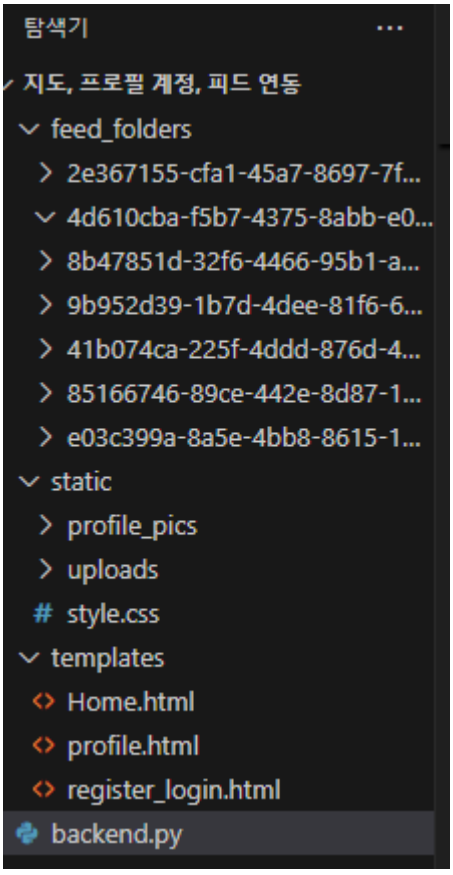
구글 지도와 프로필 계정 연동 + 네이버 oauth 로그인 기능 구현 및 사용법.

<1> 우선 비주얼 스튜디오 코드에 프로젝트 폴더를 하나 만드세요.

다음으로 해당 프로젝트 폴더에 아래와 같이 feed_folders(feed_folders 안에 있는 파일명들은 안 쓰시고 비워두셔도 됩니다.)폴더, static 폴더, templates 폴더를 만드세요.



다음으로 위의 사진처럼 templates 폴더 아래에 Home.html , profile,html, register_login.html, backend.py 파일을 만드세요.

이어서 위의 사진처럼 static 폴더 아래에 profile_pics 폴더, uploads 폴더, style.css 파일을 만드세요.

이렇게 하면 기본적으로 우리가 작성할 파일과 필요한 폴더는 모두 만들었습니다 😊

—

<2> 3개의 html 파일과 backend.py 의 내용을 아래와 같이 채우세요.

제가 보내드린 압축파일을 압축 해제하시면 아래와 같은 내용들은 이미 채워져 있을 겁니다 😊

Home.html 파일의 내용은 **성인** 님의 지도 api html 자료를 많이 참고했고, backend.py 파일의 내용은 **차통** 님의 데이터 베이스 구조와 데이터 베이스 연동 방식을 많이 참고해 구성해 봤습니다 😊

