const month = searchParams.get('month'); *// YYYY-MM*      if (!courseId || !month) {          return NextResponse.json({ message: '강의 ID와 월 정보가 필요합니다.' }, { status: 400 });      }      const startDate = `${month}-01`;      const [year, monthNum] = month.split('-').map(Number);      const nextMonthDate = new Date(year, monthNum, 1);      const nextMonth = nextMonthDate.toISOString().split('T')[0];      const query = `          SELECT              T1.attendance\_date,              T3.user\_id AS studentId,              T3.user\_name AS studentName,              T1.absent\_hours,              T1.note  -- attendance 테이블에 메모 필드가 있다고 가정          FROM              attendance T1          JOIN              enrollment T2 ON T1.enrollment\_id = T2.enrollment\_id          JOIN              student T4 ON T2.student\_id = T4.student\_id          JOIN              user T3 ON T4.student\_id = T3.user\_id          WHERE              T2.lecture\_id = ? AND T1.attendance\_date >= ? AND T1.attendance\_date < ?          ORDER BY              T1.attendance\_date, T3.user\_id;      `; *//*      try {          const [results] = await db.execute(query, [courseId, startDate, nextMonth]);            const attendanceData = results.map(row => {              const { status, minutes } = mapAttendanceStatus(row.absent\_hours);              return {                  date: row.attendance\_date ? row.attendance\_date.toString().substring(0, 10) : null,                  studentId: row.studentId,                  studentName: row.studentName,                  attendance\_status: status,                  late\_minutes: minutes,                  memo: row.note || '',              };          });            return NextResponse.json(attendanceData, { status: 200 });      } catch (error) {          console.error('월별 출결 데이터 조회 중 서버 오류 발생:', error);          return NextResponse.json({ message: 'Internal Server Error' }, { status: 500 });      }  } |

|  |
| --- |
| *// src/app/api/univer\_city/attendance\_data/route.js*  export const runtime = "nodejs";  import { db } from "../../../../lib/db";  import { NextResponse } from 'next/server';  export async function GET(request) {      let connection;      try {          const { searchParams } = new URL(request.url);          const lectureId = searchParams.get('lectureId');          const attendanceDate = searchParams.get('date');          if (!lectureId || !attendanceDate) {              return NextResponse.json({ success: false, message: '강의 ID와 날짜가 필요합니다.' }, { status: 400 });          }          connection = await db.getConnection();          const query = `              SELECT                  u.user\_id AS studentId, -- 클라이언트에서 studentId로 사용                  u.user\_name AS name,                  COALESCE(a.absent\_hours, 0) AS absent\_hours, -- 출석 기록이 없으면 0 (출석)으로 간주                  e.enrollment\_id              FROM                  user u              JOIN                  student s ON u.user\_id = s.student\_id              JOIN                  enrollment e ON s.student\_id = e.student\_id              LEFT JOIN -- LEFT JOIN을 사용하여 출석 기록이 없는 학생도 포함 (미처리/출석)                  attendance a                  ON e.enrollment\_id = a.enrollment\_id AND a.attendance\_date = ? -- 해당 날짜만 필터링              WHERE                  e.lecture\_id = ? -- 해당 강의만 필터링              ORDER BY                  u.user\_id;          `;            const [results] = await connection.query(query, [attendanceDate, lectureId]);            const studentsWithStatus = results.map(student => {              let status = '미처리';              if (student.absent\_hours === 0) {                  status = '출석';              } else if (student.absent\_hours === 1) {                  status = '지각';              } else if (student.absent\_hours >= 2) {                  status = '결석';              }                return {                  ...student,                  id: student.studentId,                  attendance\_status: status,                  memo: '',                  late\_reason: '',              };          });          return NextResponse.json(studentsWithStatus, { status: 200 });      } catch (error) {          console.error('출석 데이터 조회 중 서버 오류 발생:', error);          return NextResponse.json({              success: false,              message: 'DB 조회 중 서버 오류가 발생했습니다.',              detail: error.message,          }, { status: 500 });      } finally {          if (connection) {              connection.release();          }      }  } |

