

나혼자는 배불러

팀장 : 윤우혁(2017114051)

팀원 : 이승민 (2017113931)

송정현(2018113300)



INDEX

01 프로젝트 소개

- 현황 분석
- 해결 방안
- 프로젝트 개요
- 역할 분담

02 프로젝트 설명

- 주요 기능
- 동시성 제어
- Workflow

03 진행 과정

- 진행 및 코드 관리
- 진행 해보며 느낀 점

04 데모 영상

- 데모 영상 재생



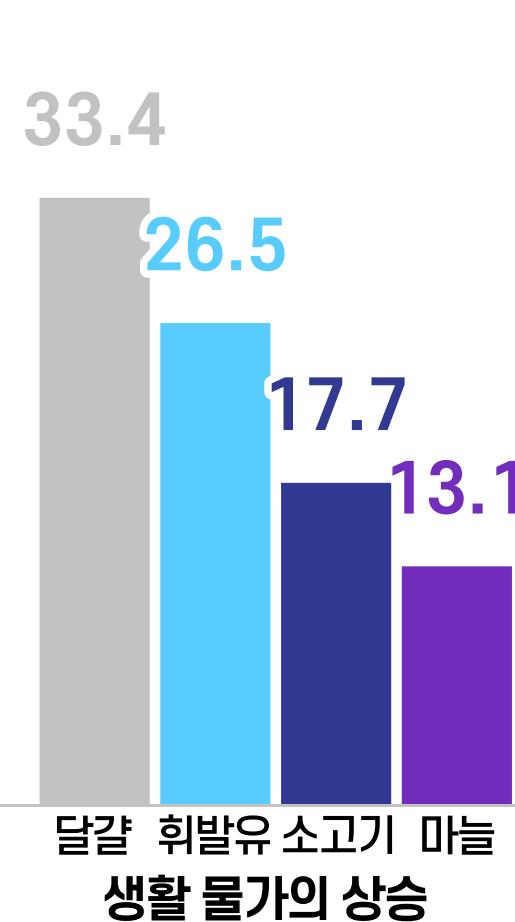
01

프로젝트 소개

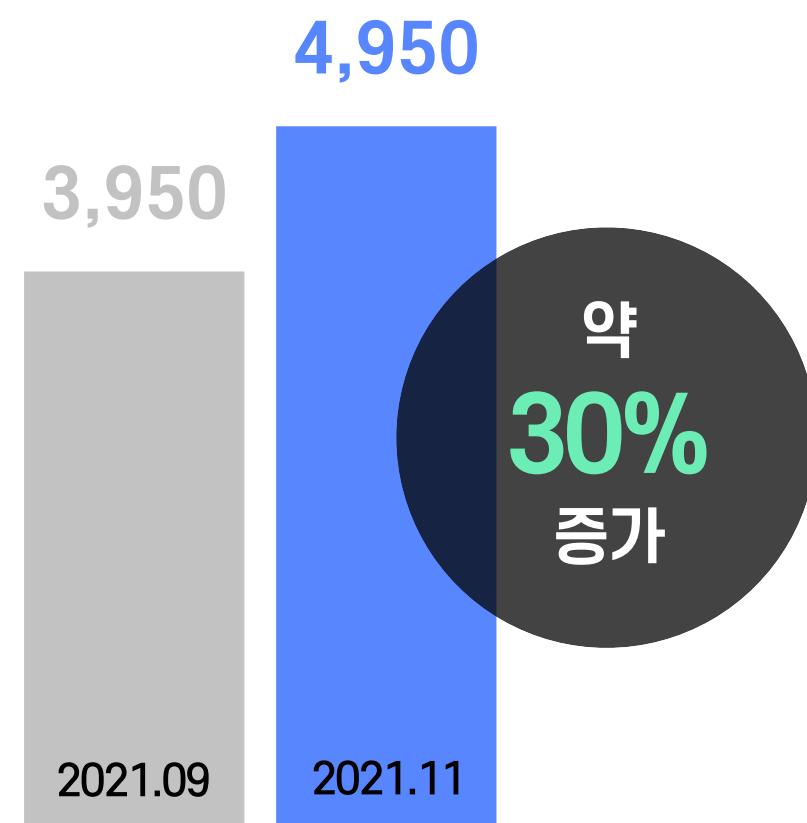
- 현황 분석
- 해결 방안
- 역할 분담

현황 분석

자취생의 음식 구매 시 발생하는 금액



생활 물가의 상승



장기적인 코로나 사태로 인한 배달료의 인상



