

DB 프로그래밍 프로젝트



수 강 과 목 DB 프로그래밍

담 당 교 수 김수현 교수님

학 과 컴퓨터소프트웨어공학과

학 년 3학년

학 번 20233531

이 름 한정민

제 출 일 자 2025.06.12.

- 목 차 -

1. 프로젝트 개요 : 대학생 맞춤형 자취방 정보 관리 시스템	3
- 프로젝트 개요	3
- 프로젝트 목적	3
2. 시스템 요구사항 분석	4
3. 데이터베이스 설계	6
- 데이터 모델링	6
- E-R 다이어그램	13
- 테이블 정의 및 설명	18
4. 데이터베이스 정규화	24
- 제1정규형 (1NF)	25
- 제2정규형 (2NF)	26
- 제3정규형 (3NF)	27
- 최종 정규화 된 스키마	28
5. 사용자 인터페이스	30
- PHP 파일 연동	30
- 주요 기능 설명 및 주요 코드	30
6. 결론	59
- 프로젝트 결과	59
- 개선사항	59
- 느낀점	60

1. 서론

1-1. 프로젝트 개요

제가 고안한 프로젝트는 대학생 맞춤형 자취방 정보를 열람하고 검색할 수 있는 시스템입니다. 저는 실제로 자취방을 구하는 과정에서 자취방을 구하는데 실제 후기나 방 구조를 보기 힘들어 어려움을 겪었고 이 문제를 해결하기 위해 대학생들을 위한 자취방 정보 관리 시스템을 생각하게 되었습니다.

이 서비스를 통하여 대학생들이 자취방을 구할 때 보다 쉽게 지역, 가격, 평수, 생활 옵션, 후기 등 다양한 조건에 따라 자취방을 검색하고, 신뢰도 높은 실제 사용자 후기를 통해 보다 합리적이고 편리한 거주지를 쉽게 찾아볼 수 있는 서비스를 제공하고자 합니다.

1-2. 프로젝트 목적

최근 대학가 주변의 자취방 수요는 지속적으로 증가하고 있으며, 특히 신입생이나 타지역에서 학교를 다니는 대학생들에게는 거주 공간을 선택하는 과정이 매우 중요한 문제로 여기지고 있습니다.

기존의 커뮤니티나 부동산 앱은 상업적 목적이 강하거나 후기를 볼 수 없거나 신뢰도가 낮은 경우가 많아, 실제 거주자가 작성한 솔직한 정보가 부족한 경우가 다반수입니다. 이 서비스를 통해 지역, 가격, 평수, 생활 옵션 및 실제 사용자 후기를 포함한 자취방 정보를 한눈에 열람할 수 있는 서비스를 제공하여 대학생들이 자취방을 찾을 때 겪는 정보 부족, 신뢰성 문제, 검색의 불편함 등을 해소 할 수 있는 맞춤형 자취방 정보 통합 시스템을 구축하는 것입니다.

2. 시스템 요구사항 분석

1. 사용자는 시스템에 가입하기 위해 이름, 이메일, 비밀번호를 입력해야 한다.
2. 이메일은 사용자 식별을 위한 고유 값이며, 중복 가입이 불가능해야 한다.
3. 로그인한 사용자는 전체 자취방 목록을 열람할 수 있으며, 지역·가격·평수·옵션 등의 조건으로 자취방을 검색할 수 있어야 한다.
4. 사용자는 자취방 목록 중 원하는 방의 상세 정보를 열람할 수 있어야 하며, 상세 정보에는 주소, 가격, 평수, 설명, 옵션, 이미지, 등록일, 후기 등이 포함되어야 한다.
5. 사용자는 자신이 관심 있는 자취방을 북마크 할 수 있어야 하며, 북마크한 자취방은 개인 마이페이지를 통해 확인할 수 있어야 한다.
6. 사용자는 자신이 열람한 자취방에 대해 별점(1~5점)과 텍스트로 후기를 작성할 수 있어야 하며, 하나의 자취방에 대해 한 번만 작성할 수 있어야 한다.
7. 자취방에 작성된 모든 후기는 누구나 열람할 수 있어야 하며, 최신순 정렬 기능을 지원해야 한다.
8. 사용자는 자취방에 대해 좋아요를 누를 수 있어야 하며, 좋아요는 동일한 사용자가 동일 방에 한 번만 누를 수 있어야 한다.
9. 좋아요 수는 자취방 정보에 표시되며, 인기순 정렬이나 추천 기능에 활용될 수 있어야 한다.
10. 사용자는 마이페이지를 통해 자신이 쓴 후기, 북마크 목록, 로그아웃 기능에 접근할 수 있어야 한다.
11. 관리자는 전체 자취방에 대해 등록, 수정, 삭제할 수 있어야 하며, 자취방 등록 시 옵션 항목도 함께 지정할 수 있어야 한다.
12. 관리자는 자취방에서 제공되는 옵션 목록을 관리할 수 있어야 하며, 옵션 항목을 추가하거나 삭제할 수 있어야 한다.
13. 관리자는 사용자들이 작성한 후기 중 부적절한 내용을 삭제할 수 있어야 하며,

악성 사용자를 제한하거나 계정을 삭제할 수 있어야 한다.

14. 사용자는 자신의 계정을 탈퇴할 수 있으며, 탈퇴 시 관련된 후기, 북마크, 좋아요 등의 정보도 함께 삭제되어야 한다.
15. 모든 데이터의 무결성을 위해 이메일은 고유해야 하며, 자취방-옵션은 다대다 관계로 구성되어야 하고, 후기와 좋아요는 중복되지 않도록 복합키로 관리되어야 한다.

3. 데이터 베이스 설계

3-1. 데이터 모델링

3-1-1. 개념적 설계

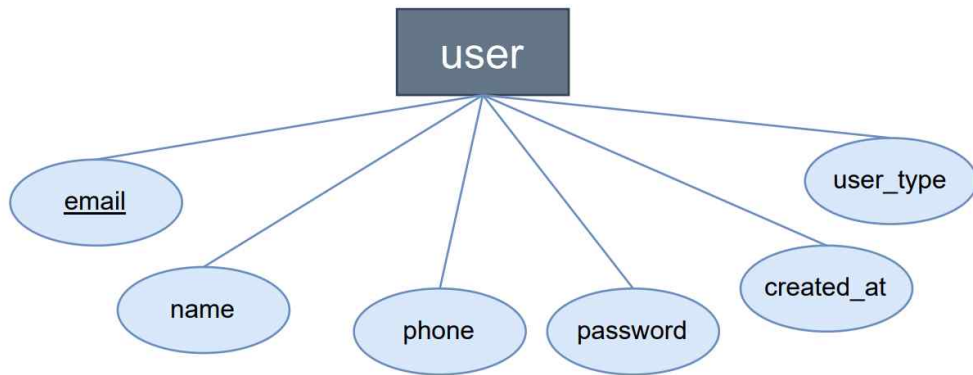
- 개체, 속성 추출

개체는 저장할 만한 가치가 있는 중요 데이터를 가진 데이터입니다. 요구사항 문장에서 업무와 관련이 깊은 의미 있는 명사를 찾아 개체와 속성으로 분리했습니다.

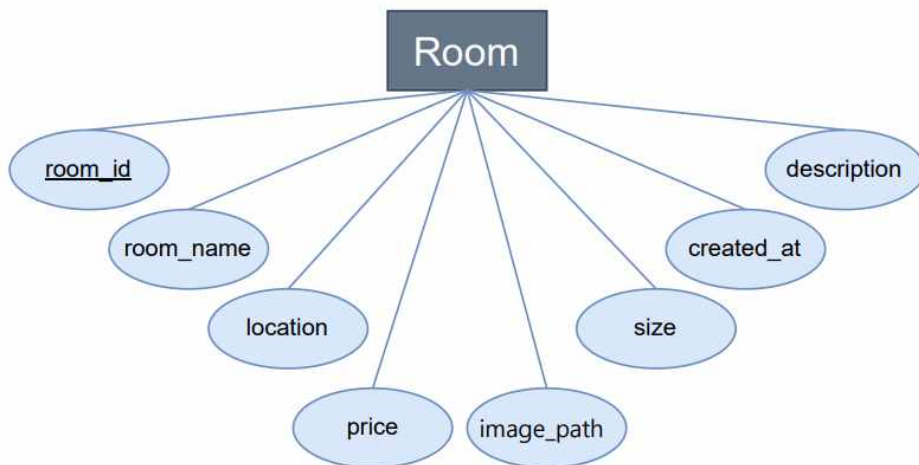
- 사용자는 시스템에 가입하기 위해 이름, 이메일, 비밀번호를 입력해야 한다.
- 이메일은 사용자 식별을 위한 고유 값이며, 중복 가입이 불가능해야 한다.
- 로그인한 사용자는 전체 자취방 목록을 열람할 수 있으며, 지역·가격·평수 등의 조건으로 자취방을 검색할 수 있어야 한다.
- 사용자는 자취방 목록 중 원하는 방의 상세 정보를 열람할 수 있어야 하며, 상세 정보에는 주소, 가격, 평수, 설명, 이미지, 등록일, 후기 등이 포함되어야 한다.
- 사용자는 자신이 열람한 자취방에 대해 별점(1~5점)과 텍스트로 후기를 작성할 수 있어야 하며, 하나의 자취방에 대해 한 번만 작성할 수 있어야 한다.
- 좋아요 수는 자취방 정보에 표시되며, 인기순 정렬이나 추천 기능에 활용될 수 있어야 한다.
- 관리자는 전체 자취방에 대해 등록, 수정, 삭제할 수 있어야 하며, 자취방 등록 시 옵션 항목도 함께 지정할 수 있어야 한다.
- 관리자는 자취방에서 제공되는 옵션 목록을 관리할 수 있어야 하며, 옵션 항목을 추가하거나 삭제할 수 있어야 한다.

요구사항을 분석한 후 추출한 요구사항 기반 개체와 속성들을 정리하면 아래 표처럼 나타납니다.

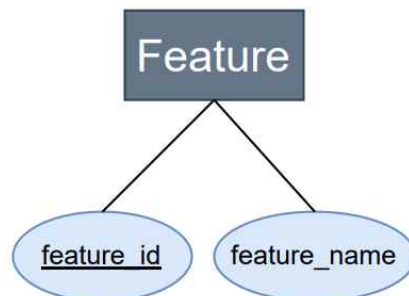
개체	속성
User(사용자)	user_id, name, email, phone, password, user_type, create_at
Room(자취방)	room_id, room_name, description, location, price, size, created_at, image_path, created_at
Feature(옵션)	feature_id, feature_name



[user 개체 E-R 다이어그램]



[Room 개체 E-R 다이어그램]



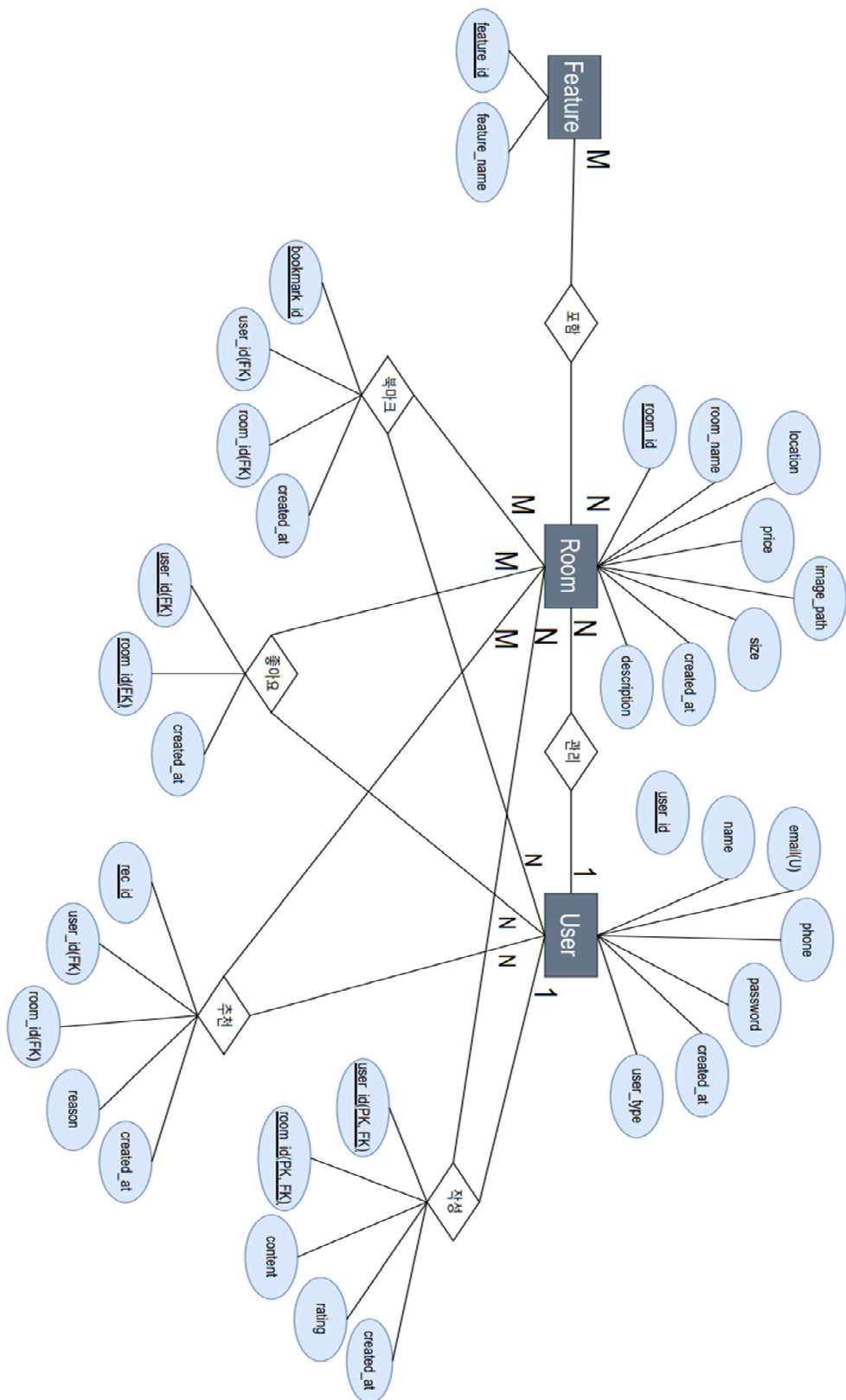
[Feature 개체 E-R 다이어그램]

- 관계 추출

관계는 개체 간의 의미 있는 연관성입니다. 일반적으로 요구사항 문장에서 개체간의 연관성을 의미 있게 표현한 동사로 표현됩니다. 의미가 같은 동사가 여러개일 경우에는 대표 동사를 하나만 선택하였습니다.

- 사용자는 자취방 목록 중 원하는 방의 상세 정보를 열람할 수 있어야 하며, 상세 정보에는 주소, 가격, 평수, 설명, 이미지, 등록일, 후기 등이 포함되어야 한다.
- 사용자는 자신이 관심 있는 자취방을 북마크 할 수 있어야 하며, 북마크한 자취방은 개인 마이페이지를 통해 확인할 수 있어야 한다.
- 사용자는 자신이 열람한 자취방에 대해 별점(1~5점)과 텍스트로 후기를 작성할 수 있어야 하며, 하나의 자취방에 대해 한 번만 작성할 수 있어야 한다.
- 사용자는 자취방에 대해 좋아요를 누를 수 있어야 하며, 좋아요는 동일한 사용자가 동일 방에 한 번만 누를 수 있어야 한다.
- 관리자는 전체 자취방에 대해 등록, 수정, 삭제할 수 있어야 하며, 자취방 등록 시 옵션 항목도 함께 지정할 수 있어야 한다.