<2>-1. Home.html

```
<!DOCTYPE html>
<!-- HTML5 문서 타입 선언 -->
<html lang="ko">
<!-- 문서의 언어를 한국어로 설정 -->

<head>
  <meta charset="UTF-8" />
  <!-- 문자 인코딩을 UTF-8로 설정 -->

  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <!-- 반응형 디자인을 위해 화면 너비에 따라 콘텐츠 크기 조절 -->

  <title>지도 검색 기능 + 피드 저장</title>
  <!-- 브라우저 탭에 표시될 문서 제목 -->

  <style>
    html, body {
      margin: 0;
      height: 100%;
    }
    /* 브라우저 기본 여백 제거, 전체 높이를 100%로 설정 */

    #map {
      width: 100%;
      height: 100%;
```

```
    }
    /* 지도 div를 화면 전체로 설정 */

    #search-container {
      position: absolute;
      top: 10px;
      left: 50%;
      transform: translateX(-50%);
      z-index: 1000;
      background: white;
      padding: 8px 12px;
      border-radius: 6px;
      box-shadow: 0 2px 6px rgba(0,0,0,0.3);
      display: flex;
      gap: 8px;
    }
    /* 검색창 스타일링: 상단 중앙 고정, 흰색 배경, 그림자 및 라운드 처리 */

    #search-input {
      padding: 6px;
      font-size: 14px;
      width: 200px;
    }
    /* 검색 입력창 스타일 */

    #search-btn {
      padding: 6px 10px;
      font-size: 14px;
      cursor: pointer;
    }
    /* 검색 버튼 스타일 */
  </style>
</head>

<body>
  <!-- 페이지 본문 시작 -->

  <div id="search-container">
```

```
  <!-- 검색창 UI 컨테이너 -->
  <input type="text" id="search-input" placeholder="지역명 입력 (예: 강남구)" /
  <!-- 지역명을 입력할 텍스트 필드 -->
  <button id="search-btn">이동</button>
  <!-- 지도 이동 버튼 -->
</div>

<!-- 프로필 버튼 -->
<form action="{{ url_for('login') }}">
  <button id="profile-button" type="submit">프로필 보기</button>
</form>

<div id="map"></div>
<!-- 지도를 표시할 div 요소 -->

<script>
  // 지도 스타일 설정 (버스 정류장 숨기기)
  const mapStyles = [
    { featureType: "transit.station.bus", stylers: [{ visibility: "off" }] }
  ];

  const fallbackPosition = { lat: 37.5665, lng: 126.9780 };
  // 위치 권한 거부 시 기본 위치 (서울 시청 좌표)

  let map; // 전역 변수: Google 지도 객체
  const markers = []; // 생성된 마커들을 저장할 배열

  // Google Maps API 로드 후 실행될 콜백 함수
  window.initMap = function () {
    // 지도 초기화
    map = new google.maps.Map(document.getElementById("map"), {
      center: fallbackPosition,
      zoom: 16,
      disableDefaultUI: true,  // 기본 UI 제거
      zoomControl: true,       // 줌 컨트롤만 활성화
      styles: mapStyles,       // 커스텀 스타일 적용
    });
```

```
// 브라우저 위치 정보 접근
if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(
    ({ coords }) => {
      const pos = { lat: coords.latitude, lng: coords.longitude };
      map.setCenter(pos);  // 현재 위치로 지도 이동
      console.log("📍 사용자 위치로 지도 이동:", pos);
    },
    () => console.warn("⚠️ 위치 권한 거부됨, fallback 위치 사용")
  );
}

setupSearch();       // 검색 기능 초기화
loadExistingFeeds(); // 저장된 피드 불러오기

// 지도에서 마우스 오른쪽 클릭 시 이벤트 처리
map.addListener("rightclick", (event) => {
  const lat = event.latLng.lat();   // 클릭한 위도
  const lng = event.latLng.lng();   // 클릭한 경도
  const intro = prompt("피드 설명을 입력하세요:"); // 사용자에게 설명 입력 받기

  if (!intro) return;  // 설명 없으면 종료

  // 서버에 피드 저장 요청
  fetch("/add_feed", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ lat, lng, feed_introduction: intro })
  })
  .then(res => res.json())
  .then(data => {
    if (data.success) {
      alert("✅ 피드 저장 완료!");
      addMarker({ lat, lng });  // 저장 성공 시 마커 추가
    } else {
      alert("❌ 피드 저장 실패: " + data.message);
    }
  });
```

```
  });
};

// 검색창 입력 및 버튼 이벤트 설정
const setupSearch = () => {
  const input = document.getElementById("search-input");
  const button = document.getElementById("search-btn");
  const geocoder = new google.maps.Geocoder(); // 주소를 좌표로 변환하는 객

  // 검색 실행 함수
  const handleSearch = () => {
    const query = input.value.trim();
    if (!query) return;

    // 주소 -> 좌표 변환 요청
    geocoder.geocode({ address: query }, (results, status) => {
      if (status === "OK" && results.length > 0) {
        const location = results[0].geometry.location;
        map.setCenter(location); // 검색된 위치로 지도 이동
        console.log("🔍 검색 위치 이동:", location.toJSON());
      } else {
        alert("검색 결과가 없습니다.");
        console.warn("❌ Geocoding 실패:", status);
      }
    });
  };

  button.onclick = handleSearch;  // 버튼 클릭 시 검색 실행
  input.addEventListener("keydown", (e) => {
    if (e.key === "Enter") handleSearch(); // 엔터 입력 시 검색 실행
  });
};

// 마커 추가 함수
const addMarker = ({ lat, lng }) => {
  const marker = new google.maps.Marker({
    position: { lat, lng },  // 마커 위치
    map: map,                // 표시할 지도 객체
```

```
        icon: {
          url: "http://maps.google.com/mapfiles/ms/icons/red-dot.png" // 마커 아이
        }
      });
      markers.push(marker); // 마커 배열에 저장
    };

    // 서버에서 기존 피드 불러와 마커로 표시
    const loadExistingFeeds = () => {
      fetch("/get_feeds")
        .then(res => res.json())
        .then(data => {
          if (data.success && Array.isArray(data.feeds)) {
            data.feeds.forEach(feed => {
              addMarker({ lat: feed.lat, lng: feed.lng });
            });
          } else {
            console.warn("❌ 피드 로딩 실패 또는 형식 오류");
          }
        })
        .catch(err => console.error("❌ 피드 목록 요청 실패:", err));
    };
  </script>

  <!-- Google Maps JavaScript API를 비동기로 불러오고 initMap 실행 -->
  <script async defer
    src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBtqYND3hYOJx
  </script>
</body>
</html>
```