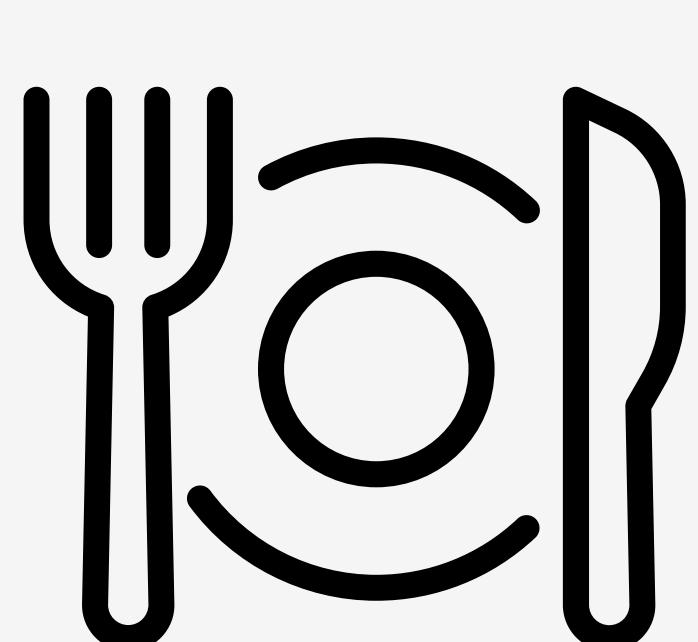
해결 방안

원하는 음식 + 가까운 위치 = 음식 공동구매

원하는 음식

가까운 위치에 거주

음식 공동구매



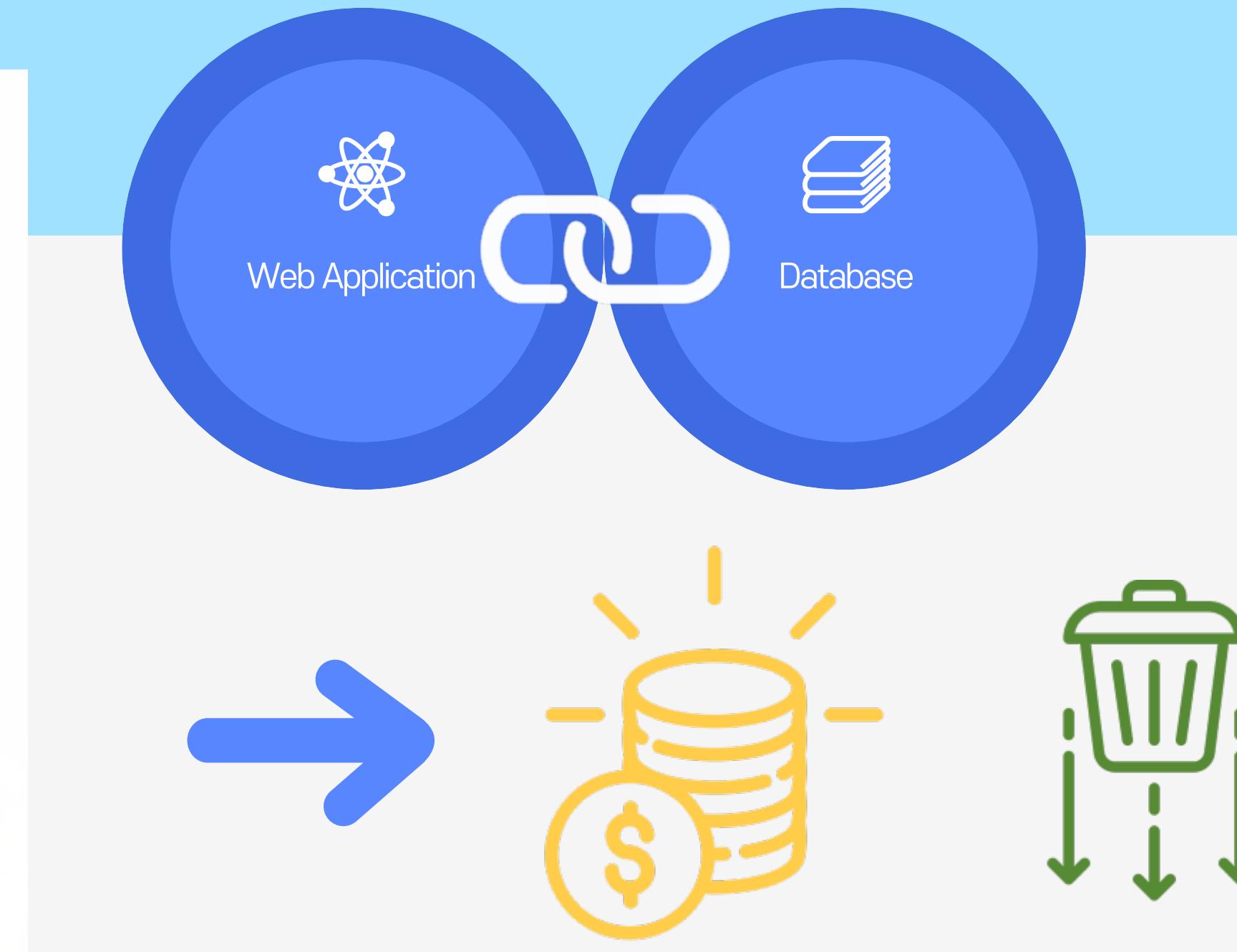
+



=



나 혼자는 배불러 프로젝트 개요



사용자들이 활용할 수 있는 음식 공동구매 서비스 제공
-> 금전적 부담 완화, 음식물 쓰레기 감소

역할 분담

개별 역할

윤우혁

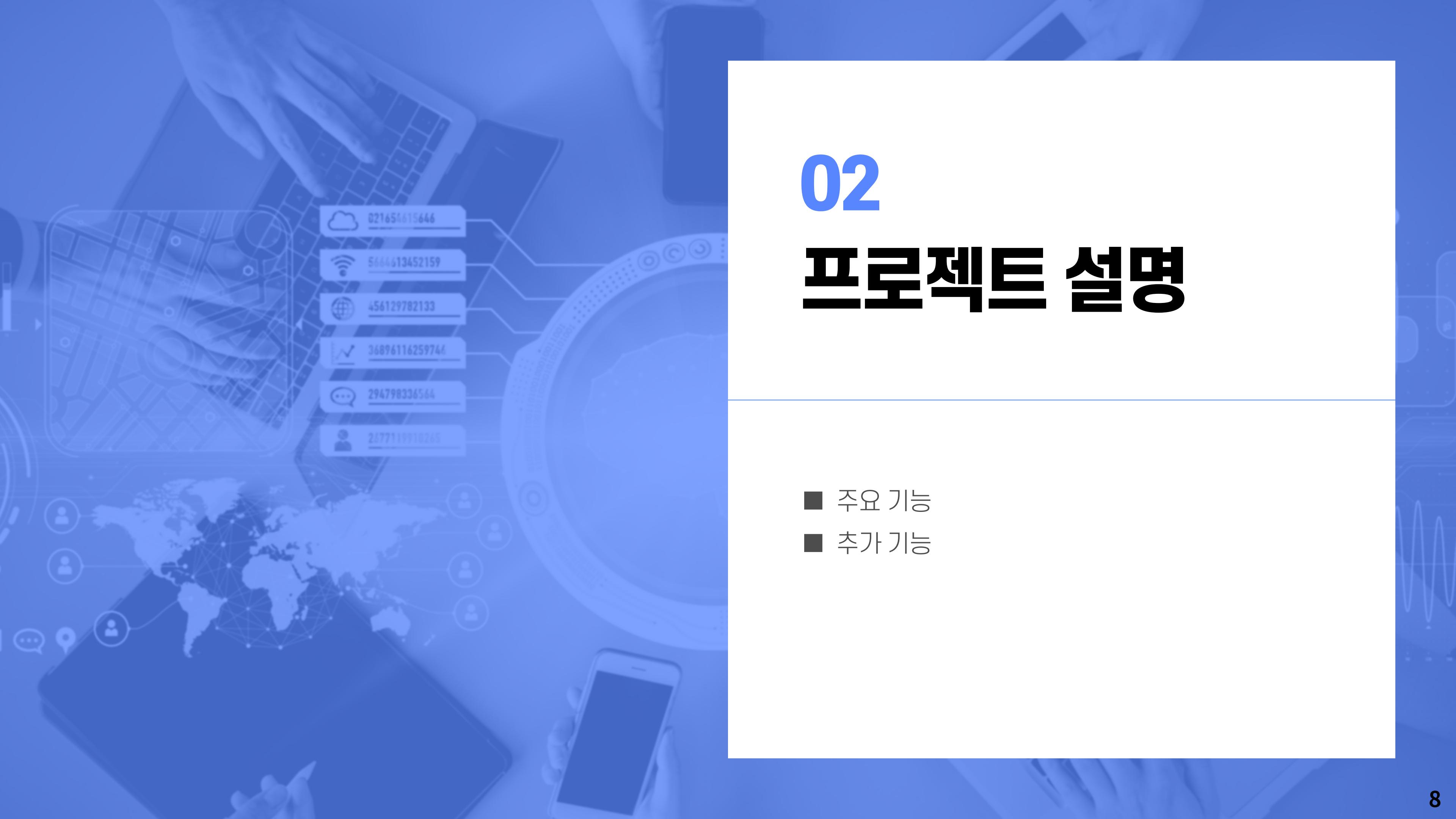
	Post - 게시판 게시판에 게시글 추가/수정/삭제
	Reply - 댓글 게시글의 댓글 추가/수정/삭제
	Category - 게시판 분류 카테고리별 게시글 분류
	관리자 페이지 유저 검색/ 제재, 게시글 수정