관계	관계에 참여하는 개체	관계 유형	속성
관리	User-Room	1:N	X
작성	User-Room	M:N	rating, content, created_at
좋아요	User-Room	M:N	X
북마크	User-Room	M:N	X
포함	Room-Feature	M:N	X



[전체 E-R 다이어그램]

3-1-2. 논리적 설계

규칙 1 : 모든 개체를 릴레이션으로 변환한다.

- 개체를 릴레이션으로 변환하는 규칙입니다. 시스템에서 정의된 각 개체는 하나의 테이블로 변환됩니다. User 개체는 User 테이블로 변환되고, user_id, name, email 등의 속성을 가집니다.
- Room 개체는 Room 테이블로 변환되어 room_id, room_name, price 등의 속성을 가집니다. Feature 개체는 Feature 테이블로 변환되어 feature_id, feature_name 등의 속성을 가집니다.
- 이와 같이 모든 개체는 하나의 릴레이션으로 변환되어 데이터베이스에 저장됩니다.

[User 릴레이션]

<u>user_id</u>	name	email(UNIQUE)	phone	password	created_at	user_type
----------------	------	---------------	-------	----------	------------	-----------

[Room 릴레이션]

<u>room_id</u>	user_id(FK)	room_name	location	price	size	image_path	created_at	description
----------------	-------------	-----------	----------	-------	------	------------	------------	-------------

[Feature 릴레이션]

<u>feature_id</u>	feature_name
-------------------	--------------

규칙 2 : 다대다(N:M) 관계는 릴레이션으로 변환한다.

- 다대다(N:M) 관계는 중간 테이블을 사용하여 두 개체를 연결하는 릴레이션으로 변환됩니다.
- Room 테이블과 Feature 개체는 다대다 관계입니다. 하나의 자취방은 여러 옵션을 가질 수 있고, 하나의 옵션은 여러 자취방에 적용될 수 있습니다.
- 이를 위해 중간 테이블인 RoomFeature(포함) 테이블을 사용하여 Room과 Feature을 연결합니다.

[포함 릴레이션]

<u>room_id(PK, FK)</u>	<u>feature_id(PK, FK)</u>
------------------------	---------------------------

[작성 릴레이션]

<u>user_id(PK, FK)</u>	<u>room_id(PK, FK)</u>	content	rating	created_at
------------------------	------------------------	---------	--------	------------

[북마크 릴레이션]

<u>room_id</u> (PK, FK)	user_id(PK, FK)	created_at
-------------------------	-----------------	------------

[좋아요 릴레이션]

<u>user_id</u> (PK, FK)	room_id(PK, FK)	created_at
-------------------------	-----------------	------------

규칙 3 : 일대다(1:N) 관계는 외래키로 표현한다.

- 일대다(1:N) 관계는 외래키를 사용하여 한 테이블의 속성이 다른 테이블을 참조하도록 변환합니다.
- User와 Room은 1:N 관계를 가집니다. 한 명의 사용자는 여러 자취방을 등록할 수 있지만, 하나의 자취방은 하나의 사용자에게 의해서만 등록될 수 있습니다. 이를 나타내기 위해 Room 테이블에 user_id를 외래키로 표현합니다.

[Room 릴레이션]

<u>room_id</u>	user_id(FK)	room_name	location	price	size	image_path	created_at	description
----------------	-------------	-----------	----------	-------	------	------------	------------	-------------

규칙 4: 일대일(1:1) 관계는 외래키로 표현한다.

- 일대일(1:1) 관계는 외래키를 사용하여 두 테이블 간의 관계를 표현합니다. 일반적으로 일대일 관계는 하나의 테이블에 외래키를 두거나, 두 테이블을 병합하여 하나의 테이블로 관리할 수 있습니다.

해당하는 값이 없습니다.

규칙 5: 다중 값 속성은 릴레이션으로 변환

- 다중 값 속성은 한 개체가 여러 값을 가질 수 있는 속성입니다. 해당 속성은 별도의 릴레이션으로 변환됩니다.

해당하는 값이 없습니다.

최종 릴레이션

[User 릴레이션]

<u>user_id</u>	name	email(UNIQUE)	phone	password	created_at	user_type
----------------	------	---------------	-------	----------	------------	-----------

[Room 릴레이션]

<u>room_id</u>	user_id(FK)	room_name	location	price	size	image_path	created_at	description
----------------	-------------	-----------	----------	-------	------	------------	------------	-------------

[Feature 릴레이션]

<u>feature_id</u>	feature_name
-------------------	--------------

[포함 릴레이션]

<u>room_id(PK, FK)</u>	feature_id(PK, FK)
------------------------	--------------------

[작성 릴레이션]

<u>user_id(PK, FK)</u>	<u>room_id(PK, FK)</u>	content	rating	created_at
------------------------	------------------------	---------	--------	------------

[북마크 릴레이션]

<u>room_id(PK, FK)</u>	<u>user_id(PK, FK)</u>	created_at
------------------------	------------------------	------------

[좋아요 릴레이션]

<u>user_id(PK, FK)</u>	<u>room_id(PK, FK)</u>	created_at
------------------------	------------------------	------------

3-1-3. 물리적 설계

```
-- 데이터베이스 생성 및 사용
CREATE DATABASE living_alone;
USE living_alone;
```

● 사용자 테이블 생성 쿼리문

```
CREATE TABLE User (
  user_id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  phone VARCHAR(20),
  password VARCHAR(255) NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  user_type ENUM('user', 'admin') DEFAULT 'user'
);
```

-- 사용자 고유 ID
-- 사용자 이름
-- 이메일 (로그인 ID로 사용, 중복 불가)
-- 연락처
-- 비밀번호
-- 가입 일시
-- 사용자 유형 (일반 or 관리자)

● 자취방 테이블 생성 쿼리문

```
CREATE TABLE Room (
  room_id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  room_name VARCHAR(100) NOT NULL,
  location VARCHAR(255),
  price DECIMAL(10, 2),
  size FLOAT,
  image_path TEXT,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  description TEXT,
  FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE
);
```

-- 자취방 고유 ID
-- 등록한 사용자 ID (방 등록자)
-- 자취방 이름
-- 위치 정보
-- 월세
-- 평수
-- 이미지 경로
-- 등록 일시
-- 상세 설명

● 옵션(Future) 테이블 생성 쿼리문

```
CREATE TABLE Feature (
  feature_id INT AUTO_INCREMENT PRIMARY KEY,
  feature_name VARCHAR(100) UNIQUE NOT NULL
);
```

-- 옵션 고유 ID
-- 옵션 이름 (중복 불가)

● 방-옵션 중간 테이블 생성 쿼리문

```
CREATE TABLE RoomFeature (
  room_id INT NOT NULL,
  feature_id INT NOT NULL,
  PRIMARY KEY (room_id, feature_id),
  FOREIGN KEY (room_id) REFERENCES Room(room_id) ON DELETE CASCADE,
  FOREIGN KEY (feature_id) REFERENCES Feature(feature_id) ON DELETE CASCADE
);
```

-- 자취방 ID
-- 옵션 ID
-- 복합 기본키

● 리뷰 테이블 생성 쿼리문

```

CREATE TABLE Review (
    user_id INT NOT NULL,           -- 작성자 ID
    room_id INT NOT NULL,          -- 대상 자취방 ID
    content TEXT,                  -- 후기 텍스트
    rating INT CHECK (rating BETWEEN 1 AND 5), -- 별점 (1~5점 사이만 허용)
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP, -- 후기 작성일시
    PRIMARY KEY (user_id, room_id), -- 하나의 유저는 하나의 방에만 한 번 후기 작성 가능
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
    FOREIGN KEY (room_id) REFERENCES Room(room_id) ON DELETE CASCADE
);

```

● 북마크 테이블 생성 쿼리문

```

CREATE TABLE Bookmark (
    user_id INT NOT NULL,          -- 북마크한 사용자
    room_id INT NOT NULL,          -- 북마크한 자취방
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP, -- 북마크한 시각
    PRIMARY KEY (user_id, room_id), -- 같은 유저가 같은 방 북마크 1회만 가능
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
    FOREIGN KEY (room_id) REFERENCES Room(room_id) ON DELETE CASCADE
);

```

● 좋아요 테이블 생성 쿼리문

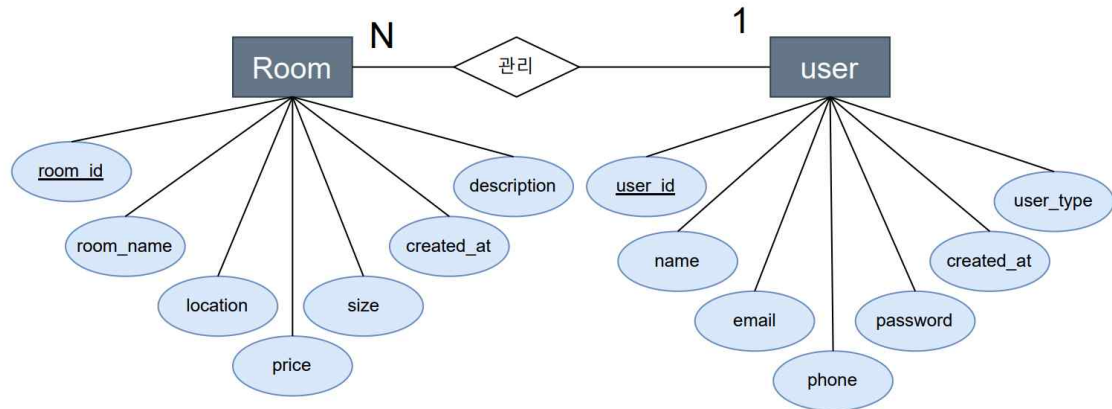
```

CREATE TABLE Likes (
    user_id INT NOT NULL,          -- 좋아요 누른 사용자
    room_id INT NOT NULL,          -- 좋아요 누른 자취방
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP, -- 좋아요 누른 시각
    PRIMARY KEY (user_id, room_id), -- 같은 유저가 같은 방 좋아요 한 번만 가능
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
    FOREIGN KEY (room_id) REFERENCES Room(room_id) ON DELETE CASCADE
);

```

3-2. E-R 다이어그램

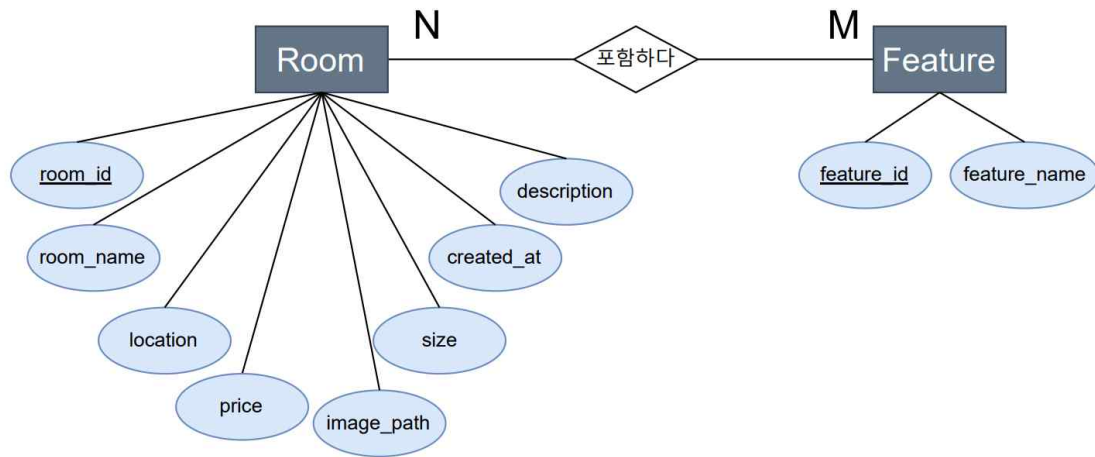
- Room 테이블과 user 테이블의 관계



[Room 테이블과 user 테이블의 E-R 다이어그램]

- User 테이블과 Room 테이블은 1:N(일대다) 관계로 정의됩니다. 한명의 사용자가 여러 자취방을 등록, 수정, 삭제 할 수 있습니다. 사용자 정보를 저장하고 있는 User 테이블의 기본키인 user_id는 Room 테이블의 외래키로 연결됩니다. 이 다이어그램을 통해 각 방이 어떤 사용자에게 의해 등록되었는지 구분 할 수 있습니다.
- 사용자가 방을 등록하게 되면, 해당 방에 대한 정보가 Room 테이블에 저장되고 동시에 등록자의 user_id도 함께 기록됩니다.

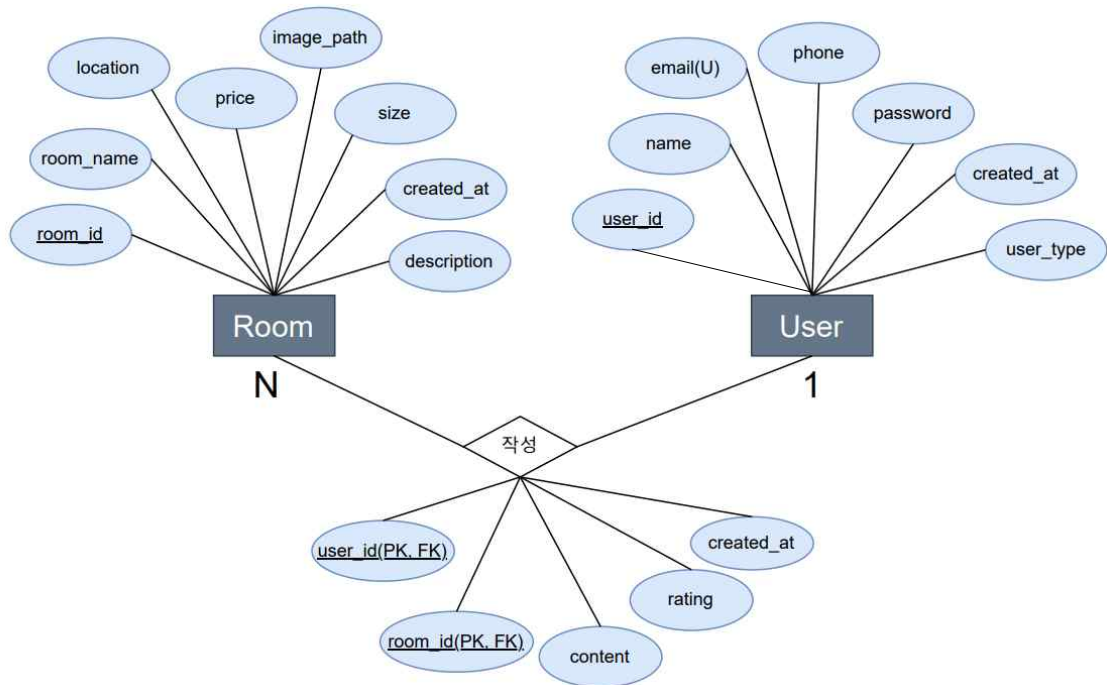
- Room 테이블과 Feature 테이블의 관계



[Room 테이블과 Feature 테이블의 E-R 다이어그램]

- Room 테이블과 Feature 테이블은 N:M(다대다) 관계로 정의됩니다. 이 두 테이블은 '포함하다'라는 관계로 연결되어 있습니다. 하나의 자취방(Room)은 여러 개의 옵션(Feature)을 포함할 수 있고, 하나의 옵션 역시 여러 자취방에 중복으로 저장될 수 있습니다.
- 실제 논리 또는 물리 설계 단계에서는 이 관계를 구현하기 위해 중간 테이블을 생성하게 됩니다. 중간 테이블인 포함 테이블에는 room_id와 feature_id 두 개의 외래키가 저장되며, 이를 통해 어떤 방에 어떤 옵션이 포함되어 있는지를 쉽게 나타낼 수 있습니다.