<2>-2. profile.html

```
<!DOCTYPE html>
<html lang="ko"> <!-- 문서 언어를 한국어로 설정 -->

<head>
```

```
<meta charset="UTF-8"> <!-- 문자 인코딩을 UTF-8로 설정 -->
<title>프로필</title> <!-- 브라우저 탭에 표시될 제목 -->

<style>
  /* 전체 프로필 영역을 수직 정렬 및 중앙 정렬 */
  .profile-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin-top: 50px;
  }

  /* 프로필 사진 스타일: 원형, 회색 배경, 커버 이미지 처리 등 */
  .profile-pic {
    width: 170px;
    height: 170px;
    border-radius: 50%; /* 원형으로 만들기 */
    background-color: #ccc; /* 기본 배경색 (이미지 없을 경우 표시) */
    background-size: cover;
    background-position: center;
    cursor: pointer;
    transition: box-shadow 0.3s; /* hover 시 그림자 효과 부드럽게 적용 */
  }

  /* 마우스를 올렸을 때 그림자 효과 */
  .profile-pic:hover {
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);
  }

  /* 파일 입력 필드를 숨김 (이미지 클릭으로 파일 업로드 트리거) */
  #fileInput {
    display: none;
  }

  /* 자기소개 입력창 스타일 */
  textarea {
    width: 300px;
    resize: none; /* 크기 조절 비활성화 */
```

```
        font-size: 16px;
      }

      /* 버튼에 상단 마진 추가 */
      button {
        margin-top: 10px;
      }

      /* 제목 가운데 정렬 */
      h3 {
        text-align: center;
      }
    </style>
</head>

<body>
    <div class="profile-container"> <!-- 전체 프로필 UI 컨테이너 -->
      <h3>{{ user.nickname }} 님의 프로필</h3> <!-- 사용자 닉네임 출력 -->

      <!-- 프로필 사진 업로드 폼 -->
      <form action="{{ url_for('upload_profile_pic') }}" method="POST" enctyp
        <!-- 라벨 클릭 시 파일 선택창 열기 -->
        <label for="fileInput">
          <!-- 프로필 사진 영역: 배경 이미지로 프로필 사진 표시 -->
          <div class="profile-pic" id="profilePic"
            style="background-image:url('/static/uploads/{{ user.profile_pictu
          </div>
        </label>

        <!-- 실제 파일 선택 input (숨김 처리됨) -->
        <input type="file" id="fileInput" name="profile_pic" accept="image/pn
        <!-- 업로드 버튼 -->
        <button type="submit">프로필 사진 저장</button>
      </form>

      <!-- 자기소개 수정 폼 -->
      <form action="{{ url_for('update_intro') }}" method="POST">
        <!-- 자기소개 텍스트 입력 -->
```

```
        <textarea name="intro_text" rows="5" placeholder="자기소개를 작성하세
        <!-- 저장 버튼 -->
        <button type="submit">프로필 소개문구 저장</button>
      </form>

      <!-- 계정 삭제 폼 -->
      <form action="{{ url_for('delete_account') }}" method="POST" onsubmit=
        <button type="submit" style="background-color:red; color:white;">해당
      </form>
    </div>

    <!-- 자바스크립트: 파일 선택 시 미리보기 적용 -->
    <script>
      function loadPreview(event) {
        const input = event.target;

        if (input.files && input.files[0]) {
          const reader = new FileReader(); // 파일 읽기 객체 생성

          reader.onload = function (e) {
            // 파일 로드 완료되면 프로필 사진 영역에 미리보기 표시
            document.getElementById('profilePic').style.backgroundImage = `
          };
          reader.readAsDataURL(input.files[0]); // 파일을 데이터 URL로 읽기
        }
      }
    </script>
  </body>

  </html>
```