이승민

	User - 유저 정보 관리 유저 정보 수정
	Friend - 친구 관리 친구 추가/삭제/요청
	메인 + 공유 게시판 공유 게시판 내용 수정

송정현

	Chatroom - 채팅방 친구와 채팅방 생성/삭제
	Message - 메세지 친구 간 메세지 전송



02

프로젝트 설명

- 주요 기능
- 추가 기능

주요 기능

유저 관리

회원가입/로그인하여 접속 유저를 식별



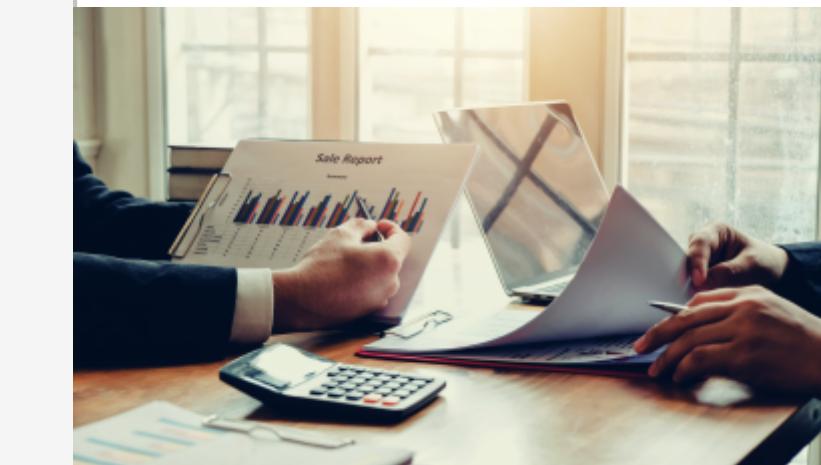
친구 관리

친구 추가를 통해 채팅방 생성



게시판

게시글을 작성하여 음식을 같이 먹을 사람을 구함



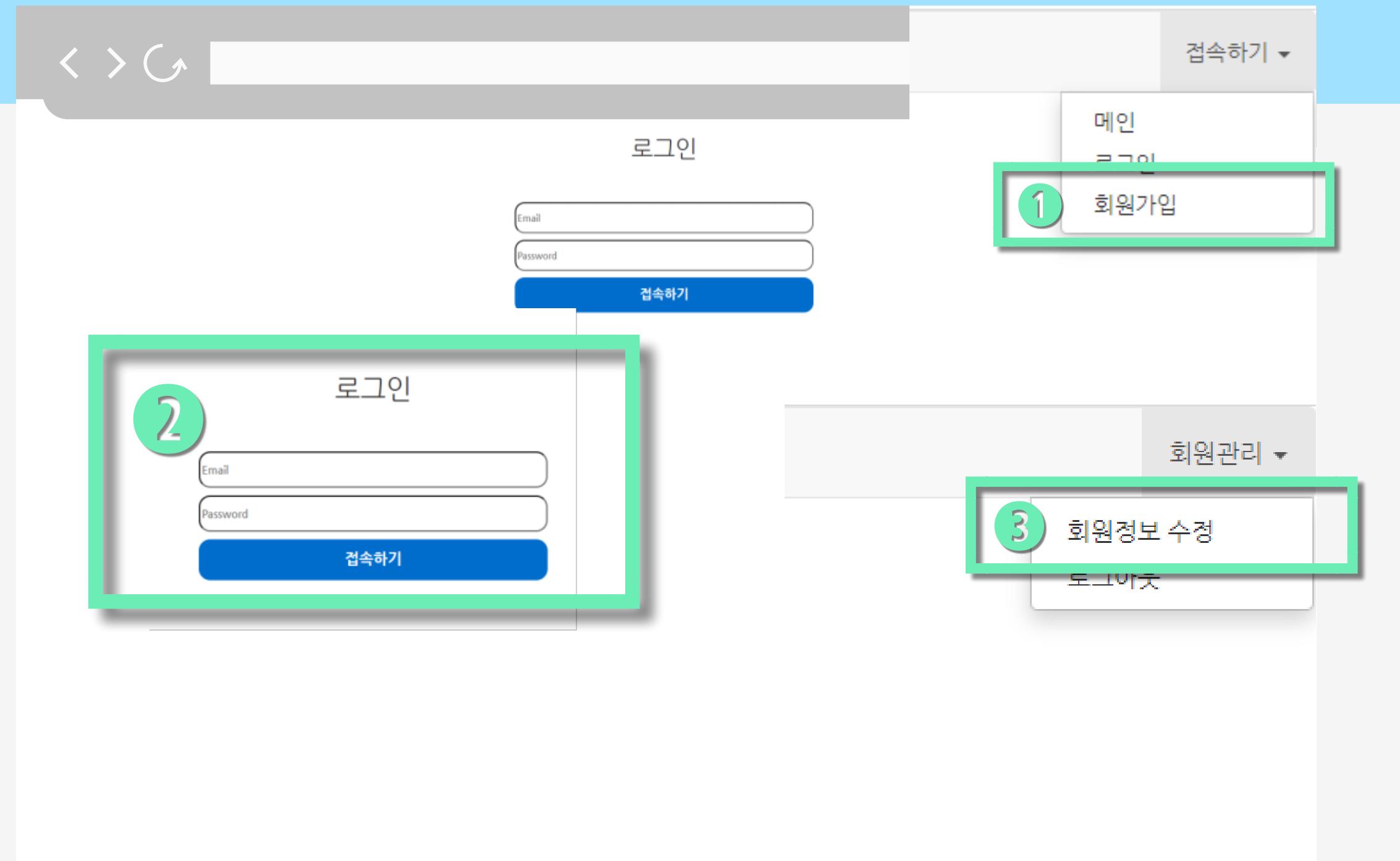
채팅방

채팅방을 통해 상대방과 조율



유저 관리

로그인 회원가입



① 회원 가입하기

- 첫 화면에서 상단 메뉴 [접속하기]에서 [회원가입] 버튼 클릭 -> 회원 가입 진행

② 로그인 하기

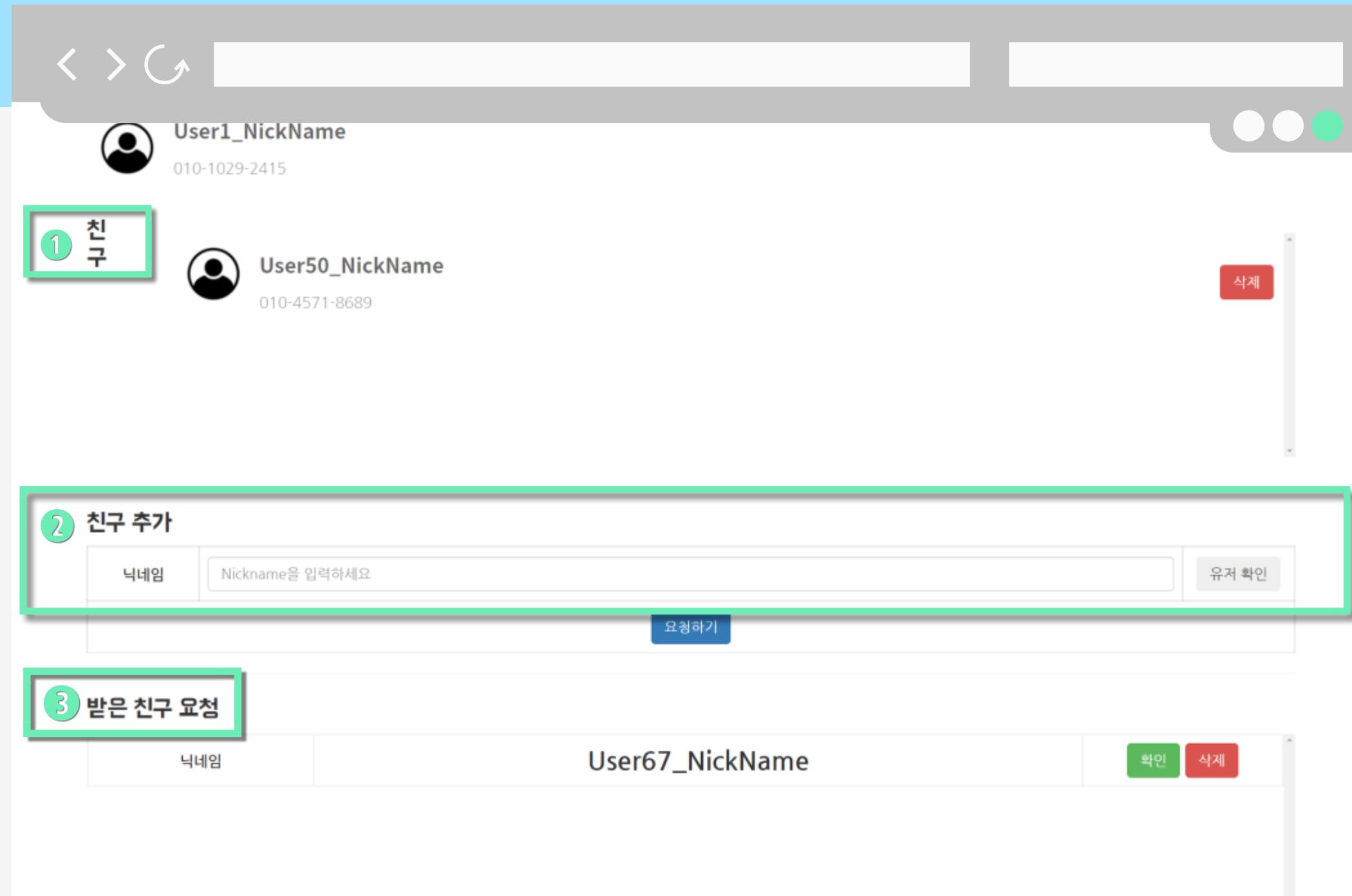
- 자신의 Email과 Password를 입력한 후 로그인

③ 회원 정보 수정 하기

- 로그인 후 메인 화면에서 [회원정보 수정] 버튼 클릭
-> 회원 정보 수정

친구 관리

친구 추가/ 요청/ 삭제



① 친구

- 각 친구 옆의 [삭제] 버튼
-> 해당 친구 삭제

② 친구 추가

- 닉네임 입력 후 [유저 확인] 버튼 -> 유저 존재 여부
- [요청하기] 버튼 -> 친구 추가 요청 전송

③ 받은 친구 요청

- 해당 친구 옆의 [확인] 버튼
-> 친구 요청 수락
- 해당 친구 옆의 [삭제] 버튼
-> 친구 요청 거절

게시판

로그인 회원가입

The screenshot shows a web-based forum interface. At the top, there is a navigation bar with icons for back, forward, and refresh. Below the navigation bar is a horizontal menu bar containing eight category icons: korean, chinese, snack, dessert, japanese, chicken, pizza, and soup. Each icon has a corresponding label below it. A green rectangular box highlights the first two categories: 'korean' and 'chinese'. Below the menu bar is a table listing ten posts. The table has columns for 제목 (Title), 작성자 (Author), and 작성일 (Date). The posts are as follows:

제목	작성자	작성일
post88 title	User3_NickName	2020-09-19 12:32
post64 title	User79_NickName	2020-08-08 05:37
post40 title	User4_NickName	2020-08-03 10:34
post56 title	User41_NickName	2020-06-04 22:12
post32 title	User57_NickName	2020-05-20 12:18
post16 title	User4_NickName	2020-05-09 15:37
post8 title	User53_NickName	2020-03-20 01:19
post72 title	User40_NickName	2020-03-16 19:12
post96 title	User99_NickName	2020-02-20 14:28
post48 title	User76_NickName	2020-02-13 15:13

At the bottom left of the table area is a green button labeled '다음' (Next). At the bottom right is a blue button labeled '글쓰기' (Write Article).