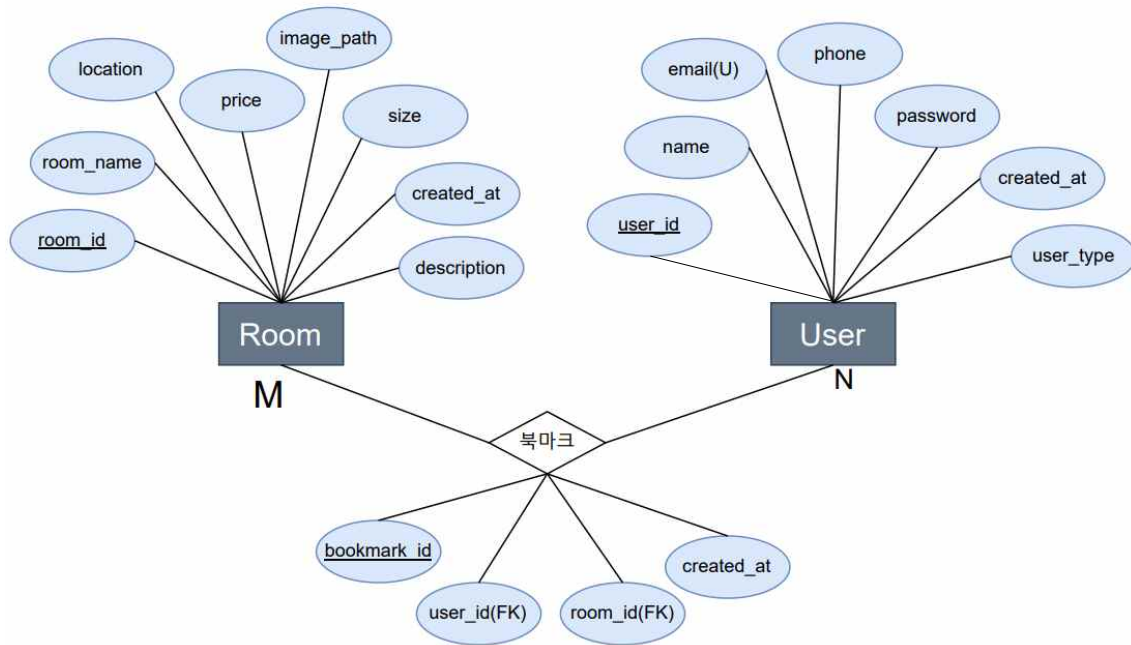
- User - 작성 - Room 테이블의 관계



[User - 작성 - Room 테이블의 E-R 다이어그램]

- User 테이블과 Room 테이블은 N:M(다대다) 관계로 정의됩니다. 두 테이블은 '작성'이라는 관계로 연결되어 있습니다. 한 명의 사용자는 여러 자취방에 대해 리뷰를 작성할 수 있고, 하나의 자취방은 여러 사용자로부터 리뷰를 받을 수 있습니다.
- '작성' 관계에는 리뷰 내용을 담는 content, 평점을 나타내는 rating, 작성 시점을 기록하는 created_at과 같은 관계 속성이 포함되며, 사용자 후기 정보(review)를 관리할 수 있도록 해 줍니다. 또한 동일 사용자가 동일 방에 중복 리뷰를 작성하지 못하도록 user와 room 간의 조합이 하나만 존재하게 해야합니다.

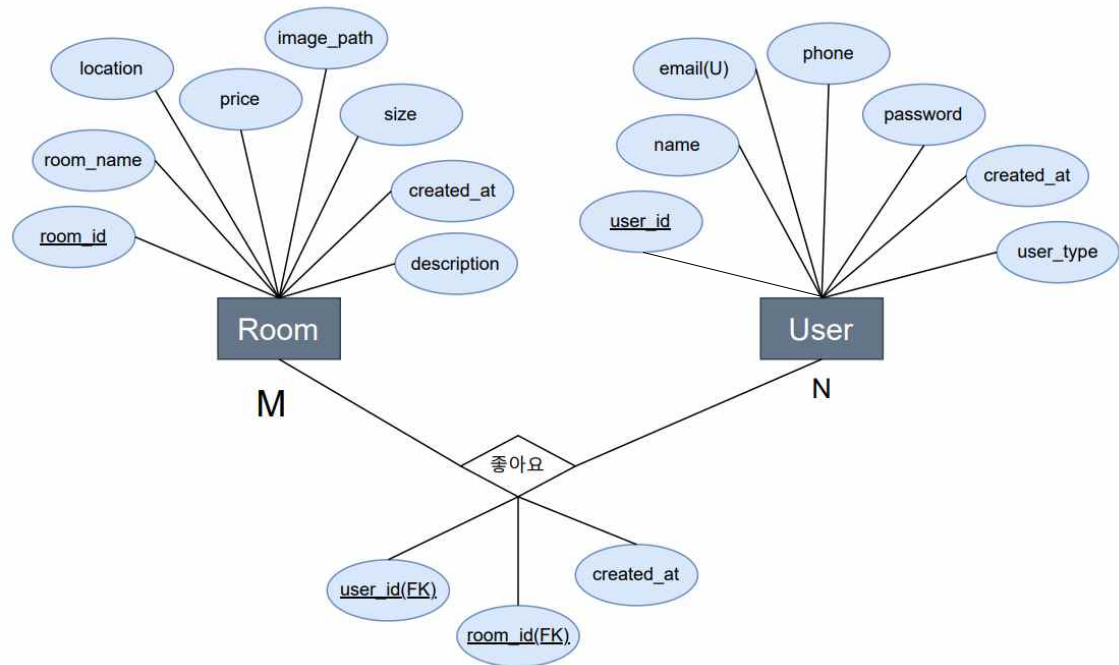
- User - 북마크 - Room 테이블의 관계



[User - 북마크 - Room 테이블의 E-R 다이어그램]

- User 테이블과 Room 테이블은 N:M(다대다) 관계로 정의됩니다. 이 두 테이블은 '북마크'라는 관계를 통해 연결되어 있습니다. 한 명의 사용자는 여러 자취방을 북마크할 수 있고, 하나의 자취방도 여러 사용자에 의해 북마크 될 수 있습니다.
- '북마크' 관계에는 북마크한 시점을 나타내는 created_at과 북마크 식별을 위한 bookmark_id 등의 관계 속성이 포함되어 있습니다.

- User - 좋아요 - Room 테이블의 관계



[User - 좋아요 - Room 테이블의 E-R 다이어그램]

- User 테이블과 Room 테이블은 N:M(다대다) 관계로 정의됩니다. 이 테이블은 '좋아요'라는 관계명을 통해 연결되어 있습니다. 하나의 사용자는 여러 자취방에 '좋아요'를 누를 수 있고, 하나의 자취방도 여러 사용자로부터 '좋아요'를 받을 수 있습니다.
- '좋아요' 관계에는 좋아요를 누른 시점을 나타내는 created_at이라는 관계 속성이 포함되어 있으며, 자취방 인기 순위를 정하는 데 활용될 수 있습니다.

3-3. 테이블 정의 및 설명

USER 테이블

- 자취방 서비스를 사용하는 회원 정보를 저장하는 테이블입니다. 자취방 서비스를 이용하는 사용자의 고유한 정보들을 관리합니다. 사용자 고유 ID, 이름, 이메일, 연락처, 비밀번호, 가입일, 사용자 유형(일반/관리자) 등을 저장합니다.

속성	설명	데이터타입	제약조건
user_id	사용자 고유 ID	INT	PRIMARY KEY, AUTO_INCREMENT
name	사용자 이름	VARCHAR(50)	NOT NULL
email	이메일 주소	VARCHAR(100)	NOT NULL, UNIQUE
phone	연락처	VARCHAR(20)	NULL
password	비밀번호	VARCHAR(225)	NOT NULL
user_type	사용자 유형	ENUM('user', 'admin')	DEFAULT 'user'
created_at	가입 일시	DATETIME	DEFAULT CURRENT_TIMESTAMP

Room 테이블

- 등록된 자취방 정보를 저장하는 테이블입니다. 각 방의 고유 ID, 이름, 위치, 월세 가격, 면적, 설명, 등록일을 저장합니다.

속성	설명	데이터타입	제약조건
room_id	방 고유 ID	INT	PRIMARY KEY, AUTO_INCREMENT
room_name	방 이름	VARCHAR(100)	NOT NULL
location	위치 정보	VARCHAR(200)	NOT NULL
price	월세 가격	INT	NOT NULL
size	평수	FLOAT	NOT NULL
description	상세 설명	TEXT	NULL
image_path	이미지 파일 경로	VARCHAR(255)	NULL 허용
created_at	등록 일시	DATETIME	DEFAULT CURRENT_TIMESTAMP

Feature 테이블

- 자취방에 포함될 수 있는 옵션을 저장하는 테이블입니다. 각 옵션은 고유 ID와 이름을 가지며, 여러 자취방에서 공통적으로 사용될 수 있습니다

속성	설명	데이터타입	제약조건
feature_id	옵션 고유 ID	INT	PRIMARY KEY, AUTO_INCREMENT
feature_name	옵션 이름	VARCHAR(50)	NOT NULL, UNIQUE

포함 테이블

- Romm 테이블과 Feature 테이블의 N:M(다대다) 관계를 관리하기 위한 중간 테이블입니다.

속성	설명	데이터타입	제약조건
room_id	자취방 ID	INT	PRIMARY KEY, FOREIGN KEY → Room(room_id), ON DELETE CASCADE
feature_id	옵션 ID	INT	PRIMARY KEY, FOREIGN KEY → feature(feature_id), ON DELETE CASCADE

Review(작성) 테이블

- 사용자가 작성한 자취방 후기를 저장하는 테이블입니다. 리뷰는 특정 자취방과 사용자에게 연결되며, 평점 (1~5), 텍스트 내용, 작성 일시 등의 정보를 포함합니다.

속성	설명	데이터타입	제약조건
user_id	작성자 ID	INT	PRIMARY KEY (user_id, room_id), FOREIGN KEY → User(user_id), ON DELETE CASCADE
room_id	대상 방 ID	INT	PRIMARY KEY (user_id, room_id), FOREIGN KEY → Room(room_id), ON DELETE CASCADE
content	후기내용	TEXT	NULL
rating	평점(1~5)	INT	CHECK (rating BETWEEN 1 AND 5), NOT NULL
created_at	등록 일시	DATETIME	DEFAULT CURRENT_TIMESTAMP

Bookmark(북마크) 테이블

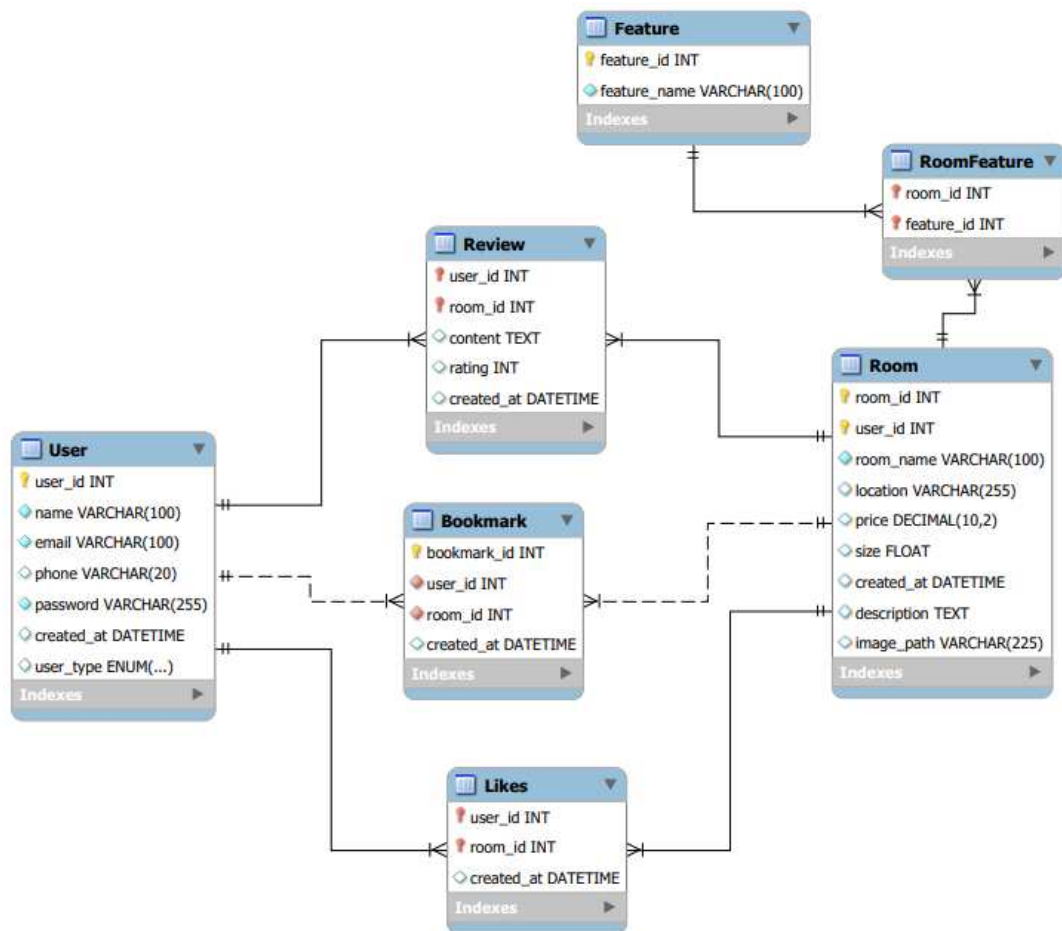
- 사용자가 관심 있는 자취방을 북마크한 기록을 저장합니다.

속성	설명	데이터타입	제약조건
bookmark_id	북마크 ID	INT	PRIMARY KEY, AUTO_INCREMENT
user_id	사용자 ID	INT	FOREIGN KEY → User(user_id), ON DELETE CASCADE
room_id	북마크 대상 방 ID	INT	FOREIGN KEY → Room(room_id), ON DELETE CASCADE
created_at	북마크 일시	DATETIME	DEFAULT CURRENT_TIMESTAMP

Likes(좋아요) 테이블

- 사용자가 자취방에 '좋아요'를 누른 기록을 저장합니다. 사용자와 자취방 간 중복을 허용하지 않습니다.

속성	설명	데이터타입	제약조건
user_id	좋아요한 사용자 ID	INT	PRIMARY KEY, FOREIGN KEY → User(user_id), ON DELETE CASCADE
room_id	좋아요한 방 ID	INT	PRIMARY KEY, FOREIGN KEY → Room(room_id), ON DELETE CASCADE
created_at	좋아요 일시	DATETIME	DEFAULT CURRENT_TIMESTAMP



[MySQL Workbench에서 각 쿼리문으로 직접 설계한 테이블]

4. 데이터 베이스 정규화

데이터 베이스 설계에서 데이터의 중복을 최소화 하고 데이터의 무결성을 유지하기 위해 사용되는 프로세스입니다.