|  |
| --- |
| *// src/app/api/univer\_city/insert\_attendance/route.js*  import { db } from "@/lib/db";  import { NextResponse } from "next/server";  export const runtime = "nodejs";  export async function POST(request) {      let connection;      try {          const body = await request.json();          const { lectureId, attendanceDate, professorId, records } = body;            if (!lectureId || !attendanceDate || !professorId || !records || records.length === 0) {              return new NextResponse(JSON.stringify({ message: "필수 데이터(강의ID, 날짜, 교수ID, 기록)가 누락되었습니다." }), { status: 400 });          }            const statusToDbMap = {              'present': '출석',              'late': '지각',              'absent': '결석',              'leave': '조퇴',              '미처리': '미처리'          };          connection = await db.getConnection();          await connection.beginTransaction();          for (const record of records) {              const dbStatus = statusToDbMap[record.status] || '미처리';              const studentId = record.studentId;              const [existingRows] = await connection.execute(                  `SELECT COUNT(\*) AS count FROM attendance WHERE lecture\_id = ? AND attendance\_date = ? AND student\_id = ?`,                  [lectureId, attendanceDate, studentId]              );              if (existingRows[0].count > 0) {                  await connection.execute(                      `                      UPDATE attendance                      SET attendance\_status = ?, reason = ?, late\_reason = ?, recorded\_by = ?                      WHERE lecture\_id = ? AND attendance\_date = ? AND student\_id = ? `,                      [dbStatus, record.memo || '', record.lateReason || '', professorId, lectureId, attendanceDate, studentId]                  );              } else {                  await connection.execute(                      `                      INSERT INTO attendance (lecture\_id, student\_id, attendance\_date, attendance\_status, reason, late\_reason, recorded\_by)                      VALUES (?, ?, ?, ?, ?, ?, ?) `,                      [lectureId, studentId, attendanceDate, dbStatus, record.memo || '', record.lateReason || '', professorId]                  );              }          }          await connection.commit(); /          connection.release();          return NextResponse.json({ message: "출석 정보가 성공적으로 저장/수정되었습니다." }, { status: 200 });      } catch (e) {          if (connection) {              await connection.rollback();              connection.release();          }          console.error('출석 데이터 저장 오류:', e);          return new NextResponse(JSON.stringify({ message: "서버 오류로 인해 저장에 실패했습니다.", error: e.message }), { status: 500 });      }  } |

|  |
| --- |
| *// src/app/api/insert/route.js*  export const runtime = "nodejs";  import { db } from "@/lib/db";  export async function GET(request) {    try {      const { searchParams } = new URL(request.url);      const table = searchParams.get("table");      const colsParam = searchParams.get("cols");      const valuesParam = searchParams.get("values");      if (!table) {        return Response.json({ error: "Missing 'table' query parameter" }, { status: 400 });      }      if (!valuesParam) {        return Response.json({ error: "Missing 'values' query parameter" }, { status: 400 });      }      let colsClause = "";      if (colsParam) {        if (colsParam.trim().startsWith("[")) {          try {            const arr = JSON.parse(colsParam);            if (Array.isArray(arr)) {              colsClause = `(${arr.join(", ")})`;            } else {              colsClause = `(${String(colsParam)})`;            }          } catch {            colsClause = `(${String(colsParam)})`;          }        } else {          colsClause = `(${String(colsParam)})`;        }      } else {        return Response.json({ error: "Missing 'cols' query parameter" }, { status: 400 });      }      let sql = "";      let result;      if (valuesParam.trim().startsWith("[")) {        try {          const parsed = JSON.parse(valuesParam);          let rows = [];          if (Array.isArray(parsed) && parsed.length > 0 && Array.isArray(parsed[0])) {            rows = parsed;          } else if (Array.isArray(parsed)) {            rows = [parsed];          } else {            const rawValues = String(valuesParam);            sql = `INSERT INTO ${table} ${colsClause} VALUES ${rawValues}`;            [result] = await db.query(sql);            return Response.json({ sql, affectedRows: result?.affectedRows ?? null, insertId: result?.insertId ?? null });          }          const one = `(${rows[0].map(() => "?").join(", ")})`;          const placeholders = rows.map(() => one).join(", ");          const params = rows.flat();          sql = `INSERT INTO ${table} ${colsClause} VALUES ${placeholders}`;          const [execRes] = await db.execute(sql, params);          result = execRes;        } catch {          const rawValues = String(valuesParam);          sql = `INSERT INTO ${table} ${colsClause} VALUES ${rawValues}`;          const [qRes] = await db.query(sql);          result = qRes;        }      } else {        const rawValues = String(valuesParam);        sql = `INSERT INTO ${table} ${colsClause} VALUES ${rawValues}`;        const [qRes] = await db.query(sql);        result = qRes;      }      return Response.json({        sql,        affectedRows: result?.affectedRows ?? null,        insertId: result?.insertId ?? null,      });    } catch (e) {      const payload = {        error: String(e?.message || e),        name: e?.name,        code: e?.code,        errno: e?.errno,        sqlMessage: e?.sqlMessage,        sqlState: e?.sqlState,        stack: e?.stack,        node: process?.versions?.node,        platform: process?.platform,      };      console.error("[/api/insert] Error:", payload);      return new Response(JSON.stringify(payload, null, 2), {        status: 500,        headers: { "content-type": "application/json" },      });    }  } |

ㅇ