① 카테고리

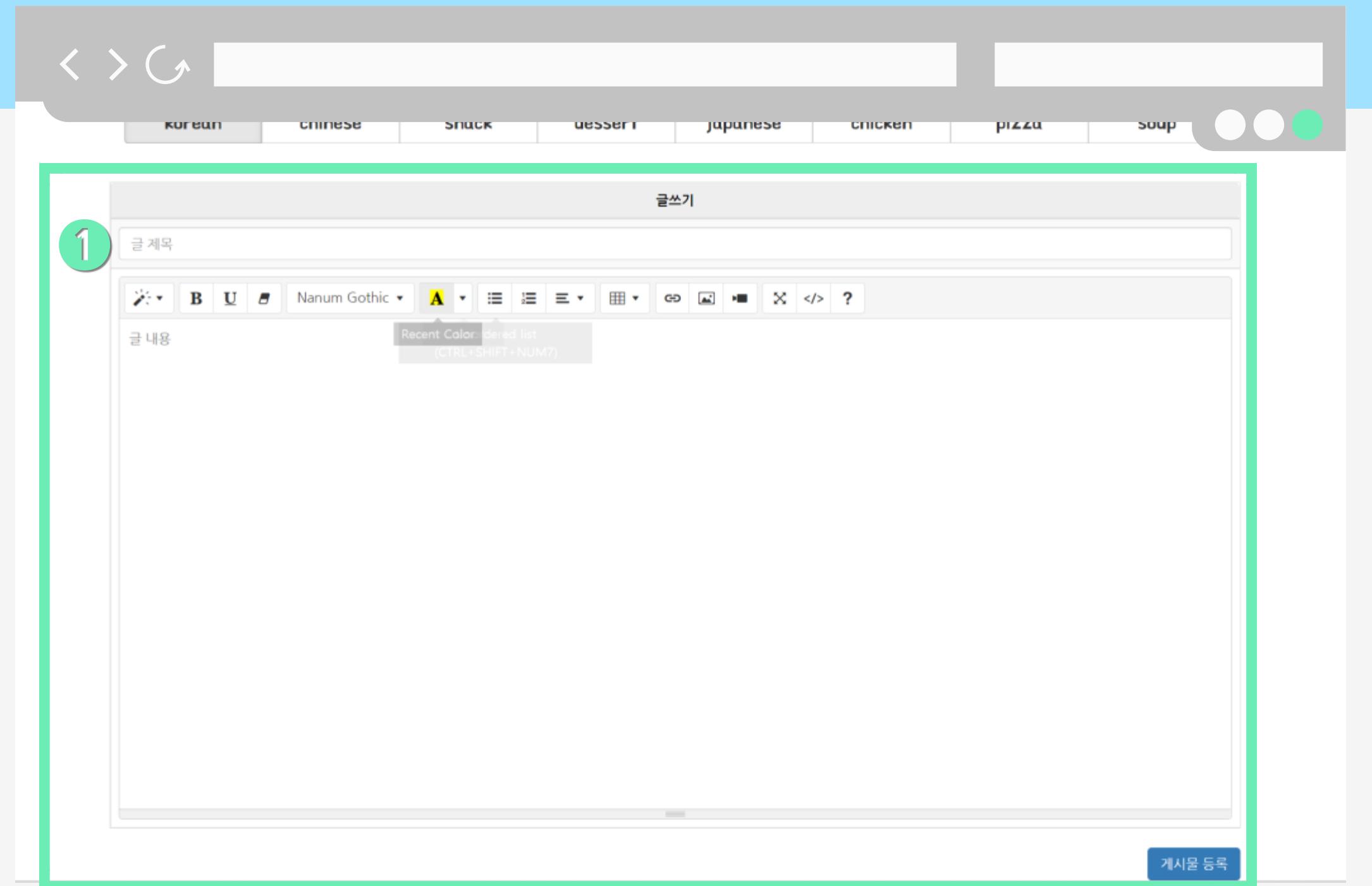
- 카테고리 이미지 클릭
-> 해당 카테고리의 게시물 목록 표출

② 게시글 목록

- 게시글을 클릭
-> 해당 게시글 내부로 진입
- [다음] 버튼
-> 다음 페이지의 게시물 목록 표출
- [글쓰기] 버튼
-> 게시물 작성

게시판

로그인 회원가입



① 게시물 작성

- 게시물 작성 후 [게시물 등록] 버튼 클릭

III. 디자인

```
1 try {
    // empty_clob()을 통해 공간을 확보
    String sql = "insert into post values(Post_SEQ.nextval, ?, empty_clob(), ?, to_date(?, 'yyyy-mm-dd hh24:mi:ss'), ?)"

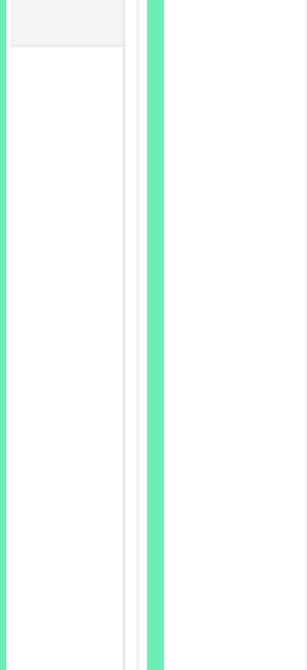
    // 트랜잭션 시작
    conn.setAutoCommit(false);
    pstmt = conn.prepareStatement(sql);

    pstmt.setInt(1, creator_id);
    pstmt.setString(2, postTitle);
    pstmt.setString(3, sdf.format(timestamp).toString());
    pstmt.setInt(4, category);
    pstmt.executeUpdate();

2 // 방금 삽입한 pid 검색
String sql2 = "select pid from post where creator_id = ? and rownum = 1 order by pid desc";
pstmt = conn.prepareStatement(sql2);
pstmt.setInt(1, creator_id);
rs = pstmt.executeQuery();
int prid = -1;
if (rs.next()) {
    prid = rs.getInt(1);
}

3 // for update를 통해 CLOB column을 lock한다.
String sql3 = " select content from post where pid = ? for update wait 5";
pstmt = conn.prepareStatement(sql3);
pstmt.setInt(1, prid);
rs = pstmt.executeQuery();
if (rs.next()) {
    CLOB clob = (CLOB) rs.getBlob(1);
    Writer writer = clob.getCharacterOutputStream();
    Reader src = new CharArrayReader(postContent.toCharArray());
    char[] buffer = new char[1024];
    int read = 0;
    try {
        while ((read = src.read(buffer, 0, 1024)) != -1) {
            writer.write(buffer, 0, read);
        }
        src.close();
        writer.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
conn.commit();
conn.setAutoCommit(true);
```

글 가입



게시물 등록

① 게시글 등록

- 게시글의 내용의 탑입 : CLOB
-> LOB 위치자 사용

② 동시 등록

- LOB 위치자의 변경 발생 가능
-> 정확한 위치자의 참조 필요

1. empty_clob()을 통한 공간 확보
2. 트랜잭션 시작
3. for update wait 5 : clob column lock
fail -> Transaction Rollback

게시판

로그인 회원가입

The screenshot shows a web interface for a food review board. At the top, there are navigation icons (back, forward, search) and a category bar with icons for Korean, Chinese, Snack, Dessert, Japanese, Chicken, Pizza, and Soup. Below this is a post card template.

1 게시글 정보

글 제목	post32 title
작성자	User57_NickName
작성일자	2020-05-20 12:18

32번째 post 입니다.

2 댓글 정보

1 User13_NickName	32번째 post의 1번째 댓글	0020-05-20 12:19
2 User74_NickName	32번째 post의 2번째 댓글	0020-05-20 12:20

댓글을 입력하세요.