정규화를 통해 데이터를 여러 릴레이션으로 나누어 중복 저장되는 데이터를 줄일 수 있고, 중복된 데이터를 수정할 때 발생하는 데이터 불일치 문제를 방지할 수 있습니다. 또한 정규화 된 데이터베이스는 구조가 논리적으로 잘 구성되어 있어 데이터 수정, 삭제, 삽입 등 유지보수가 용이하고, 데이터 검색과 수정이 간단하고 일관되게 이루어집니다. 마지막으로 데이터가 잘 구조화되어 있으면 릴레이션 간의 관계가 명확하기 때문에 새로운 릴레이션이나 속성을 추가할 때다른 데이터에 영향을 주지 않고 쉽게 확장할 수 있습니다.

```
-- User 테이블
CREATE TABLE User (
  user_id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(50),
  email VARCHAR(100),
  department_name VARCHAR(50)
);

-- Room 테이블
CREATE TABLE Room (
  room_id INT PRIMARY KEY AUTO_INCREMENT,
  room_name VARCHAR(100) NOT NULL,
  location VARCHAR(200) NOT NULL,
  price INT NOT NULL,
  size FLOAT NOT NULL,
  description TEXT,
  image_path VARCHAR(255),
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Feature 테이블
CREATE TABLE Feature (
  feature_id INT PRIMARY KEY AUTO_INCREMENT,
  feature_name VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE RoomFeature (
  room_id INT,
  feature_id INT,
  feature_name VARCHAR(50),
  PRIMARY KEY (room_id, feature_id)
);
```



```

-- Review 테이블
CREATE TABLE Review (
    user_id INT,
    room_id INT,
    content TEXT,
    rating INT NOT NULL CHECK (rating BETWEEN 1 AND 5),
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (user_id, room_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
    FOREIGN KEY (room_id) REFERENCES Room(room_id) ON DELETE CASCADE
);

-- Bookmark 테이블
CREATE TABLE Bookmark (
    bookmark_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    room_id INT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
    FOREIGN KEY (room_id) REFERENCES Room(room_id) ON DELETE CASCADE
);

-- Recommendation 테이블
CREATE TABLE Recommendation (
    rec_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    room_id INT,
    reason VARCHAR(200),
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
    FOREIGN KEY (room_id) REFERENCES Room(room_id) ON DELETE CASCADE
);

-- Likes 테이블
CREATE TABLE Likes (
    user_id INT,
    room_id INT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (user_id, room_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
    FOREIGN KEY (room_id) REFERENCES Room(room_id) ON DELETE CASCADE
);

```

[초기의 스키마]

4-1. 제 1 정규형 (1NF)

제 1정규형은 릴레이션에 속한 모든 속성의 도메인이 원자값(Atomic Value)을 가져야 한다는 특징이 있습니다.

초기 스키마는 이 요구사항을 만족합니다

4-2. 제 2 정규형 (2NF)

제 2 정규형에서는 모든 속성이 기본 키 전체에 완전 함수 종속되어야 합니다.

기본 키가 복합키일 경우, 기본 키의 일부에만 종속된 속성이 존재하면 2NF를 위반한 것입니다. 현재 스키마 중 RoomFeature 테이블은 (room_id, feature_id)를 복합 기본 키로 가지고 있습니다. 이 테이블 안에 feature_name 속성이 존재한다면, 이 값은 feature_id에만 종속되어 있습니다.

기본 키 전체가 아닌 feature_id 하나에만 종속된 속성이므로 부분 함수 종속에 해당하며, 2NF를 위반합니다. 이 문제를 해결하기 위해 feature_name 속성을 별도의 테이블인 Feature로 분리합니다. Feature 테이블은 feature_id를 기본 키로 하여, 각 옵션 이름을 고유하게 저장합니다.

RoomFeature 테이블은 room_id와 feature_id의 관계만 저장하며, feature_name에 직접 접근하지 않도록 설계하여 2정규형을 만족하도록 정규화합니다.

```
-- Feature 테이블
CREATE TABLE Feature (
  feature_id INT PRIMARY KEY AUTO_INCREMENT,
  feature_name VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE RoomFeature (
  room_id INT,
  feature_id INT,
  feature_name VARCHAR(50),
  PRIMARY KEY (room_id, feature_id)
);
```

[제 2 정규형을 만족하지 않는 스키마]

```
CREATE TABLE Feature (
  feature_id INT PRIMARY KEY AUTO_INCREMENT,
  feature_name VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE RoomFeature (
  room_id INT,
  feature_id INT,
  PRIMARY KEY (room_id, feature_id),
  FOREIGN KEY (room_id) REFERENCES Room(room_id) ON DELETE CASCADE,
  FOREIGN KEY (feature_id) REFERENCES Feature(feature_id) ON DELETE CASCADE
);
```

[제 2 정규형을 만족하는 스키마]

4-3. 제 3 정규형 (3NF)

제 3 정규형에서는 모든 속성이 기본 키에 직접적으로 종속되어야 하며, 기본 키가 아닌 속성 간의 종속(이행적 종속)이 존재하면 안 됩니다. 스키마 중 User 테이블의 구조를 보면, user_id가 기본 키입니다. 그 외 속성으로는 email, name, phone, password, user_type 등이 존재합니다.

email에는 UNIQUE 제약조건이 설정되어 있어 중복이 허용되지 않으며, name, phone, password, user_type은 email을 통해서 유추되거나 결정되지 않습니다.

속성 간에 실질적인 종속 관계가 존재하지 않으며, 모든 속성이 user_id에 직접적으로 종속되도록 설계되어 있습니다. User 테이블 내의 모든 속성은 user_id를 통해서만 유일하게 식별되며, 다른 일반 속성에 종속되지 않습니다. 정규화된 설계를 통해 이행적 종속 관계를 제거하고, 결과적으로 User 테이블은 제 3 정규형(3NF)을 만족하는 구조가 됩니다.

```
-- User 테이블
CREATE TABLE User (
  user_id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(50),
  email VARCHAR(100),
  department_name VARCHAR(50)
);
```

[제 3 정규형 만족하지 않는 스키마]

```
CREATE TABLE User (
  user_id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  phone VARCHAR(20),
  password VARCHAR(255) NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
  user_type ENUM('user', 'admin') DEFAULT 'user'
);
```

[제 3 정규형 만족하는 스키마]

4-4. 최종 정규화된 스키마

● Recommendation 테이블 제거

>> 초기 데이터베이스 설계 단계에서는 사용자가 자취방을 추천하고, 간단한 추천 이유를 남길 수 있는 기능을 고려하여 Recommendation 테이블을 포함하였습니다. 그러나 정규화 과정 및 전체 시스템 흐름을 재검토한 결과, 해당 기능은 다음과 같은 이유로 필요하지 않다고 판단되어 최종 스키마에서 제거하였습니다.

1. 역할 중복

- 자취방에 대한 사용자 반응을 표현하는 기능으로는 이미 '좋아요', '리뷰', '북마크'가 구현되어 있으며, Recommendation은 이들의 목적과 사용 방식이 유사하여 기능적인 중복이 발생합니다.

2. 기능 구현 대비 효과 부족

- 추천을 위한 별도의 입력 폼, 저장 로직, 관리자 검토 기능을 구현하는 데에 비해, 그 기능이 사용자 또는 관리자에게 실질적인 이점을 제공하기 어렵다고 판단하였습니다.

3. 시스템 유지보수 단순화

- 불필요한 테이블을 제거함으로써 데이터 베이스 구조를 보다 단순하게 유지하고, 기능 유지보수 및 확장 과정에서 혼란을 줄일 수 있습니다.

위와 같은 이유로 Recommendation 테이블은 실제 구현 및 최종 스키마에서 제외하였으며, 사용자 반응 수집은 '좋아요', '리뷰', '북마크' 기능을 통해 충분히 대체 가능하도록 설계하였습니다.

아래 스키마는 Recommendation 테이블을 제거한 최종 스키마입니다.

```
-- 사용자 테이블: 회원 계정 정보 저장
CREATE TABLE User (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(20),
    password VARCHAR(255) NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    user_type ENUM('user', 'admin') DEFAULT 'user'
);

-- 자취방 테이블: 방 상세 정보 저장
CREATE TABLE Room (
    room_id INT AUTO_INCREMENT PRIMARY KEY,
    room_name VARCHAR(100) NOT NULL,
    location VARCHAR(255),
    price DECIMAL(10, 2),
    size FLOAT,
    image_path TEXT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    description TEXT
);

-- 옵션 테이블: 자취방에 제공되는 옵션 목록 저장
CREATE TABLE Feature (
    feature_id INT AUTO_INCREMENT PRIMARY KEY,
    feature_name VARCHAR(100) UNIQUE NOT NULL
);

-- 자취방-옵션 관계 테이블 (N:M 매핑을)
CREATE TABLE RoomFeature (
    room_id INT NOT NULL,
    feature_id INT NOT NULL,
    PRIMARY KEY (room_id, feature_id),
    FOREIGN KEY (room_id) REFERENCES Room(room_id) ON DELETE CASCADE,
    FOREIGN KEY (feature_id) REFERENCES Feature(feature_id) ON DELETE CASCADE
);

-- 후기 테이블: 사용자들이 남긴 리뷰 정보 저장
CREATE TABLE Review (
    user_id INT NOT NULL,
    room_id INT NOT NULL,
    content TEXT,
    rating INT CHECK (rating BETWEEN 1 AND 5),
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (user_id, room_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
    FOREIGN KEY (room_id) REFERENCES Room(room_id) ON DELETE CASCADE
);

-- 북마크 테이블
CREATE TABLE Bookmark (
    user_id INT NOT NULL,
    room_id INT NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (user_id, room_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
    FOREIGN KEY (room_id) REFERENCES Room(room_id) ON DELETE CASCADE
);

-- 좋아요 테이블: 자취방 좋아요 기록
CREATE TABLE Likes (
    user_id INT NOT NULL,
    room_id INT NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (user_id, room_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
    FOREIGN KEY (room_id) REFERENCES Room(room_id) ON DELETE CASCADE
);
```

5. 사용자 인터페이스

5-1. PHP 파일 연동

```
<?php
$host = 'localhost'; $db = 'living_Room'; $user = 'root'; $pass = '0000';

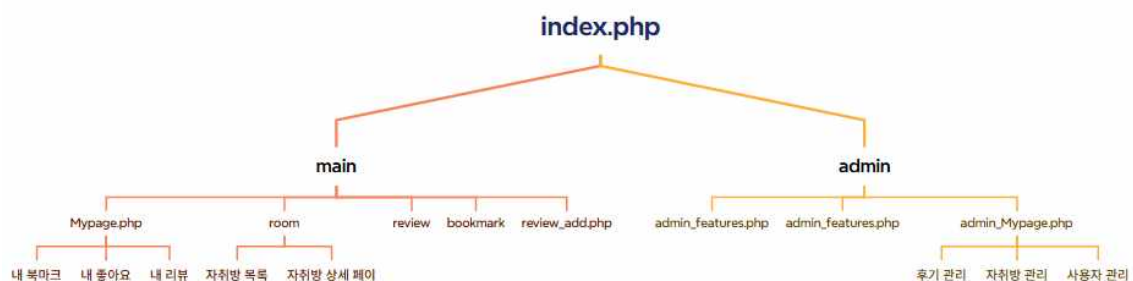
$conn = new mysqli(hostname: $host, username: $user, password: $pass, database: $db);

if ($conn->connect_error) {
    die("DB 연결 실패: " . $conn->connect_error);
}
?>
```

MySQL 데이터베이스와 PHP 파일을 연결하여 데이터를 조회하거나 수정할 수 있도록 합니다. \$conn은 데이터 베이스 연결 객체로, 이후 SQL 쿼리를 실행할 때 사용됩니다.

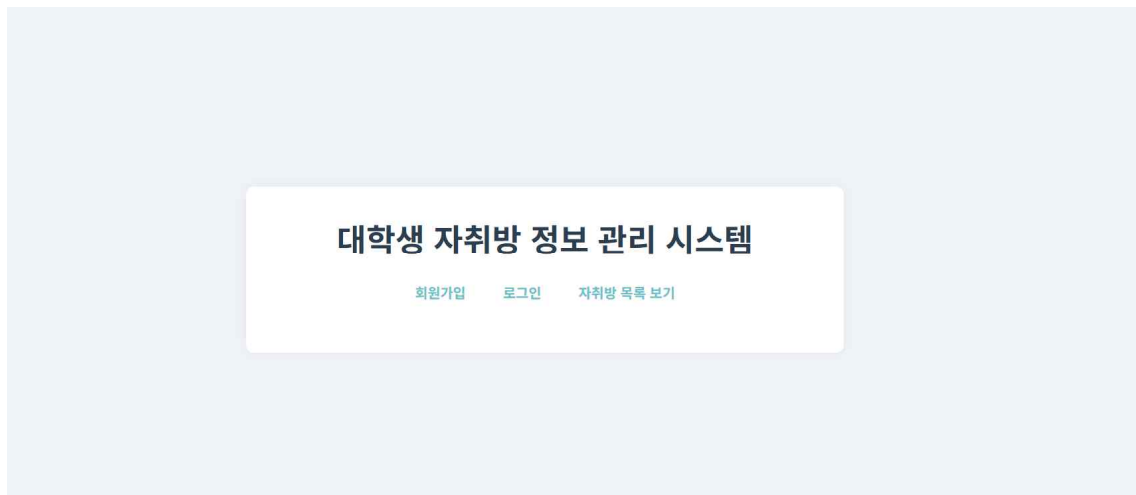
new mysqli(\$host, \$user, \$pass, \$db)는 PHP의 mysqli 확장 모듈을 사용하여 MySQL 데이터베이스에 연결합니다. \$host는 MySQL 서버 주소인 localhost로 설정하고, \$user는 MySQL 사용자 이름, \$pass는 MySQL 비밀번호, \$db는 데이터베이스 이름입니다.

5-2. 주요 기능 설명 및 주요 코드



위 그림은 대략적인 php 파일의 구현 설계입니다.

- main.php



이 페이지에서는 회원 가입과 로그인 페이지로 이동이 가능하고 자취방 목록은 로그인 없이 볼 수 있습니다.