<2>-3. register_login.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
```

```
    <meta charset="UTF-8">
    <title>네이버 로그인 연동</title>
</head>
<body>
    <h1>네이버 계정으로 로그인</h1>
    <a href="{{ naver_auth_url }}">
        <img src="https://static.nid.naver.com/oauth/small_g_in.PNG" alt="네이버
    </a>
</body>
</html>
```

<2>-4. backend.py

```
# Flask 웹 프레임워크, 요청 및 JSON 응답, 템플릿 렌더링을 위한 함수 임포트
from flask import Flask, request, jsonify, render_template, redirect, url_for, ses
# CORS(Cross-Origin Resource Sharing)를 허용하기 위한 모듈
from flask_cors import CORS
# 고유 ID 생성을 위한 uuid 모듈
import uuid
# 디렉토리 및 파일 작업을 위한 os 모듈
import os
import pymysql  # MySQL 데이터베이스와의 연결을 위한 pymysql 모듈 불러오기
import requests  # HTTP 요청을 보내기 위한 requests 모듈 불러오기
from urllib.parse import urlencode  # URL 인코딩을 위한 urlencode 함수 불러오기
import mysql.connector  # MySQL 데이터베이스 접속을 위한 또 다른 라이브러리 (머

# Flask 앱 생성
app = Flask(__name__)
app.secret_key = os.urandom(24)  # 세션 보호를 위한 랜덤한 비밀키 설정

# 네이버 앱 등록 후 받은 정보 입력
NAVER_CLIENT_ID = 'E9J2BHv7nU9IlUjaOHF3'  # 네이버 클라이언트 ID
NAVER_CLIENT_SECRET = 'm3QABNs2um'  # 네이버 클라이언트 시크릿
NAVER_REDIRECT_URI = 'http://localhost:5000/naver_callback'  # 네이버 OAut

# 모든 도메인에 대해 CORS 허용 설정
```

```
CORS(app)

# 피드별로 이미지 폴더를 저장할 상위 디렉토리 이름 설정
UPLOAD_FOLDER = 'feed_folders'
# 해당 폴더가 존재하지 않으면 생성
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

# MySQL 데이터베이스 연결 설정
conn = mysql.connector.connect(
    host="localhost",       # MySQL 호스트명 (로컬호스트)
    user="root",            # MySQL 사용자명
    password="*******",  # MySQL 비밀번호
    database="feeds"        # 사용할 데이터베이스 이름
)
# 전역 커서 생성 (CRUD 실행에 사용)
cursor1 = conn.cursor()

db = pymysql.connect(  # pymysql을 사용해 MySQL에 연결
    host="localhost",  # MySQL 호스트 주소
    user="root",  # MySQL 사용자명
    password="*******",  # MySQL 비밀번호

    database="user",  # 사용할 데이터베이스 이름
    charset='utf8mb4',  # 문자 인코딩 설정
    cursorclass=pymysql.cursors.DictCursor  # 결과를 딕셔너리 형식으로 반환
)
cursor2 = db.cursor()  # 데이터베이스 커서 객체 생성

# 루트 URL('/') 접속 시 Home.html 템플릿을 반환
@app.route("/")
def index():
    return render_template("Home.html")  # templates 폴더 내의 HTML 파일 렌더링

# 홈 화면 - 네이버 로그인 링크 생성
@app.route('/register')  # 기본 라우트, 홈 페이지
def login():
    state = str(uuid.uuid4())  # CSRF 방지를 위한 상태 값 생성
    session['oauth_state'] = state  # 세션에 상태 값을 저장
```