등록

목록 글쓴이에게 친구요청

① 게시글 정보

- 게시글의 정보

② 댓글 정보

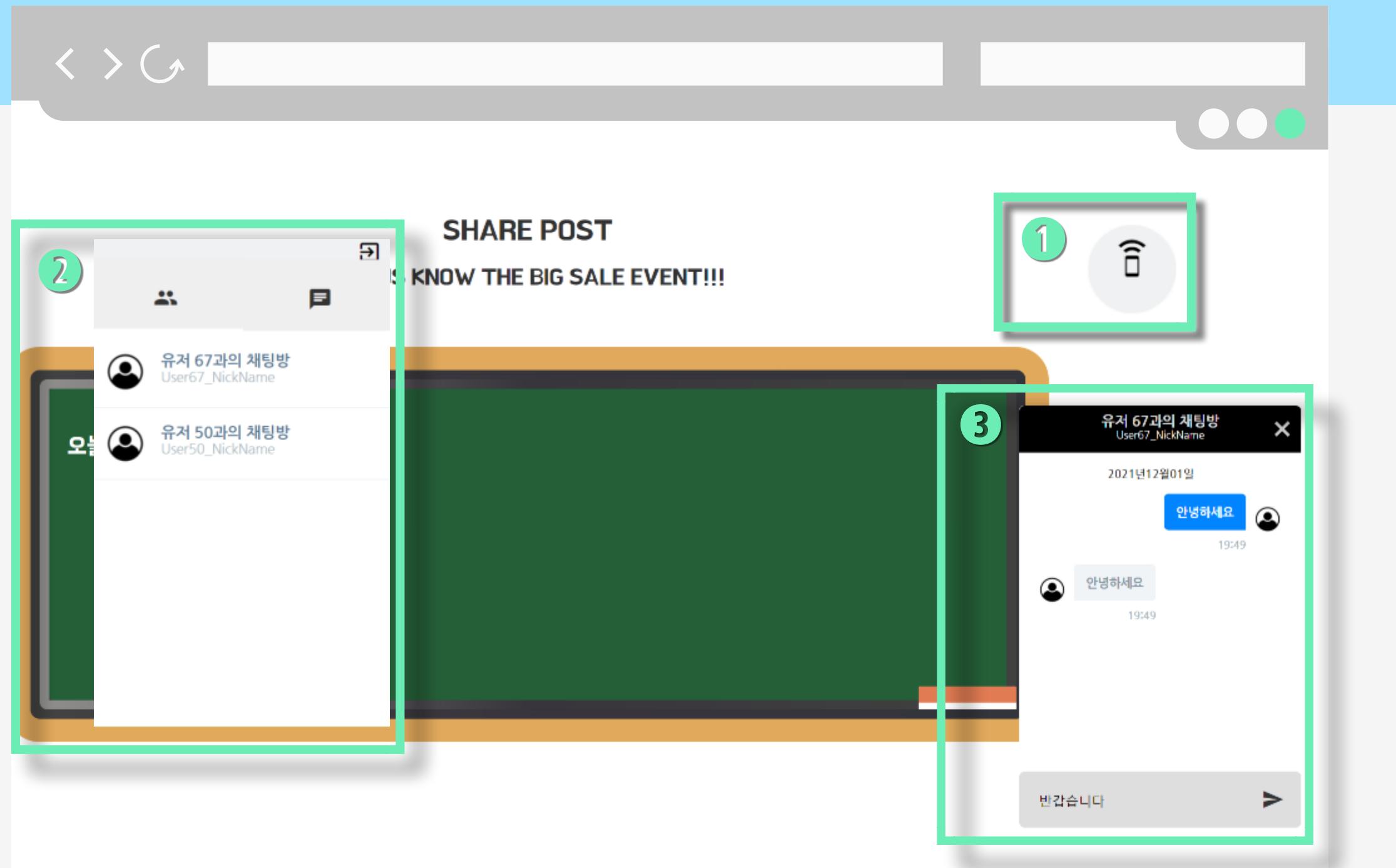
- 댓글 내용 입력 후 [등록] 버튼 클릭
-> 댓글 등록

- [글쓴이에게 친구요청] 버튼 클릭
-> 글쓴이에게 친구 요청

- [목록 버튼] 클릭
-> 게시글 목록으로 되돌아가기

채팅방

채팅방 생성/삭제 + 채팅 메세지 전송



① 하단 토글 버튼

- 버튼을 클릭
-> 확장 -> 친구목록 / 채팅방 목록

② 채팅방 목록

- 채팅방 클릭
-> 해당 채팅방 진입

③ 채팅 하기

- 채팅할 내용을 입력 후 버튼
-> 채팅 전송

관리자 메뉴

유저 검색 + 제재

The screenshot shows the DataBase App interface with the following components:

- Header:** DataBase App, 메인, 게시판, 친구, 관리자 메뉴.
- User Search Section (1):** Contains input fields for Nickname, COUNTRY, CITY, Sex, and Birthdate, along with search buttons (초기화, 검색).
- Search Results (조회 결과):** A table with columns: 성, 이름, PRID, 이메일, 경고, 전화번호, 가입날짜, COUNTRY, STATE, CITY, STREET, 닉네임, 성별, 비고. It shows two rows of data.
- Moderation Section (2):** A table titled "정지 유저 목록" with columns: 성, 이름, PRID, 이메일, 전화번호, 가입날짜, COUNTRY, STATE, CITY, STREET, 닉네임, 성별, 비고. It shows three rows of data, each with a "풀기" button in the last column.
- Action Buttons (3):** Located at the bottom right of the moderation table, these buttons are: 대댓글 입력하세요., 목록, 삭제, 카테고리 변경, 작성자 정지.