- join_user.php

<php>

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {  
    $name      = trim(string: $_POST['name'] ?? '');  
    $phone     = trim(string: $_POST['phone'] ?? '');  
    $email     = trim(string: $_POST['email'] ?? '');  
    $password  = $_POST['password'] ?? '';  
    $user_type = 'user';  
}
```



```

if (empty($name) || empty($phone) || empty($email) || empty($password)) {
    $error = '모든 필드를 입력해주세요.';
} else {
    $checkStmt = $conn->prepare(query: "SELECT user_id FROM User WHERE email = ?");
    $checkStmt->bind_param(types: "s", var: &$email);
    $checkStmt->execute();
    $result = $checkStmt->get_result();

    if ($result && $result->num_rows > 0) {
        $error = '이미 등록된 이메일입니다.';
    } elseif (!preg_match(pattern: '/^(?=[A-Za-z])(?=[\d])(?=[@!%*#?&])[A-Za-z\d@$!%*#?&]{8,16}$/', subject: $password)) {
        $error = '비밀번호 입력 방식을 다시 확인해주세요.';
    } else {
        $hashedPassword = password_hash(password: $password, algo: PASSWORD_DEFAULT);

        $insertStmt = $conn->prepare(
            query: "INSERT INTO User (name, phone, email, password, user_type, created_at)
            VALUES (?, ?, ?, ?, ?, NOW())"
        );
        $insertStmt->bind_param(types: "sssss", var: &$name, vars: &$phone, $email, $hashedPassword, $user_type);

        if ($insertStmt->execute()) {
            $insertStmt->close();
            header(header: 'Location: login.php');
            exit;
        } else {
            $error = '회원가입 실패: ' . $insertStmt->error;
        }
    }
}
$checkStmt->close();
}

```

<html>

```

<form method="post" action="">
    <input type="text" name="name" placeholder="이름을 입력하세요" required><br>
    <input type="email" name="email" placeholder="이메일을 입력하세요" required><br>
    <input type="text" name="phone" placeholder="전화번호를 입력하세요" required><br>
    <input type="password" name="password" placeholder="비밀번호를 입력하세요" required><br>
    <small class="password-info">* 비밀번호는 영문, 숫자, 특수문자를 포함한 8~16자여야 합니다.</small><br>
    <button type="submit">회원가입</button>
</form>

```

\$_POST['action'] 값이 'register'인지 확인하여, 폼에서 action 필드의 값이 'register'일 경우에만 그 블록의 코드가 실행되도록 합니다. 이 값은 일반적으로 회원가입 폼에서 hidden input 필드로 설정하여 구분합니다.

- \$_POST['name']: 사용자가 입력한 이름
- \$_POST['phone']: 사용자가 입력한 전화번호
- \$_POST['email']: 사용자가 입력한 이메일
- \$_POST['password']: 사용자가 입력한 비밀번호
- 비밀번호 해시화: password_hash() 함수를 사용

사용자가 입력한 이름, 전화번호, 이메일, 비밀번호가 모두 입력되었는지 확인합니다. 만약 하나라도 입력이 되어있지 않다면, 오류 메시지를 출력합니다.

사용자가 입력한 이메일이 이미 데이터베이스에 등록되어 있는지 확인하기 위해 SELECT를 실행합니다. 입력한 이메일이 이미 존재하면 오류 메시지를 출력합니다.

사용자가 입력한 비밀번호가 영문, 숫자, 특수문자를 포함한 8~16자 형식인지 검사합니다. 형식에 맞지 않으면 오류 메시지를 출력합니다.

비밀번호는 password_hash() 함수를 사용하여 해시 처리 후, 사용자 정보를 데이터베이스에 insert해 줍니다.

- login.php

관리자 로그인용 [로그인](#)을 클릭하세요.

로그인

이메일을 입력하세요

비밀번호를 입력하세요

로그인

메인으로 돌아가기

이직 직장이 아니신가요? [회원가입](#)
비밀번호를 잊으셨나요? [비밀번호 재설정](#)

<php>

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $email = trim(string: $_POST['email'] ?? '');
    $password = $_POST['password'] ?? '';

    if (empty($email) || empty($password)) {
        $error = '이메일과 비밀번호를 모두 입력해주세요.';
    } else {
        $stmt = $conn->prepare(query: "SELECT * FROM User WHERE email = ?");
        $stmt->bind_param(types: "s", var: &$email);
        $stmt->execute();
        $result = $stmt->get_result();

        if ($result->num_rows === 1) {
            $User = $result->fetch_assoc();

            // 관리자는 로그인 차단
            if ($User['user_type'] === 'admin') {
                $error = '관리자는 이 페이지에서 로그인할 수 없습니다.';
            }
            // 일반 사용자만 로그인 허용
            else if ($User['password'] === $password || password_verify(password: $password, hash: $User['password'])) {
                $_SESSION['user_id'] = $User['user_id'];
                $_SESSION['user_name'] = $User['name'];
                $_SESSION['user_type'] = $User['user_type'];

                header(header: 'Location: main.php');
                exit;
            } else {
                $error = '비밀번호가 일치하지 않습니다.';
            }
        } else {
            $error = '해당 이메일로 등록된 사용자가 없습니다.';
        }
        $stmt->close();
    }
}
```

```
<html>
<form method="post" action="">
  <input type="email" name="email" placeholder="이메일을 입력하세요" required><br>
  <input type="password" name="password" placeholder="비밀번호를 입력하세요" required><br>
  <button type="submit">로그인</button>
</form>
```

로그인페이지에서 로그인 시 \$_POST로 받은 이메일과 비밀번호를 trim()을 사용하여 공백을 제거한 후 변수에 저장하고, 필수 입력 값이 비어있지 않은지 확인합니다.
 입력된 이메일을 사용하여 데이터베이스에서 해당 이메일을 가진 사용자를 조회해줍니다.
 사용자가 입력한 비밀번호와 데이터베이스에 저장된 비밀번호를 비교합니다. 비밀번호가 일치하면 세션에 사용자 정보를 저장하고 로그인 한 메인 페이지로 이동하게 됩니다.
 비밀번호 불일치나 사용자가 없을 경우 적절한 오류 메시지를 출력합니다.

```
<div class="info-text">
  <p>관리자 로그인은 <a href="admin_passcheck.php">여기</a>를 클릭하세요.</p>
</div>
```

관리자 로그인은 여기를 클릭하면 관리자 로그인 창으로 연결합니다.

```
<p>아직 회원이 아니신가요? <a href="join_user.php">회원가입</a></p>
<p>비밀번호를 잊으셨나요? <a href="reset_password.php">비밀번호 재설정</a></p>
```

회원이 아니라면 회원가입 페이지로, 비밀번호를 잊었다면 비밀번호 재설정 페이지로 이동한다.

비밀번호 재설정

등록된 이메일을 입력하세요 재설정 링크 받기

[로그인 페이지로 돌아가기](#)

[비밀번호 재설정 페이지]

- main.php



회원가입 후 로그인에 성공하게 되면 main.php가 위와 같은 화면으로 바뀌게 됩니다.
 사용자는 메인 페이지에서 마이페이지, 자취방 목록을 이용할 수 있고 필요에 따라 로그아웃을 할 수 있습니다.

- mypage.php



마이페이지로 이동하면 다음과 같은 화면이 나옵니다. 현재 user1이라는 이름의 사용자가 로그인 한 상태입니다. 마이페이지에서는 내 북마크 목록, 내 좋아요 목록, 내가 쓴 후기를 볼 수 있고 자취방 전체 목록, 로그아웃, 메인으로 돌아가기의 버튼을 만들었습니다. 또한 회원 탈퇴가 가능합니다.

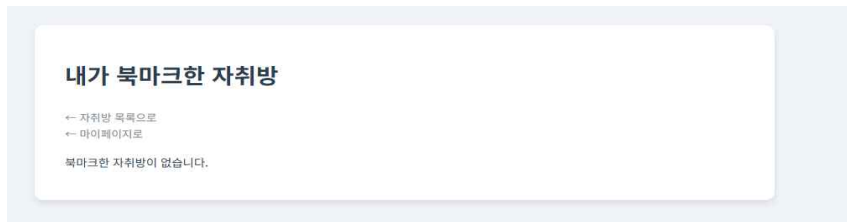
```
<php>
session_start();
if (!isset($_SESSION['user_id'])) {
    header(header: 'Location: login.php');
    exit;
}

<html>

<ul>
    <a href="my_bookmarks.php"><button>내 북마크 목록</button></a>
    <a href="my_likes.php"><button>내 좋아요 목록</button></a>
    <a href="my_reviews.php"><button>내가 쓴 후기</button></a>
    <a href="rooms_list.php"><button>자취방 전체 목록</button></a>
    <a href="logout.php"><button>로그아웃</button></a>
    <a href="main.php"><button>메인으로 돌아가기</button></a>
    <a href="withdraw.php" style="color:red;"><button>회원 탈퇴</button></a>
</ul>
```

session_start()로 세션을 시작하고, 로그인된 사용자만 접근할 수 있도록 합니다. 사용자가 로그인하지 않은 상태에서 이 페이지에 접근하려고 하면, login 페이지로 이동하게 됩니다.

- my_bookmarks.php



[북마크 한 자취방이 없을 때]



[사용자가 자취방에 북마크를 하고 마이페이지에서 확인 했을 때]
마이 페이지에서 “내 북마크 목록”을 클릭하면 다음 페이지로 넘어 갑니다.

<php>

```
$sql = "SELECT r.room_id, r.room_name, r.location, r.price, r.size, r.created_at
        FROM Bookmark b
        JOIN Room r ON b.room_id = r.room_id
        WHERE b.user_id = ?
        ORDER BY b.created_at DESC";

$stmt = $conn->prepare(query: $sql);
$stmt->bind_param(types: "i", var: &$user_id);
$stmt->execute();
$result = $stmt->get_result();
?>
```

<html>

```
<?php if ($result->num_rows > 0): ?>
    <?php while ($row = $result->fetch_assoc()): ?>
        <div class="room-box">
            <p><strong><?php echo htmlspecialchars(string: $row['room_name']); ?></strong></p>
            <p>위치: <?php echo htmlspecialchars(string: $row['location']); ?></p>
            <p>가격: <?php echo number_format(num: $row['price']); ?> 원</p>
            <p>평수: <?php echo htmlspecialchars(string: $row['size']); ?> 평</p>
            <p>등록일: <?php echo htmlspecialchars(string: $row['created_at']); ?></p>
            <p><a href="room_detail.php?room_id=<?php echo $row['room_id']; ?>">상세 보기</a></p>
        </div>
    <?php endwhile; ?>
<?php else: ?>
    <p>북마크한 자취방이 없습니다.</p>
<?php endif; ?>
```

Bookmark와 Room테이블을 JOIN하여 사용자가 북마크한 자취방 목록을 가져옵니다. 해당 사용자를 기준으로 사용자가 북마크한 자취방을 찾고, 생성일 내림차순으로 정렬합니다. 사용자가 북마크한 자취방 목록을 출력하며, 간단한 자취방 정보를 제공하고 자세한 정보를 원한다면 상세보기가 가능합니다. 또한 '← 자취방 목록으로', '← 마이페이지로' 버튼을 클릭해 각각의 페이지로 넘어갈 수 있습니다.

- my_likes.php

내가 좋아요한 자취방

← 자취방 목록으로
← 마이페이지로

원룸 20

위치: 천안시 동남구 청룡동 56
가격: 350,000 원
평수: 8 평
등록일: 2025-06-07 16:32:35
[상세 보기](#)

원룸 17

위치: 천안시 서북구 두정동 76
가격: 470,000 원
평수: 10 평
등록일: 2025-06-07 16:32:35
[상세 보기](#)

원룸 18

위치: 천안시 동남구 봉명동 78
가격: 360,000 원
평수: 8 평
등록일: 2025-06-07 16:32:35
[상세 보기](#)

[사용자가 자취방에 좋아요를 누르고 마이페이지에서 확인 했을 때]

<php>

```
$sql = "SELECT r.room_id, r.room_name, r.location, r.price, r.size, r.created_at
FROM Likes l
JOIN Room r ON l.room_id = r.room_id
WHERE l.user_id = ?
ORDER BY l.created_at DESC";

$stmt = $conn->prepare(query: $sql);
$stmt->bind_param(types: "i", var: &$user_id);
$stmt->execute();
$result = $stmt->get_result();
?>
```

<html>

```
<?php if ($result->num_rows > 0): ?>
    <?php while ($row = $result->fetch_assoc()): ?>
        <div class="room-box">
            <p><strong><?php echo htmlspecialchars(string: $row['room_name']); ?></strong></p>
            <p>위치: <?php echo htmlspecialchars(string: $row['location']); ?></p>
            <p>가격: <?php echo number_format(num: $row['price']); ?> 원</p>
            <p>평수: <?php echo htmlspecialchars(string: $row['size']); ?> 평</p>
            <p>등록일: <?php echo htmlspecialchars(string: $row['created_at']); ?></p>
            <p><a href="room_detail.php?room_id=<?php echo $row['room_id']; ?>">상세 보기</a></p>
        </div>
    <?php endwhile; ?>
<?php else: ?>
    <p>좋아요한 자취방이 없습니다.</p>
<?php endif; ?>
```

사용자가 좋아요 한 자취방을 조회하기 위해 **Likes**와 **Room** 테이블을 **JOIN**하여 데이터를 가져옵니다. 사용자를 기준으로 사용자가 좋아요 한 자취방을 찾고, 생성일 내림차순 정렬합니다. 좋아요한 자취방 목록을 출력합니다. 이름, 위치, 가격, 평수, 등록일 등의 정보를 보여주고, 상세 보기 링크를 통해 사용자가 자취방의 상세 정보를 볼 수 있도록 합니다.

- my_reviews.php

내가 작성한 후기

← 자취방 목록으로
← 마이페이지로

원룸 1

평점: 5/5

작성일: 2025-06-11 22:10:21

내용:

별이 잘 들고 이동하기 좋은 위치 같아요

[방 보러가기](#)

마이페이지에서 “내가 쓴 후기”를 클릭하면 다음과 같은 화면으로 넘어가게 됩니다.

<php>

```
$sql = "SELECT r.room_name, rv.rating, rv.content, rv.created_at, rv.room_id
FROM Review rv
JOIN Room r ON rv.room_id = r.room_id
WHERE rv.user_id = ?
ORDER BY rv.created_at DESC";

$stmt = $conn->prepare(query: $sql);
$stmt->bind_param(types: "i", var: &$user_id);
$stmt->execute();
$result = $stmt->get_result();
?>
```

<html>

```
<?php if ($result->num_rows > 0): ?>
    <?php while ($row = $result->fetch_assoc()): ?>
        <div class="room-box">
            <p><strong><?php echo htmlspecialchars(string: $row['room_name']); ?></strong></p>
            <p>평점: <?php echo $row['rating']; ?>/5</p>
            <p>작성일: <?php echo $row['created_at']; ?></p>
            <p>내용:</p>
            <p><?php echo nl2br(string: htmlspecialchars(string: $row['content'])); ?></p>
            <p><a href="room_detail.php?room_id=<?php echo $row['room_id']; ?>">방 보러가기</a></p>
        </div>
    <?php endwhile; ?>
<?php else: ?>
    <p>작성한 후기가 없습니다.</p>
<?php endif; ?>
```

사용자가 작성한 후기를 조회하기 위해 Review와 Room 테이블을 JOIN하여 데이터를 가져옵니다. 사용자를 기준으로 사용자가 작성한 자취방의 리뷰를 찾고 리뷰 작성일을 내림차순으로 정렬합니다. 그 후 사용자가 작성한 각 후기에 대한 정보를 출력합니다. 각 후기는 방 이름, 평점,작성일, 내용과 함께 보여지며, 상세 보기 링크를 제공합니다.

- rooms_list.php

The screenshot shows a web application titled "자취방 목록" (Room List). It features a search section with the following elements:

- 지역(구, 동 단위): [Text Input]
- 최소 가격: 0 [Text Input] 최대 가격: 10000000 [Text Input]
- 최소 평수: [Text Input] 평형: --선택-- [Dropdown]
- Buttons: 검색 (Search), 메인으로 돌아가기 (Return to Main)

Below the search section, there are three room listings, each with a title, location, price, area, and buttons for "상세 보기" (View Details) and "리뷰 수정하기" (Edit Review):

- 원룸 1**
위치: 천안시 서북구 두정동 37
가격: 500,000 원
평수: 13 평
Buttons: 상세 보기, 리뷰 수정하기
- 원룸 2**
위치: 천안시 동남구 신방동 48
가격: 450,000 원
평수: 13 평
Buttons: 상세 보기, 리뷰 작성하기
- 원룸 3**
위치: 천안시 동남구 신방동 77
가격: 440,000 원
평수: 12 평
Buttons: 상세 보기, 리뷰 작성하기

마이페이지에서 "자취방 전체 목록"을 클릭하면 다음과 같은 화면으로 넘어가게 됩니다.