```
    naver_auth_url = (  # 네이버 인증 URL 생성

        "https://nid.naver.com/oauth2.0/authorize?" +
        urlencode({  # URL 인코딩을 위한 파라미터 설정
            'response_type': 'code',  # 응답 타입은 code
            'client_id': NAVER_CLIENT_ID,  # 네이버 클라이언트 ID
            'redirect_uri': NAVER_REDIRECT_URI,  # 리다이렉트 URI
            'state': state  # 생성된 상태 값
        })
    )
    return render_template('register_login.html', naver_auth_url=naver_auth_url)

# 네이버 콜백 처리
@app.route('/naver_callback')  # 네이버 로그인 후 리디렉션될 콜백 URL
def naver_callback():
    code = request.args.get('code')  # URL에서 전달된 코드 파라미터를 가져옴
    state = request.args.get('state')  # URL에서 전달된 상태 값을 가져옴

    if state != session.get('oauth_state'):  # CSRF 공격 방지를 위한 상태 값 검증
        return "잘못된 접근입니다. (CSRF)", 400  # 상태 값이 다르면 에러 반환

    # 액세스 토큰 요청
    token_url = 'https://nid.naver.com/oauth2.0/token'  # 네이버 액세스 토큰 요청
    token_params = {  # 액세스 토큰 요청 파라미터
        'grant_type': 'authorization_code',  # grant_type은 authorization_code
        'client_id': NAVER_CLIENT_ID,  # 네이버 클라이언트 ID
        'client_secret': NAVER_CLIENT_SECRET,  # 네이버 클라이언트 시크릿
        'code': code,  # 사용자로부터 받은 코드
        'state': state  # 상태 값
    }
    token_res = requests.get(token_url, params=token_params)  # 액세스 토큰을
    token_data = token_res.json()  # JSON 형식으로 응답을 파싱

    access_token = token_data.get('access_token')  # 액세스 토큰 추출
    if not access_token:  # 액세스 토큰이 없으면 에러 반환
        return "토큰 발급 실패", 400

    # 사용자 정보 요청
```

```
    headers = {'Authorization': f'Bearer {access_token}'}  # Authorization 헤더어
    profile_res = requests.get('https://openapi.naver.com/v1/nid/me', headers=h
    profile_data = profile_res.json()  # JSON 형식으로 응답을 파싱

    if profile_data['resultcode'] != '00':  # 사용자 정보 조회 실패 시 에러 반환
        return "사용자 정보 조회 실패", 400

    user_info = profile_data['response']  # 사용자 정보
    naver_id = user_info['id']  # 네이버 사용자 ID
    nickname = user_info.get('nickname', '네이버사용자')  # 네이버 닉네임 (없으면 

    # DB에 사용자 존재 확인
    cursor2.execute("SELECT * FROM users WHERE user_id = %s", (naver_id,))
    user = cursor2.fetchone()  # 사용자 정보 가져오기

    if not user:  # 사용자가 없으면 새로 등록
        user_uuid = str(uuid.uuid4())  # 새로운 UUID 생성
        cursor2.execute(
            "INSERT INTO users (id, user_id, nickname) VALUES (%s, %s, %s)",  #
            (user_uuid, naver_id, nickname)
        )
        db.commit()  # DB 커밋
        session['user_id'] = user_uuid  # 세션에 사용자 ID 저장
    else:
        session['user_id'] = user['id']  # 이미 존재하는 사용자라면 세션에 사용자 ID 저
    return redirect(url_for('profile'))  # 프로필 페이지로 리디렉션

# 로그인 후 프로필 페이지
@app.route('/profile')  # 프로필 페이지 라우트
def profile():
    user_id = session.get('user_id')  # 세션에서 사용자 ID 가져오기
    if not user_id:  # 사용자 ID가 없으면 로그인 화면으로 리디렉션
        return redirect(url_for('register_login'))

    cursor2.execute("SELECT * FROM users WHERE id = %s", (user_id,))  # DB
    user = cursor2.fetchone()  # 사용자 정보 가져오기

    if not user:  # 사용자를 찾을 수 없으면 에러 반환
```

```
        return "사용자 정보를 찾을 수 없습니다.", 404

    return render_template('profile.html', user=user)  # 프로필 페이지에 사용자 정보

# 프로필 사진 업로드
@app.route('/upload_profile_pic', methods=['POST'])  # 프로필 사진 업로드 처리
def upload_profile_pic():
    if 'profile_pic' not in request.files:  # 파일이 전송되지 않았을 경우 에러 처리
        return "No file part", 400

    file = request.files['profile_pic']  # 업로드된 파일 가져오기
    if file.filename == '':  # 파일명이 비어있을 경우 에러 처리
        return "No selected file", 400