① 유저 검색

- 원하는 조건 입력
-> 만족하는 유저들을 조회

② 정지 유저

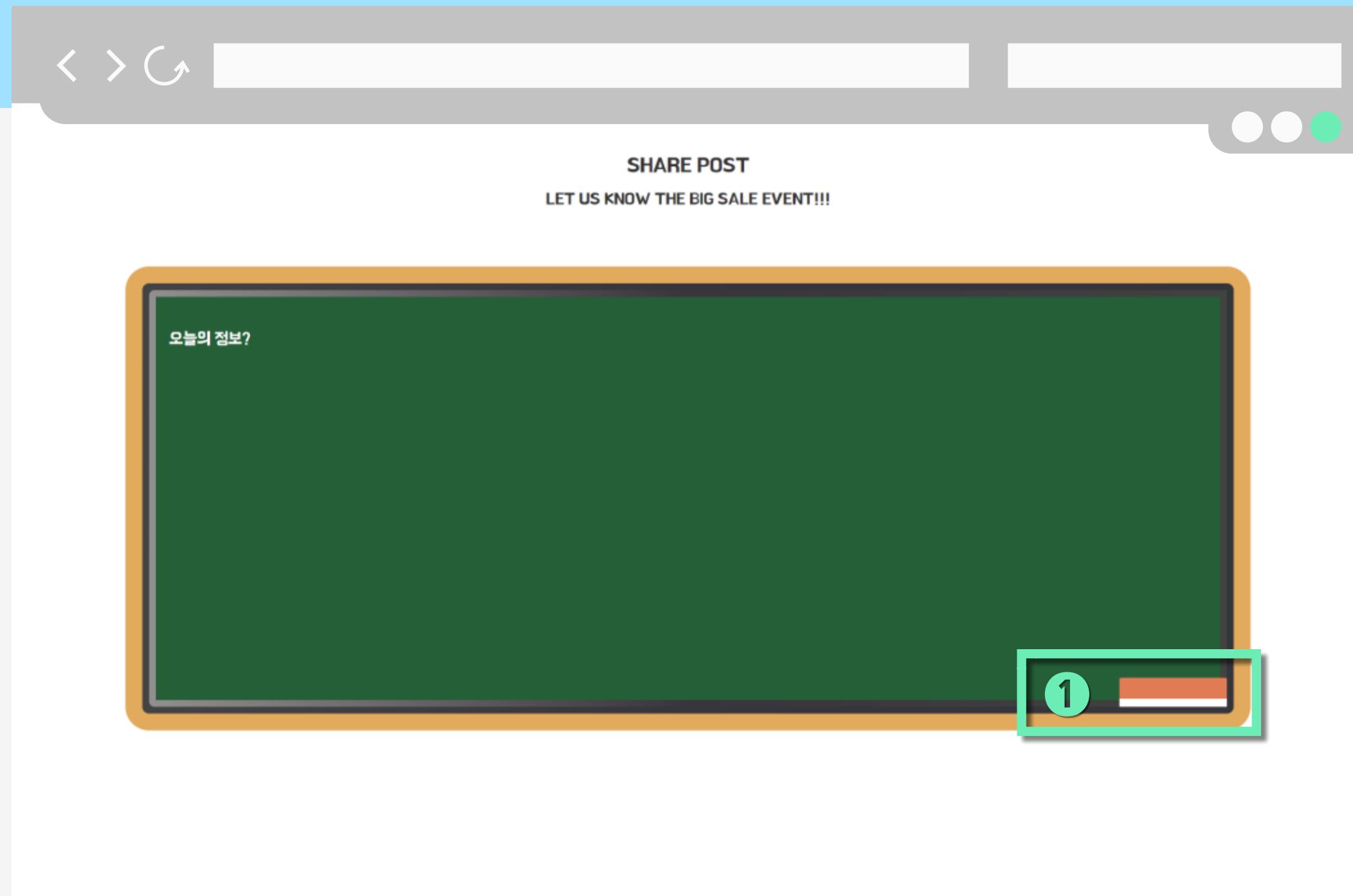
- 해당 유저 옆의 [풀기] 버튼 클릭
-> 해당 유저의 활동정지 해제

③ 게시글 삭제 및 작성자 활동 정지

- 게시글 내부 [삭제] 버튼 클릭
-> 해당 게시글 삭제
- 게시글 내부의 [작성자 정지] 버튼 클릭
-> 해당 작성자 활동 정지

공유 게시판

메인 화면의 정보 공유 게시판



① 공유 게시판 수정

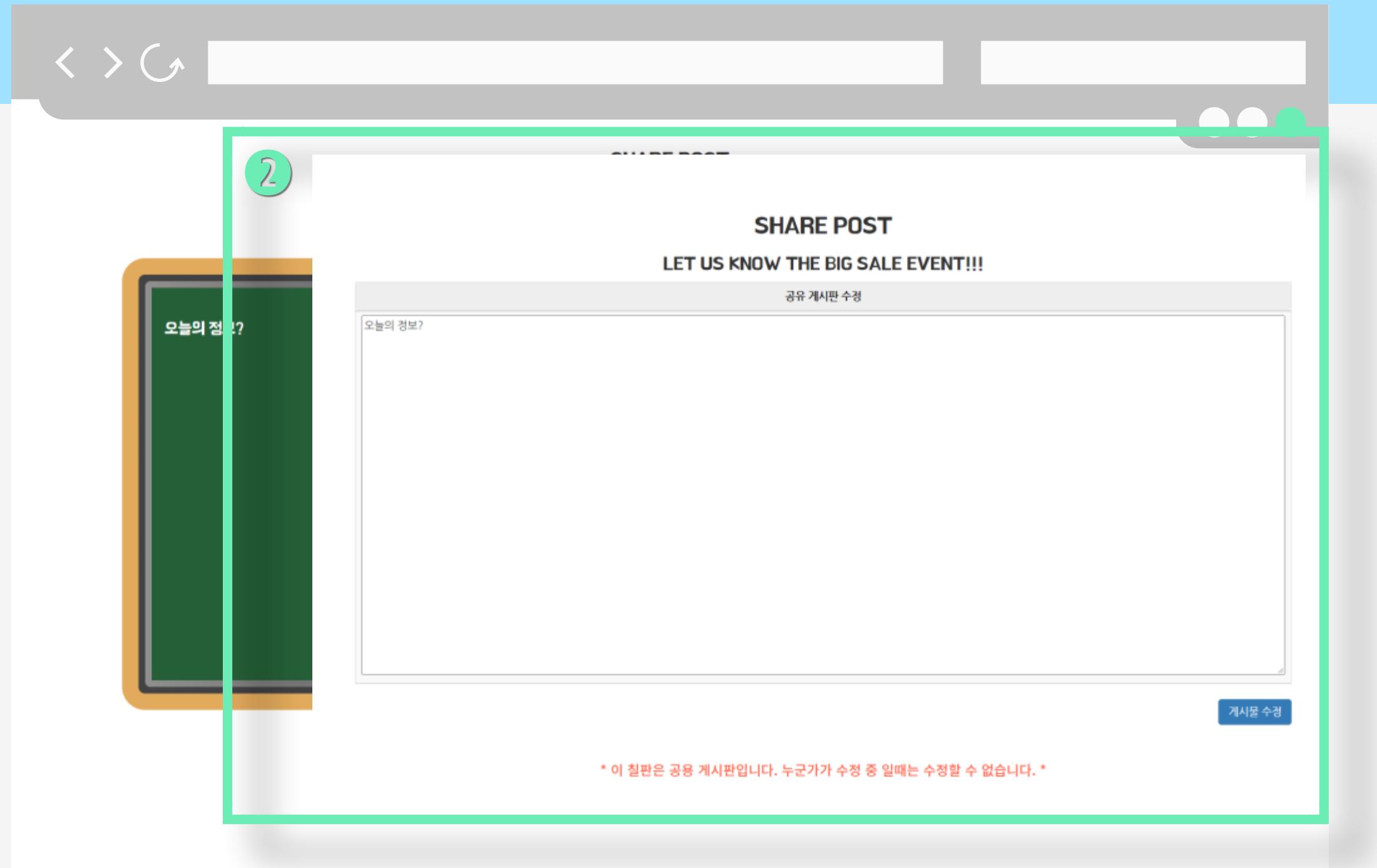
- 칠판 이미지 하단의 지우개를 클릭
-> 공유 게시판 진입

② 내용 수정

- 내용 수정 후 게시물 수정 버튼 클릭
-> 게시물 수정

공유 게시판

메인 화면의 정보 공유 게시판



① 공유 게시판 수정

- 칠판 이미지 하단의 지우개를 클릭
-> 공유 게시판 진입

② 내용 수정

- 내용 수정 후 게시물 수정 버튼 클릭
-> 게시물 수정

1

```

public boolean isUpdated(String latestTime) {
    boolean res = false; // fail
    String sql;

    System.out.println("isUpdated 호출!");
    try {
        sql = "select * from post where pid = -1";

        Statement stmt = conn.createStatement();
        rs = stmt.executeQuery(sql);

        if(rs.next()) {
            System.out.println();
            if(rs.getString("Create_date") == null || rs.getString("Create_date").trim().equals(latestTime.trim())) {
                res = false;
            }else {
                res = true;
            }
        }else {
            rs.close();
            stmt.close();
            res = true;
        }
    } catch (SQLException e) {
        e.printStackTrace();
        res = true;
    }
    return res;
}