<php>

```
// 정렬 기준에 따라 SQL 시작
if ($sort === 'likes') {
    $sql = "SELECT Room.*, (SELECT COUNT(*) FROM Likes WHERE Likes.room_id = Room.room_id) AS like_count
    FROM Room
    WHERE price BETWEEN ? AND ?";
} else {
    $sql = "SELECT * FROM Room WHERE price BETWEEN ? AND ?";
}

// 필터 조건
if (!empty($location)) {
    $sql .= " AND location LIKE ?";
    $params[] = "%$location%";
    $types .= "s";
}
if (!empty($size)) {
    $sql .= " AND size >= ?";
    $params[] = $size;
    $types .= "d";
}
```

가격 범위, 지역, 크기, 정렬 기준을 바탕으로 자취방을 조회합니다.

사용자가 입력한 지역과 최소 평에 따라 자취방을 필터링합니다. location이 입력되면, LIKE 연산자를 사용하여 해당되는 지역만 검색합니다. size가 입력되면, 해당 평수 이상인 자취방만 검색하게 됩니다.

<html>

```
<form method="get" action="">
    <div class="form-group">
        <label for="location">지역(구, 동 단위):</label>
        <input type="text" id="location" name="location" value="<?php echo htmlspecialchars(string: $location); ?>">
    </div>
    <div class="form-group">
        <label for="min_price">최소 가격:</label>
        <input type="number" id="min_price" name="min_price" value="<?php echo htmlspecialchars(string: $min_price); ?>">
        <label for="max_price">최대 가격:</label>
        <input type="number" id="max_price" name="max_price" value="<?php echo htmlspecialchars(string: $max_price); ?>">
    </div>
    <div class="form-group">
        <label for="size">최소 평수:</label>
        <input type="number" id="size" name="size" step="1" value="<?php echo htmlspecialchars(string: $size); ?>">
    </div>
    <div class="form-group">
        <label for="sort">정렬:</label>
        <select id="sort" name="sort">
            <option value="">-- 선택 --</option>
            <option value="likes" <?=$sort === 'likes' ? 'selected' : '' ?>>좋아요 많은 순</option>
            <option value="recent" <?=$sort === 'recent' ? 'selected' : '' ?>>최신순</option>
        </select>
    </div>
    <button type="submit">검색</button>
    <a href="main.php"><button type="button">메인으로 돌아가기</button></a>
</form>
```

자취방을 가격, 지역, 크기, 정렬 기준으로 필터링 할 수 있는 검색 폼을 제공합니다.


```

<div class="room-info">
    <strong><?php echo htmlspecialchars(string: $row['room_name']); ?></strong>
    위치: <?php echo htmlspecialchars(string: $row['location']); ?><br>
    가격: <?php echo number_format(num: $row['price']); ?> 원<br>
    평수: <?php echo htmlspecialchars(string: $row['size']); ?> 평<br><br>
    <a href="room_detail.php?room_id=<?php echo $row['room_id']; ?>">상세 보기</a>
</div>

<button class="icon-button like" data-room-id="<?=$row['room_id'] ?>" data-liked="<?=$is_liked ? '1' : '0' ?>">
    <?=$is_liked ? '♥' : '♡' ?>
</button>

<button class="icon-button bookmark" data-room-id="<?=$row['room_id'] ?>" data-bookmarked="<?=$is_bookmarked ? '1' : '0' ?>">
    <?=$is_bookmarked ? '🔖' : '🔖' ?>
</button>

<a href="<?=$has_review ? 'review_edit.php' : 'review_write.php' ?>?room_id=<?=$row['room_id'] ?>" class="btn-review">
    <?=$has_review ? '리뷰 수정하기' : '리뷰 작성하기' ?>
</a>

```

조회된 자취방 정보를 화면에 출력해줍니다. 각 자취방에 대해 상세보기 링크와 좋아요 북마크 버튼을 제공해줍니다. 또한 사용자가 리뷰를 작성하거나 수정할 수 있도록 리뷰 작성하기, 리뷰 수정하기 링크를 제공해줍니다.

좋아요와 북마크 버튼은 버튼을 클릭할 때마다 해당 상태가 바뀌며, 이를 like_toggle.php와 bookmark_toggle.php를 통해 서버에 전송하여 상태를 업데이트합니다. 사용자가 이미 버튼을 클릭한 경우에는 1, 그렇지 않으면 0으로 설정되어 어떤 자취방에 해당 버튼을 클릭했는지 구분합니다.

자취방 목록

지역(구, 동 단위):

최소 가격:

최대 가격:

최소 평수:

정렬: 선택

검색

메인으로 돌아가기

원룸 1

♥

🔖

위치: 천안시 서북구 두정동 37

가격: 500,000 원

평수: 13 평

상세 보기

리뷰 수정하기

원룸 6

♡

🔖

위치: 천안시 서북구 쌍용동 8

가격: 460,000 원

평수: 14 평

상세 보기

리뷰 작성하기

원룸 10

♡

🔖

위치: 천안시 서북구 성성동 7

가격: 490,000 원

평수: 12 평

상세 보기

리뷰 작성하기

[지역을 서북구로 설정 후 검색]



- 41 -

자취방 목록

지역(구, 동 단위):

최소 가격: 최대 가격:

최소 평수: 정렬:



원룸 1



위치: 천안시 서북구 두정동 37

가격: 500,000 원

평수: 13 평

[상세 보기](#)
[리뷰 수정하기](#)



원룸 10



위치: 천안시 서북구 성성동 7

가격: 490,000 원

평수: 12 평

[상세 보기](#)
[리뷰 작성하기](#)

원룸 11



위치: 천안시 서북구 두정동 22

가격: 470,000 원

평수: 10 평

[상세 보기](#)
[리뷰 작성하기](#)

[지역을 서북구로 설정하고 최소 가격을 47만원으로 설정 후 검색]

자취방 목록

지역(구, 동 단위):

최소 가격: 최대 가격:

최소 평수: 정렬:

원룸 1



위치: 천안시 서북구 두정동 37

가격: 500,000 원

평수: 13 평

[상세 보기](#)
[리뷰 수정하기](#)

원룸 25



위치: 천안시 서북구 두정동 14

가격: 510,000 원

평수: 14 평

[상세 보기](#)
[리뷰 작성하기](#)

[지역을 서북구 최소 가격을 47만원으로 설정, 최소 평수를 13평으로 설정 후 검색]

자취방 목록

지역(구, 동 단위):

최소 가격:

최대 가격:

최소 평수:

정렬:

검색

메인으로 돌아가기

원룸 2

위치: 천안시 동남구 신방동 48
가격: 450,000 원
평수: 13 평

상세 보기

리뷰 작성하기

원룸 3

위치: 천안시 동남구 신방동 77
가격: 440,000 원
평수: 12 평

상세 보기

리뷰 작성하기

원룸 4

위치: 천안시 동남구 원성동 56
가격: 390,000 원
평수: 12 평

상세 보기

리뷰 작성하기

원룸 12


위치: 천안시 동남구 원성동 80
가격: 360,000 원
평수: 9 평

상세 보기

리뷰 작성하기

[지역을 동남구로 설정 후 좋아요 많은 순으로 검색]

원룸 2 상세 정보



위치: 천안시 동남구 신방동 48
 가격: 450,000 원
 평수: 13 평
 등록일: 2025-06-07 16:32:35
 설명: 신방동 지역의 원룸입니다. 조용하고 편리한 곳입니다.

제공 옵션

- 에어컨
- 냉장고
- 세탁기
- 인터넷
- 가스레인지

좋아요 ♥ (4)
 후기
 User4 (4/5)
 2025-06-11 22:56:17
 조금 비싸긴 하지만 주변에 편의시설이 잘 구축되어 있어요

User2 (4/5)
 2025-06-11 22:12:06
 평 평수에 비해 너무 비싼것 같아요. 이동하기는 좋은 위치 같습니다.

[← 목록으로 돌아가기](#)

원룸 3 상세 정보



위치: 천안시 동남구 신방동 77
 가격: 440,000 원
 평수: 12 평
 등록일: 2025-06-07 16:32:35
 설명: 신방동 지역의 원룸입니다. 조용하고 편리한 곳입니다.

제공 옵션

- 에어컨
- 냉장고
- 세탁기
- 인터넷
- 가스레인지

좋아요 ♥ (3)
 후기
 아직 후기가 없습니다.

[← 목록으로 돌아가기](#)

원룸 4 상세 정보



위치: 천안시 동남구 원성동 56
 가격: 390,000 원
 평수: 12 평
 등록일: 2025-06-07 16:32:35
 설명: 원성동 지역의 원룸입니다. 주거 환경이 쾌적한 곳입니다.

제공 옵션

- 에어컨
- 냉장고
- 세탁기
- 인터넷
- 가스레인지

좋아요 ♥ (2)
 후기
 아직 후기가 없습니다.

[← 목록으로 돌아가기](#)

원룸 12 상세 정보



위치: 천안시 동남구 원성동 80
 가격: 360,000 원
 평수: 9 평
 등록일: 2025-06-07 16:32:35
 설명: 원성동 지역의 원룸입니다. 주거 환경이 쾌적한 곳입니다.

제공 옵션

- 에어컨
- 냉장고
- 세탁기
- 인터넷
- 가스레인지

좋아요 ♥ (2)
 후기
 아직 후기가 없습니다.

[← 목록으로 돌아가기](#)

해당 상세 정보를 보게 되면 차례대로 원룸 2, 원룸 3, 원룸 4, 원룸 12 순으로 좋아요가 많은 것을 볼 수 있습니다. 만약 좋아요 수가 같다면 자취방의 이름 순으로 정렬이 됩니다.

자취방 목록

지역(구, 동 단위):

최소 가격: 최대 가격:

최소 평수: 정렬: 최신순 ↓ 검색 메인으로 돌아가기

소망빌

위치: 아산시 신창면 순천향로 16-22
 가격: 350,000 원
 평수: 12 평

상세 보기

리뷰 작성하기

학생사 1

위치: 아산시 신창면 순천향로 22
 가격: 200,000 원
 평수: 12 평

상세 보기

리뷰 작성하기

[최신순 정렬 설정 후 검색]

<php>

```
// 정렬 기준
if ($sort === 'likes') {
    $sql .= " ORDER BY like_count DESC";
} elseif ($sort === 'recent') {
    $sql .= " ORDER BY created_at DESC";
}
```

사용자가 선택한 정렬 기준에 따라 자취방을 정렬합니다. likes를 선택하면, 좋아요 수를 기준으로 자취방을 내림차순 정렬합니다. recent를 선택하면, 최신 등록일을 기준으로 자취방을 정렬합니다. 이를 통해 사용자들이 가장 관심있어 하는 방과, 가장 신축 방이 어떤 방인지 알 수 있습니다.

```
<?php
$check_like = $conn->prepare(query: "SELECT 1 FROM Likes WHERE user_id = ? AND room_id = ?");
$check_like->bind_param(types: "ii", var: &$_SESSION['user_id'], vars: &$row['room_id']);
$check_like->execute();
$is_liked = $check_like->get_result()->num_rows > 0;
$check_like->close();
```

사용자가 해당 자취방에 좋아요를 눌렀는지 확인한 후, 버튼의 상태를 업데이트 합니다.

```
<?php
$check_bookmark = $conn->prepare(query: "SELECT 1 FROM Bookmark WHERE user_id = ? AND room_id = ?");
$check_bookmark->bind_param(types: "ii", var: &$_SESSION['user_id'], vars: &$row['room_id']);
$check_bookmark->execute();
$is_bookmarked = $check_bookmark->get_result()->num_rows > 0;
$check_bookmark->close();
```

사용자가 해당 자취방에 북마크를 했는지 확인한 후, 버튼의 상태를 업데이트 합니다.

```
$check_review = $conn->prepare(query: "SELECT 1 FROM Review WHERE user_id = ? AND room_id = ?");
$check_review->bind_param(types: "ii", var: &$_SESSION['user_id'], vars: &$row['room_id']);
$check_review->execute();
$has_review = $check_review->get_result()->num_rows > 0;
$check_review->close();
```

사용자가 해당 자취방에 대해 리뷰를 작성했는지 확인한 후, 리뷰 작성/수정 링크를 동적으로 표시해줍니다.

- room_detail.php

원룸 2 상세 정보



위치: 천안시 동남구 신방동 48

가격: 450,000 원

평수: 13 평

등록일: 2025-06-07 16:32:35

설명: 신방동 지역의 원룸입니다. 조용하고 편리한 곳입니다.

제공 옵션

- 에어컨
- 냉장고
- 세탁기
- 인터넷
- 가스레인지

좋아요 ♥ (4)

후기

User4 (4/5)

2025-06-11 22:56:17

조금 비싸긴 하지만 주변에 편의시설이 잘 구축되어 있어요

User2 (4/5)

2025-06-11 22:12:06

방 평수에 비해 너무 비싼것 같아요. 이동하기는 좋은 위치 같습니다.

[← 목록으로 돌아가기](#)

<php>

// 방 정보 조회

```
$sql = "SELECT * FROM Room WHERE room_id = ?";  
$stmt = $conn->prepare(query: $sql);  
$stmt->bind_param(types: "i", var: &$room_id);  
$stmt->execute();  
$result = $stmt->get_result();  
if ($result->num_rows === 0) die('해당 방을 찾을 수 없습니다.');
```

Room 테이블에서 해당 room_id에 맞는 방 정보를 조회합니다.

// 옵션 조회

```
$sql = "SELECT f.feature_name FROM Feature f JOIN RoomFeature rf ON f.feature_id = rf.feature_id WHERE rf.room_id = ?";  
$stmt = $conn->prepare(query: $sql);  
$stmt->bind_param(types: "i", var: &$room_id);  
$stmt->execute();  
$options = $stmt->get_result();
```

// 후기 조회

```
$sql = "SELECT u.name, r.rating, r.content, r.created_at FROM Review r JOIN User u ON r.user_id = u.user_id WHERE r.room_id = ? ORDER BY r.created_at DESC";  
$review_stmt = $conn->prepare(query: $sql);  
$review_stmt->bind_param(types: "i", var: &$room_id);  
$review_stmt->execute();  
$reviews = $review_stmt->get_result();
```

// 좋아요 수 조회

```
$sql = "SELECT COUNT(*) as like_count FROM Likes WHERE room_id = ?";  
$stmt = $conn->prepare(query: $sql);  
$stmt->bind_param(types: "i", var: &$room_id);  
$stmt->execute();  
$like_count = $stmt->get_result()->fetch_assoc()['like_count'];  
$stmt->close();  
?>
```


- **RoomFeature와 Feature 테이블을 JOIN**하여 해당 자취방의 제공 옵션을 조회합니다. 각 자취방의 옵션(feature_name)을 나열합니다.
- **Review와 User 테이블을 JOIN**하여 해당 자취방에 대한 후기를 조회합니다. 최신 후기부터 순차적으로 출력합니다.
- Likes 테이블을 사용하여 해당 자취방의 좋아요 수를 조회합니다.