    # static/uploads 폴더가 없으면 생성
    upload_folder = os.path.join(app.root_path, 'static', 'uploads')  # 업로드 폴더
    if not os.path.exists(upload_folder):  # 폴더가 없으면 생성
        os.makedirs(upload_folder)

    filename = str(uuid.uuid4()) + os.path.splitext(file.filename)[1]  # 고유한 파일
    upload_path = os.path.join(upload_folder, filename)  # 파일 저장 경로

    file.save(upload_path)  # 파일 저장

    # 프로필 사진 파일명을 DB에 저장
    user_id = session.get('user_id')  # 세션에서 사용자 ID 가져오기
    cursor2.execute("UPDATE users SET profile_picture = %s WHERE id = %s",
    db.commit()  # DB 커밋
    return redirect(url_for('profile'))  # 프로필 페이지로 리디렉션

# 자기소개 업데이트
@app.route('/update_intro', methods=['POST'])  # 자기소개 수정 처리
def update_intro():
    intro_text = request.form['intro_text']  # 폼에서 자기소개 텍스트 가져오기
    user_id = session.get('user_id')  # 세션에서 사용자 ID 가져오기

    cursor2.execute("UPDATE users SET introduction = %s WHERE id = %s", (i
    db.commit()  # DB 커밋
```

```
    return redirect(url_for('profile'))  # 프로필 페이지로 리디렉션

# 계정 삭제 라우트
@app.route('/delete_account', methods=['POST'])
def delete_account():
    user_id = session.get('user_id')  # 세션에서 사용자 고유 UUID 가져오기
    if not user_id:
        return redirect(url_for('register_login'))  # 로그인 상태 아니면 홈으로

    # DB에서 해당 사용자 정보 삭제
    cursor2.execute("DELETE FROM users WHERE id = %s", (user_id,))
    db.commit()  # 변경사항 저장

    session.clear()  # 세션 초기화 (로그아웃 처리)
    return redirect(url_for('register_login'))  # 홈 페이지로 이동

# 로그아웃
@app.route('/logout')  # 로그아웃 라우트
def logout():
    session.clear()  # 세션 데이터 삭제
    return redirect(url_for('register_login'))  # 홈 페이지로 리디렉션

# 피드 추가 요청을 처리하는 API 엔드포인트
@app.route("/add_feed", methods=["POST"])
def add_feed():
    # 클라이언트에서 전송된 JSON 데이터를 파싱
    data = request.json
    # 위도, 경도, 피드 설명 텍스트 추출
    lat = data.get("lat")
    lng = data.get("lng")
    intro = data.get("feed_introduction", "")

    try:
        # UUID를 이용해 고유 피드 ID 생성
        feed_id = str(uuid.uuid4())
        # feed_id 이름의 개별 폴더 생성 경로 지정
        folder_name = os.path.join(UPLOAD_FOLDER, feed_id)
        # 해당 폴더가 존재하지 않으면 생성
```

```
        os.makedirs(folder_name, exist_ok=True)

        # 절대 경로를 상대 경로로 변환하여 DB에 저장
        relative_path = os.path.relpath(folder_name, start=os.getcwd())

        # 피드 정보를 feed 테이블에 INSERT
        cursor1.execute("""
            INSERT INTO feed (feed_id, latitude, longitude, feed_introduction, feed
            VALUES (%s, %s, %s, %s, %s)
        """, (feed_id, lat, lng, intro, relative_path))
        # 변경사항을 DB에 커밋
        conn.commit()

        # 저장 성공 메시지를 JSON으로 반환
        return jsonify({"success": True, "feed_id": feed_id}),200  # 성공 응답 코드
    except Exception as e:
        # 예외 발생 시 에러 메시지 출력 및 실패 응답 반환
        print("❌ DB 저장 오류:", e)
        return jsonify({"success": False, "message": str(e)}),500  # 실패 응답 코드