```

2

```

public int update(String postContent) {
    int res = -1;
    String sql;
    Timestamp timeStamp = new Timestamp(System.currentTimeMillis());
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    try {
        sql = "select * from post where pid = -1 for update NOWAIT";
        Statement stmt = conn.createStatement();
        rs = stmt.executeQuery(sql);
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        System.out.println(e.getErrorCode() + " : " + e.getMessage());
        return res;
    }
    try {
        sql = "UPDATE post SET content = ? , create_date = to_date(? , 'yyyy-mm-dd hh24:mi:ss') where pid = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setCharacterStream(1, new StringReader(postContent));
        pstmt.setString(2, sdf.format(timeStamp).toString());
        pstmt.setInt(3, -1);
        res = pstmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
        return res;
    }
    return res;
}

```

다

판



① 현재 게시물의 수정 여부

- 최신 갱신 시간과 기존 게시물의 갱신 시간 비교
=> 수정 사항 X
!=> 수정 사항 0 -> 사용자 알람

② 현재 게시물의 점유 여부

- row-level Lock 시스템
 - Oracle의 select for update 문
-> 게시물 점유

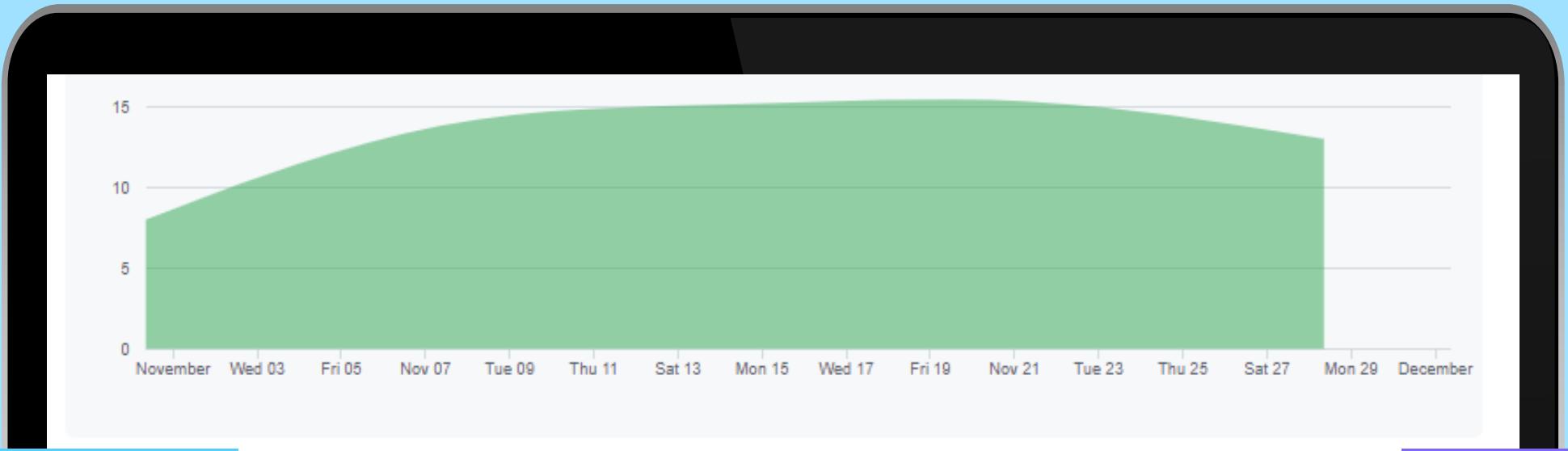


03

진행 과정

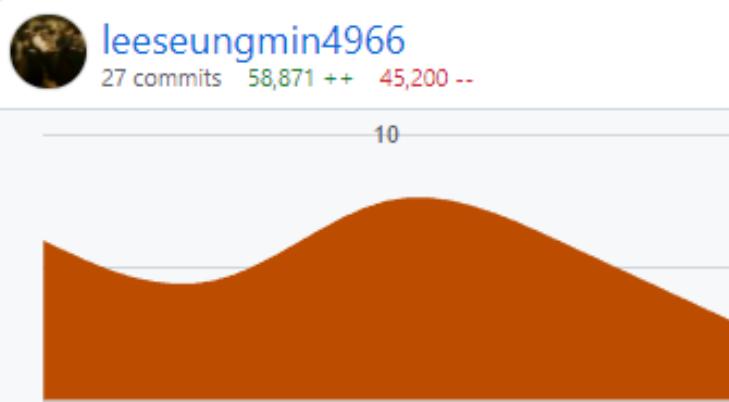
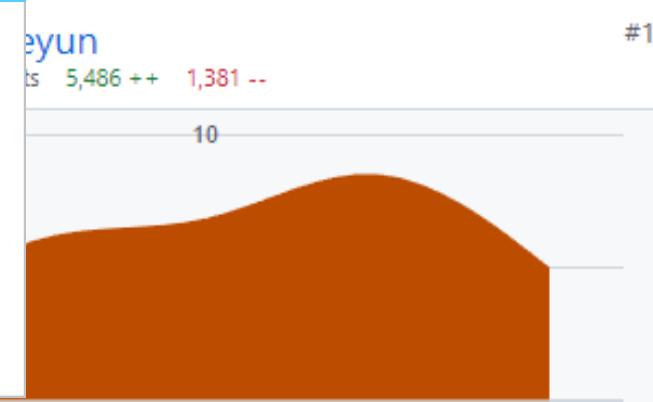
- 프로젝트 진행 과정
- 진행해보며 느낀 점

프로젝트 진행 과정



매주 1회 이상 비대면 미팅

Zoom, Google Meet 서비스를 이용한 비대면 미팅



공유 문서 관리

Google Docs 서비스를 이용한 진행 내용 기록



코드 관리

GIt을 이용한 코드 관리

코드 개발 환경

전체 코드 줄 수 - 8364줄

OS : Windows 10,11, MAC

DBMS : Oracle

Language : JAVA, JS, CSS

진행 해보며 느낀점

의견 차이 극복

- 변수 네이밍
- 함수 구조 설계
- 구현 범위(이상과 현실)

다양한 경험

- DB 설계
- 웹 개발 (html, css, js)





04

데모 영상

■ 데모 영상 재생

감사합니다.

DB 6조

팀장 : 윤우혁(2017114051)

팀원 : 이승민 (2017113931)

송정현(2018113300)