<html>

```
<?php if (empty($Room['image_path'])): ?>
    <div class="room-image">
        <!-- 이미지 경로를 동적으로 설정 (상대 경로) -->
        위치: <?= htmlspecialchars(string: $Room['location']) ?></p>
<p>가격: <?= number_format(num: $Room['price']) ?> 원</p>
<p>평수: <?= htmlspecialchars(string: $Room['size']) ?> 평</p>
<p>등록일: <?= htmlspecialchars(string: $Room['created_at']) ?></p>
<p>설명: <?= nl2br(string: htmlspecialchars(string: $Room['description'])) ?></p>
```

<h3>제공 옵션</h3>

```
<ul>
    <?php
    if ($options->num_rows > 0):
        while ($opt = $options->fetch_assoc()):
            ?>
            <li><?= htmlspecialchars(string: $opt['feature_name']) ?></li>
        <?php endwhile; ?>
    <?php else: ?>
        <li>제공 옵션이 없습니다.</li>
    <?php endif; ?>
</ul>
```

<h3>좋아요 수 (<?= \$like_count ?>)</h3>

<h3>후기</h3>

```
<?php if ($reviews->num_rows > 0): ?>
    <?php while ($r = $reviews->fetch_assoc()): ?>
        <div>
            <strong><?= htmlspecialchars(string: $r['name']) ?></strong> (<?= $r['rating'] ?>/5)<br>
            <small><?= $r['created_at'] ?></small><br>
            <p><?= nl2br(string: htmlspecialchars(string: $r['content'])) ?></p>
            <hr>
        </div>
    <?php endwhile; ?>
<?php else: ?>
    <p>아직 후기가 없습니다.</p>
<?php endif; ?>
```

자취방에 이미지가 존재할 경우, 해당 이미지를 출력합니다. 이미지 경로는 데이터베이스에서 가져온 image_path 값을 사용합니다.

자취방의 이름, 위치, 가격, 평수, 자취방 등록 일시를 보여주고 Feature 테이블에서 가져온 자취방의 옵션을 보여줍니다. 옵션은 상세 페이지에서 목록으로 표시됩니다. 해당 자취방의 좋아요 수를 출력해주고 등록된 후기가 있다면 작성자 이름, 평점, 작성일, 내용을 포함하여 출력합니다.

- review_edit.php

리뷰 수정

평점 (1~5):

내용:

별이 잘 들고 이동하기 좋은 위치 같아요

리뷰 수정

리뷰 삭제

[← 방 상세 페이지로 돌아가기](#)

[← 자취방 목록으로 돌아가기](#)

<php>

```
$stmt = $conn->prepare(query: "SELECT rating, content FROM Review WHERE user_id = ? AND room_id = ?")
$stmt->bind_param(types: "ii", var: &$user_id, vars: &$room_id);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows === 0) {
    die('작성한 리뷰가 없습니다.');
```


Review 테이블에서 현재 로그인한 사용자가 작성한 해당 자취방에 대한 작성한 리뷰를 조회합니다. 사용자가 해당 자취방에 대한 리뷰를 작성하지 않은 경우, "작성한 리뷰가 없습니다." 메시지를 출력하고 종료합니다.

<html>


```
<form method="post" action="review_update.php">
    <input type="hidden" name="room_id" value="<?= $room_id ?>">
    평점 (1~5):
    <input type="number" name="rating" min="1" max="5" value="<?= $review['rating'] ?>" required><br>
    내용:<br>
    <textarea name="content" rows="5" required><?= htmlspecialchars(string: $review['content']) ?></textarea><br>
    <button type="submit">리뷰 수정</button>
</form>
```

사용자가 기존에 작성한 평점과 내용을 수정할 수 있도록 합니다. 기존 리뷰의 평점과 내용은 \$review['rating']과 htmlspecialchars(\$review['content'])를 통해 동적으로 채워집니다. 리뷰 수정 버튼을 누르면 review_update.php로 폼을 제출하여 리뷰를 업데이트 합니다.

마이페이지에서 로그아웃을 클릭하면 로그인 화면으로 넘어가게 됩니다.

로그인 화면은 흰색 배경에 파란색 테두리입니다. 중앙에는 '로그인'이라는 제목이 있습니다. 그 아래에는 '이메일을 입력하세요'와 '비밀번호를 입력하세요'라는 두 개의 입력란이 있습니다. 입력란 아래에는 '로그인' 버튼이 있습니다. 버튼 아래에는 '메인으로 돌아가기' 버튼이 있습니다. 화면 하단에는 '아직 회원이 아니신가요? [회원가입](#)'과 '비밀번호를 잊으셨나요? [비밀번호 재설정](#)'이라는 링크가 있습니다.

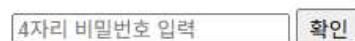
관리자모드

관리자 로그인 안내 화면은 파란색 배경에 흰색 테두리입니다. 중앙에는 '관리자 로그인은 [여기](#)를 클릭하세요.'라는 텍스트가 있습니다.

[여기](#)를 클릭하면 다음과 같은 화면으로 넘어가게 됩니다.

- admin_passcheck.php

관리자 인증

관리자 인증 폼은 흰색 배경에 파란색 테두리입니다. 중앙에는 '4자리 비밀번호 입력'이라는 텍스트와 '확인' 버튼이 있습니다.

<php>

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $input_pass = $_POST['admin_pass'] ?? '';

    // 4자리 비밀번호 설정 (예: 1234)
    $correct_pass = '1234';

    if ($input_pass === $correct_pass) {
        $_SESSION['admin_pass_verified'] = true;
        header(header: 'Location: admin_login.php');
        exit;
    } else {
        $pass_error = '비밀번호가 틀렸습니다.';
    }
}
```

사용자가 제출한 비밀번호는 \$_POST['admin_pass']로 받아옵니다. 4자리 비밀번호를 미리 설정해두고 사용자가 입력한 비밀번호와 비교합니다. 사용자가 입력한 비밀번호와 일치하면

SESSION 변수를 true로 설정하고, 관리자 로그인 페이지로 이동합니다. 비밀번호가 틀리면 오류 메시지를 \$pass_error 출력해줍니다.

```
<html>
<?php if ($pass_error): ?>
    <p style="color:red;"><?= htmlspecialchars(string: $pass_error) ?></p>
<?php endif; ?>
<form method="post" action="">
    <input type="password" name="admin_pass" maxlength="4" placeholder="4자리 비밀번호 입력" required autofocus>
    <button type="submit">확인</button>
</form>
```

사용자가 4자리 비밀번호를 입력할 수 있는 입력필드를 제공합니다. 길이를 4자리로 제한합니다.

- admin_login.php

The image shows a web form titled "관리자 로그인" (Admin Login). It contains two input fields: "이메일을 입력하세요" (Enter email) and "비밀번호를 입력하세요" (Enter password). Below these fields is a teal "로그인" (Login) button. At the bottom, there are two teal buttons: "회원가입 하러가기" (Go to sign up) and "메인으로 돌아가기" (Go back to main).

```
<php>
if (!isset($_SESSION['admin_pass_verified']) || $_SESSION['admin_pass_verified'] !== true) {
    header(header: 'Location: admin_passcheck.php');
    exit;
}

$email = trim(string: $_POST['email'] ?? '');
$password = $_POST['password'] ?? '';

if (empty($email) || empty($password)) {
    $error = '이메일과 비밀번호를 모두 입력해주세요.';
} elseif (!preg_match(pattern: '/^(?=.*[A-Za-z])(?=.*\d)(?=.*[@$!%*#?&])[A-Za-z\d@$!%*#?&]{8,16}$/', subject: $password)) {
    $error = '비밀번호는 영문, 숫자, 특수문자를 포함한 8~16자여야 합니다.';
}
```

사용자가 관리자 인증을 마친 후에만 이 페이지에 접근할 수 있도록 합니다. 인증되지 않으면 admin_passcheck 페이지로 이동합니다.

사용자가 제출한 이메일과 비밀번호를 확인하고 비어있으면 오류메시지를 출력하고 비밀번호의 형식이 맞지 않으면 형식지 맞지 않다는 오류 메시지를 출력합니다.

관리자 로그인

관리자 권한이 없는 계정입니다.

로그인

이메일을 입력하세요

비밀번호를 입력

이 입력란을 작성하세요.

[관리자가 아닌 계정으로 로그인 시도]

[필수입력란에 정보가 적혀있지 않을 때]

```
$stmt = $conn->prepare(query: "SELECT * FROM User WHERE email = ?");
$stmt->bind_param(types: "s", var: &$email);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows === 1) {
    $User = $result->fetch_assoc();

    if ($User['user_type'] !== 'admin') {
        $error = '관리자 권한이 없는 계정입니다.';
    } elseif (password_verify(password: $password, hash: $User['password'])) {
        $_SESSION['user_id'] = $User['user_id'];
        $_SESSION['user_name'] = $User['name'];
        $_SESSION['user_type'] = $User['user_type'];

        header(header: 'Location: admin_dashboard.php');
        exit;
    } else {
        $error = '비밀번호가 일치하지 않습니다.';
    }
} else {
    $error = '해당 이메일로 등록된 사용자가 없습니다.';
}
```

데이터베이스에서 사용자가 입력한 이메일을 가진 사용자를 조회하고 해당 사용자의 user_type이 admin인지 확인합니다. 만약 user_type이 admin이 아니라면 '관리자 권한이 없는 계정입니다'를 출력하고 로그인이 안되게 설정합니다.

사용자가 입력한 비밀번호와 데이터 베이스에 저장된 비밀번호를 password_verify() 함수를 사용하여 비교합니다. 비밀번호가 일치하면 세션에 사용자 정보를 저장하고 관리자 페이지로 이동합니다.

```
<html>
<form method="post" action="">
    <input type="email" name="email" placeholder="이메일을 입력하세요" required><br>
    <input type="password" name="password" placeholder="비밀번호를 입력하세요" required><br>
    <button type="submit">로그인</button>
</form>
```

사용자가 이메일과 비밀번호를 입력할 수 있는 폼을 제공합니다. 폼은 POST 방식으로 제출되고 이메일과 비밀번호 필드는 필수 항목입니다.

- admin_dashboard.php

관리자 페이지

안녕하세요, 관리자님

- [자취방 관리](#)
- [옵션 목록 관리](#)
- [후기 관리](#)
- [사용자 관리](#)
- [로그아웃](#)

관리자 페이지로 넘어오게 되면 보이는 페이지입니다.

해당 페이지에서 자취방, 옵션, 후기, 사용자를 관리 할 수 있습니다. 해당 텍스트를 클릭하면 링크로 연결된 해당 페이지로 넘어갑니다.

로그아웃을 클릭한다면 바로 로그인 되어있지 않은 main 페이지로 이동합니다.

- amdin_rooms.php

자취방 목록

자취방 추가

방 이름:

위치:

가격(원):

면적(m²):

- 소망빌 (아산시 신창면 순천향로 16-22 / 350,000원) [보기](#) [수정](#) [삭제](#)
- 학성사 1 (아산시 신창면 순천향로 22 / 200,000원) [보기](#) [수정](#) [삭제](#)
- 향실생활관 3 (아산시 신창면 순천향로 22 / 280,000원) [보기](#) [수정](#) [삭제](#)
- 향실생활관 2 (아산시 신창면 순천향로 22 / 320,000원) [보기](#) [수정](#) [삭제](#)
- 향실생활관 1 (아산시 신창면 순천향로 22 / 330,000원) [보기](#) [수정](#) [삭제](#)
- 원룸 1 (천안시 서북구 두정동 37 / 500,000원) [보기](#) [수정](#) [삭제](#)
- 원룸 2 (천안시 동남구 신방동 48 / 450,000원) [보기](#) [수정](#) [삭제](#)
- 원룸 3 (천안시 동남구 신방동 77 / 440,000원) [보기](#) [수정](#) [삭제](#)
- 원룸 4 (천안시 동남구 원성동 56 / 390,000원) [보기](#) [수정](#) [삭제](#)
- 원룸 5 (천안시 동남구 원성동 97 / 420,000원) [보기](#) [수정](#) [삭제](#)
- 원룸 6 (천안시 서북구 쌍용동 8 / 460,000원) [보기](#) [수정](#) [삭제](#)
- 원룸 7 (천안시 동남구 용곡동 20 / 360,000원) [보기](#) [수정](#) [삭제](#)
- 원룸 8 (천안시 동남구 신방동 17 / 450,000원) [보기](#) [수정](#) [삭제](#)
- 원룸 9 (천안시 동남구 원성동 35 / 390,000원) [보기](#) [수정](#) [삭제](#)
- 원룸 10 (천안시 서북구 성성동 7 / 490,000원) [보기](#) [수정](#) [삭제](#)
- 원룸 11 (천안시 서북구 두정동 22 / 470,000원) [보기](#) [수정](#) [삭제](#)
- 원룸 12 (천안시 동남구 원성동 80 / 360,000원) [보기](#) [수정](#) [삭제](#)
- 원룸 13 (천안시 동남구 봉명동 84 / 400,000원) [보기](#) [수정](#) [삭제](#)
- 원룸 14 (천안시 동남구 청수동 59 / 400,000원) [보기](#) [수정](#) [삭제](#)

<php>

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {  
    $room_name = $_POST['room_name'] ?? '';  
    $location = $_POST['location'] ?? '';  
    $price = $_POST['price'] ?? 0;  
    $size = $_POST['size'] ?? 0;  
  
    $user_id = $_SESSION['user_id']; // 여기! 관리자 ID 자동 대입  
  
    if (empty($room_name) && empty($location) && $price > 0) {  
        $stmt = $conn->prepare(query: "INSERT INTO Room (user_id, room_name, location, price, size, created_at) VALUES (?, ?, ?, ?, ?, NOW())");  
        $stmt->bind_param(types: "issdd", var: &$user_id, vars: &$room_name, $location, $price, $size);  
        $stmt->execute();  
        $stmt->close();  
        header(header: "Location: admin_rooms.php"); // 다시 목록으로 이동  
        exit;  
    } else {  
        $error = "모든 필드를 정확히 입력하세요.";  
    }  
}
```

```
$result = $conn->query(query: "SELECT * FROM Room ORDER BY created_at DESC");
```

사용자가 입력한 자취방 이름, 위치, 가격, 면적을 \$_POST 배열에서 받아옵니다. 모든 정보가 채워졌는지 확인 후, Room 테이블에 자취방 정보를 추가해줍니다.

Room 테이블에서 자취방 목록을 최신 등록일 기준으로 조회합니다.

<html>

```
<form method="post" action="">  
    <label>방 이름: <input type="text" name="room_name" required></label><br>  
    <label>위치: <input type="text" name="location" required></label><br>  
    <label>가격(원): <input type="number" name="price" required></label><br>  
    <label>면적(㎡): <input type="number" step="0.1" name="size" required></label><br>  
    <button type="submit">자취방 추가</button>  
</form>
```

```
<strong><?php echo htmlspecialchars(string: $row['room_name']); ?></strong>  
<?php echo htmlspecialchars(string: $row['location']); ?> / <?php echo number_format(num: $row['price']); ?>원)  
<a href="room_detail.php?room_id=<?php echo $row['room_id']; ?>">보기</a>  
<a href="room_edit.php?room_id=<?php echo $row['room_id']; ?>">수정</a>  
<a href="room_delete.php?room_id=<?php echo $row['room_id']; ?>" onclick="return confirm('삭제하시겠습니까?');">삭제</a>
```

관리자가 자취방 정보를 입력할 수 있도록 방 이름, 위치, 가격, 면적을 입력받는 폼을 제공해줍니다. required 속성으로 필수 값을 지정해줍니다.

Room 테이블에서 조회한 자취방 목록을 출력합니다. 각 자취방에 대해 보기, 수정, 삭제 링크를 제공하여 관리자가 자취방을 관리할 수 있습니다.

자취방 수정

방 이름:
위치:
가격(원):
면적(㎡):

[← 목록으로](#)

localhost 내용:

삭제하시겠습니까?

확인

취소

천안로 22 / 330,000원) [보기](#) [수정](#) [삭제](#)

7 / 500,000원) [보기](#) [수정](#) [삭제](#)

[자취방 수정 페이지]

[자취방 삭제 팝업]

- admin_features

옵션 관리

새 옵션 이름:

- 가스레인지
- 건조대
- 냉장고
- 세탁기
- 신발장
- 에어컨
- 엘리베이터
- 인터넷
- 주차장
- 책상
- 침대
- 티비

[← 돌아가기](#)

<php>

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (!empty($_POST['new_feature'])) {
        $stmt = $conn->prepare(query: "INSERT INTO Feature (feature_name) VALUES (?)");
        $stmt->bind_param(types: "s", var: &$_POST['new_feature']);
        $stmt->execute();
        $stmt->close();
    } elseif (isset($_POST['delete_id'])) {
        $stmt = $conn->prepare(query: "DELETE FROM Feature WHERE feature_id = ?");
        $stmt->bind_param(types: "i", var: &$_POST['delete_id']);
        $stmt->execute();
        $stmt->close();
    }
}

$features = $conn->query(query: "SELECT * FROM Feature ORDER BY feature_name ASC");
```

POST에서 요청이 오면 새 옵션을 추가하거나 기존 옵션을 삭제합니다. 옵션을 추가하면 사용자가 입력한 새로운 옵션을 Feature 테이블에 추가합니다. 사용자가 삭제 버튼을 클릭해 옵션을 삭제하면 Feature 테이블에서 해당 옵션을 삭제합니다.

<html>

```
<?php while ($row = $features->fetch_assoc()): ?>
    <li>
        <?php echo htmlspecialchars(string: $row['feature_name']); ?>
        <form method="post" style="display:inline;">
            <input type="hidden" name="delete_id" value="<?php echo $row['feature_id']; ?>">
            <button type="submit" onclick="return confirm('삭제하시겠습니까?');">삭제</button>
        </form>
    </li>
<?php endwhile; ?>
```

Feature 테이블에서 각 옵션을 리스트로 출력하고 각 옵션에 삭제 버튼을 제공합니다. 삭제 버튼을 클릭한 후 사용자가 확인하면 해당 옵션이 삭제됩니다.

- admin_reviews.php

후기 관리

- User4 → 원룸 2 (4/5)

조금 비싸긴 하지만 주변에 편의시설이 잘 구축되어 있어요

삭제

- User2 → 원룸 2 (4/5)

방 평수에 비해 너무 비싼것 같아요. 이동하기는 좋은 위치 같습니다.

삭제

- User1 → 원룸 1 (5/5)

별이 잘 들고 이동하기 좋은 위치 같아요

삭제

[← 돌아가기](#)

<php>

```
if (isset($_POST['delete_review'])) {
    $uid = (int)$_POST['user_id'];
    $rid = (int)$_POST['room_id'];
    $stmt = $conn->prepare(query: "DELETE FROM Review WHERE user_id = ? AND room_id = ?");
    $stmt->bind_param(types: "ii", var: &$uid, vars: &$rid);
    $stmt->execute();
    $stmt->close();
}
```

```
$reviews = $conn->query(query: "SELECT r.user_id, r.room_id, r.rating, r.content, u.name, rm.room_name FROM Review r
JOIN User u ON r.user_id = u.user_id
JOIN Room rm ON r.room_id = rm.room_id ORDER BY r.created_at DESC");
```

POST 요청이 들어오면 해당 리뷰를 삭제합니다. user_id와 Room_id를 이용하여 Review 테이블에서 해당 후기를 삭제합니다.

Review, User, Room 테이블을 JOIN 하여 모든 후기를 검색합니다. Review 테이블에서 사용자, 자취방, 평점을 가져오고 User 테이블에서 사용자 이름, Room 테이블에서 자취방 이름을 검색합니다.

<html>

```
<?php while ($row = $reviews->fetch_assoc()): ?>
    <li>
        <strong><?php echo htmlspecialchars(string: $row['name']); ?></strong> →
        <em><?php echo htmlspecialchars(string: $row['room_name']); ?></em> (<?php echo $row['rating']; ?>/5)
        <p><?php echo nl2br(string: htmlspecialchars(string: $row['content'])); ?></p>
        <form method="post" style="display:inline;">
            <input type="hidden" name="user_id" value="<?php echo $row['user_id']; ?>">
            <input type="hidden" name="room_id" value="<?php echo $row['room_id']; ?>">
            <button type="submit" name="delete_review" onclick="return confirm('후기를 삭제하시겠습니까?');">삭제</button>
        </form>
    </li><hr>
<?php endwhile; ?>
```

Review 테이블에서 가져온 후기를 출력해줍니다. 각 후기는 사용자 이름, 자취방 이름, 평점, 내용, 삭제버튼을 포함하고 있습니다.

- admin_users.php

사용자 관리

- 이름: User1
전화번호: 010-0000-0001
이메일: user1@naver.com

- 이름: User2
전화번호: 010-0000-0002
이메일: user2@naver.com

- 이름: User3
전화번호: 010-0000-0003
이메일: user3@naver.com

- 이름: User4
전화번호: 010-0000-0004
이메일: user4@naver.com

- 이름: User5
전화번호: 010-0000-0005
이메일: user5@naver.com

<php>

```
if (isset($_POST['delete_user'])) {  
    $uid = (int)$_POST['user_id'];  
    $stmt = $conn->prepare(query: "DELETE FROM User WHERE user_id = ?");  
    $stmt->bind_param(types: "i", var: &$uid);  
    $stmt->execute();  
    $stmt->close();  
}  
  
$users = $conn->query(query: "SELECT user_id, name, phone, email FROM User WHERE user_type != 'admin' ORDER BY CAST(SUBSTRING(name, 5) AS UNSIGNED) ASC");
```

관리자가 삭제 버튼을 클릭하면 POST 요청을 통해 삭제할 사용자의 user_id가 서버로 전달됩니다. 전달된 정보를 가지고 해당 사용자를 User 테이블에서 삭제합니다. SQL 쿼리 DELETE를 실행해 해당 사용자의 정보를 삭제합니다.

User 테이블에서 관리자를 제외한 사용자의 목록을 조회해줍니다. 이름, 전화번호, 이메일 순으로 정렬됩니다.

<html>

```
<?php while ($row = $users->fetch_assoc()): ?>  
    <li>  
        <strong>이름:</strong> <?php echo htmlspecialchars(string: $row['name']); ?><br>  
        <strong>전화번호:</strong> <?php echo htmlspecialchars(string: $row['phone']); ?><br>  
        <strong>이메일:</strong> <?php echo htmlspecialchars(string: $row['email']); ?><br>  
        <form method="post" style="display:inline;">  
            <input type="hidden" name="user_id" value="<?php echo $row['user_id']; ?>">  
            <button type="submit" name="delete_user" onclick="return confirm('사용자를 삭제하시겠습니까?');">삭제</button>  
        </form>  
    </li>  
<hr>  
<?php endwhile; ?>
```

사용자 목록을 출력해줍니다. 삭제 버튼을 클릭하면 확인 팝업이 표시됩니다. 삭제 버튼을 누르면 해당 사용자의 user_id와 함께 POST 요청이 전송됩니다. hidden 필드로 user_id를 전송해 서버에 삭제할 대상을 정확하게 인식하도록 합니다.

사용된 JOIN 연산

JOIN이란?

- 릴레이션 R과 S가 있을 때, 공통 속성을 이용해 릴레이션 R과 S의 튜플들을 연결하여 만든 새로운 튜플들을 반환하는 것을 말한다.

프로젝트에서 사용된 JOIN 연산

User, Room 테이블과 Review 테이블	
JOIN 유형	INNER JOIN
각 테이블에서 가져오는 정보	Review 테이블: user_id, room_id, rating, content User 테이블: name Room 테이블: room_name
사용 이유	리뷰 정보와 관련된 사용자 이름과 자취방 이름을 함께 보여주기 위해 사용됩니다. 사용자가 작성한 후기를 조회할 때, 해당 후기를 작성한 사용자와 자취방에 대한 정보를 함께 가져옵니다.
SQL 쿼리	<pre>"SELECT r.user_id, r.room_id, r.rating, r.content, u.name, rm.room_name FROM Review r JOIN User u ON r.user_id = u.user_id JOIN Room rm ON r.room_id = rm.room_id ORDER BY r.created_at DESC");</pre>

Room 테이블과 Likes 테이블	
JOIN 유형	INNER JOIN
각 테이블에서 가져오는 정보	Likes 테이블: user_id Room 테이블: room_id, room_name, location, price, size, created_at
사용 이유	사용자가 좋아요를 누른 자취방의 정보를 검색하기 위해 사용됩니다. 좋아요를 누른 자취방의 이름, 위치, 가격, 크기 등의 정보를 가져와 생성일 순으로 정렬해 사용자에게 보여줍니다.
SQL 쿼리	<pre>\$sql = "SELECT r.room_id, r.room_name, r.location, r.price, r.size, r.created_at FROM Likes l JOIN Room r ON l.room_id = r.room_id WHERE l.user_id = ? ORDER BY l.created_at DESC";</pre>

Room 테이블과 Bookmark 테이블	
JOIN 유형	INNER JOIN
각 테이블에서 가져오는 정보	Bookmark 테이블: user_id Room 테이블: room_id, room_name, location, price, size, created_at
사용 이유	사용자가 북마크한 자취방의 정보를 검색하기 위해 사용됩니다. 사용자가 북마크한 자취방의 이름, 위치, 가격, 크기 등의 정보를 불러오고, 이를 생성일 순으로 정렬하여 보여줍니다.
SQL 쿼리	<pre>\$sql = "SELECT r.room_id, r.room_name, r.location, r.price, r.size, r.created_at FROM Bookmark b JOIN Room r ON b.room_id = r.room_id WHERE b.user_id = ? ORDER BY b.created_at DESC";</pre>

Room 테이블과 Feature 테이블	
JOIN 유형	INNER JOIN
각 테이블에서 가져오는 정보	Feature 테이블: feature_name 포함 테이블: room_id, feature_id
사용 이유	자취방에 포함된 옵션들의 이름을 검색하기 위해 사용됩니다. 포함 테이블을 사용하여 Room과 Feature 테이블 간의 다대다 관계를 관리하고, 자취방에 제공되는 옵션 목록을 출력합니다.
SQL 쿼리	<pre>\$sql = "SELECT f.feature_name FROM Feature f JOIN RoomFeature rf ON f.feature_id = rf.feature_id WHERE rf.room_id = ?";</pre>

User 테이블과 Review 테이블	
JOIN 유형	INNER JOIN
각 테이블에서 가져오는 정보	Review 테이블: rating, content, created_at User 테이블: name
사용 이유	리뷰 작성자의 이름과 해당 리뷰의 평점 및 내용을 가져옵니다. 자취방에 대한 리뷰를 볼 때, 각 리뷰에 작성자의 이름과 평점 및 내용이 포함됩니다.
SQL 쿼리	<pre>\$sql = "SELECT u.name, r.rating, r.content, r.created_at FROM Review r JOIN User u ON r.user_id = u.user_id WHERE r.room_id = ? ORDER BY r.created_at DESC";</pre>

6. 결론

6-1. 프로젝트 결과

이 프로젝트는 자취방 검색, 좋아요, 북마크, 리뷰, 관리자 모드 등 다양한 기능을 제공하는 시스템입니다. 사용자는 로그인 후 자취방을 검색하고 정렬할 수 있으며, 각 자취방에 대해 좋아요, 북마크, 리뷰를 남길 수 있습니다. 관리자는 관리자 페이지에서 자취방, 옵션, 후기, 사용자 등을 효율적으로 관리할 수 있고, 각 항목에 대한 편리한 관리 기능을 제공합니다. 또한, 비밀번호 관리와 세션 관리를 통해 보안을 강화하고 사용자의 개인정보 보호를 할 수 있습니다. 이 시스템은 검색 및 필터링 기능을 제공해 사용자가 원하는 정보를 빠르게 찾을 수 있게 하며, 관리자는 데이터 관리와 인터페이스를 통해 운영을 쉽게 할 수 있습니다.

6-2. 개선사항

● 사용자 인터페이스 개선

- 검색 기능을 더 직관적으로 개선하고, 자신이 원하는 자취방을 더 쉽게 찾을 수 있도록, 검색 기능을 더 추가할 예정입니다.
- 디자인을 개선하여 반응형 웹으로 다양한 화면 크기에서 적절히 보이도록 할 예정입니다.

● 로그인 및 세션 관리

- 현재는 세션이 만료되는 기능이 없어 한번 로그인 하면 로그아웃을 하기 전까지 활동이 없어도 로그인이 유지됩니다. 이는 개인정보 유출의 위험이 있기 때문에 일정 시간동안 활동이 없으면 세션이 만료되도록 세션 타임아웃 기능을 추가할 예정입니다.

6-3. 느낀점

이번 프로젝트를 진행하면서, 이전에 배운 이론을 바탕으로 실제 시스템을 구축하는 과정에서 많은 것을 배우고 성장할 수 있어 매우 보람찼습니다. 특히 데이터베이스 설계의 중요성을 실감할 수 있었습니다. 초기에는 E-R 다이어그램을 작성하고, 이를 통해 테이블 간의 관계를 시각화하며, 각 테이블의 역할을 명확히 정의하는 것이 중요하다는 것을 깨달았습니다. 프로젝트 진행 중 요구사항이 변경될 때마다 설계를 유연하게 수정해야 하는 상황이 많았고, 이를 통해 실제 개발 환경에서 요구사항 변경을 효율적으로 반영하는 방법을 배웠습니다.

또한, 정규화를 통해 데이터 중복을 줄이고 데이터 무결성을 유지하는 것이 얼마나 중요한지 깨달았습니다. 초기 설계에서 부족한 점을 수정하고, 테이블 간의 관계를 명확히 하면서, 데이터베이스가 더욱 일관성 있게 동작하도록 만들 수 있었습니다. 프로젝트 시작 시에는 검색, 정렬, 좋아요 기능처럼 간단한 기능을 구현하려 했지만, 점차 사용자의 편리성을 고려한 추가 기능들을 도입하며 설계를 변경하고 구현을 수정하는 경험을 하였습니다. 이를 통해 효율적인 데이터 관리와 사용자 경험 개선을 위한 기술적 접근법을 익힐 수 있었습니다.

혼자서 설계, 구현, 디자인을 진행하면서 협업의 중요성을 실감했습니다. 만약 팀 프로젝트였다면 각 기능을 분담하고 협력하는 과정에서 더 많은 것을 배울 수 있었을 것입니다. 이번 프로젝트는 웹 개발과 데이터베이스 관리의 전반적인 흐름을 이해하고, 요구사항을 수정하는 유연성뿐만 아니라 중요한 기술적 지식들을 깊이 있게 배울 수 있는 기회였습니다.