# 클라이언트가 기존 피드 목록을 요청할 때 사용하는 GET API
@app.route('/get_feeds', methods=['GET'])
def get_feeds():
    # 결과를 딕셔너리 형태로 받기 위해 cursor 설정
    cursor1 = conn.cursor(dictionary=True)
    # 피드 테이블에서 위도, 경도, 설명을 선택하여 조회
    cursor1.execute("SELECT latitude AS lat, longitude AS lng, feed_introductio
    # 모든 결과를 리스트 형태로 저장
    feeds = cursor1.fetchall()
    # JSON 형식으로 응답 반환
    return jsonify(success=True, feeds=feeds)  # ✅ success 필드를 명시적으로 포

# 이 파일이 직접 실행될 경우 개발 서버 실행
if __name__ == "__main__":
    app.run(debug=True)  # 디버그 모드로 실행 (변경 시 자동 리로드 등 편의 기능 제공
```

—

<3> 2개의 데이터베이스를 MySQL workbench 에 만들 겁니다. MySQL workbench 에 접속 후,

‘create database feeds;

create database user;’

쿼리문을 작성해서 feeds, user 데이터 베이스를 만듭니다.

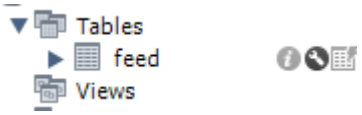
이어서 다음의 쿼리문을 작성해서 feeds 데이터 베이스의 feed 테이블을 생성합니다.

```
CREATE TABLE feeds.feed (
feed_id VARCHAR(255) PRIMARY KEY NOT NULL
);
```

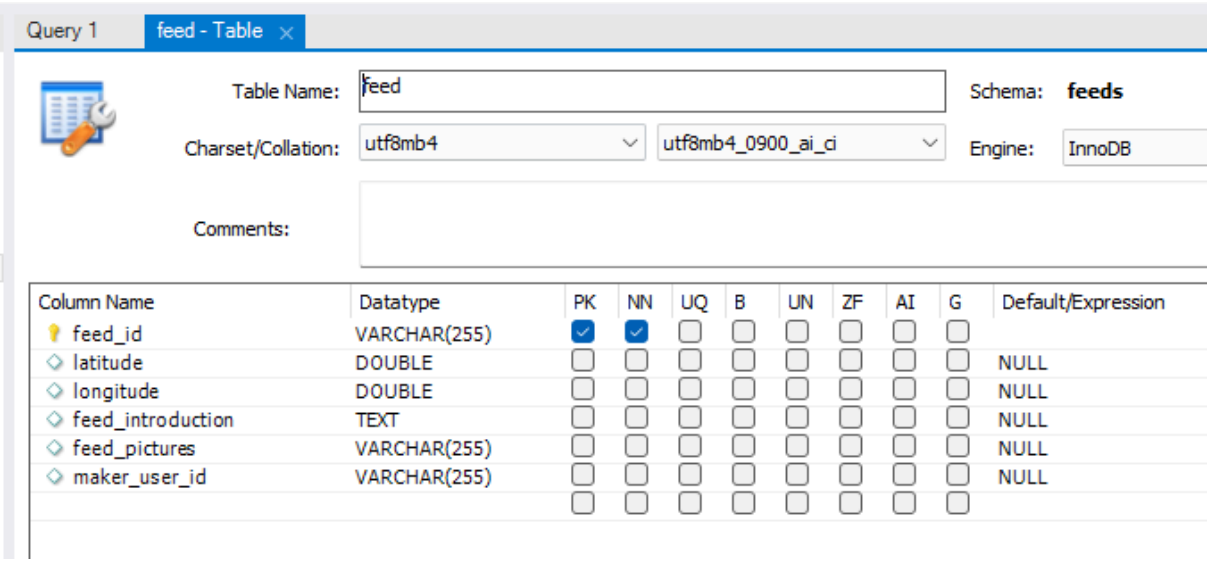
이어서 다음의 쿼리문을 작성해서 user 데이터 베이스의 users 테이블을 생성합니다.

```
CREATE TABLE user.users (
id VARCHAR(300) PRIMARY KEY NOT NULL,
user_id VARCHAR(500) NOT NULL UNIQUE
);
```

그리고 feed 테이블의 ‘수정’ 버튼 = ‘렌치’ 버튼을 아래와 같이 클릭합니다.

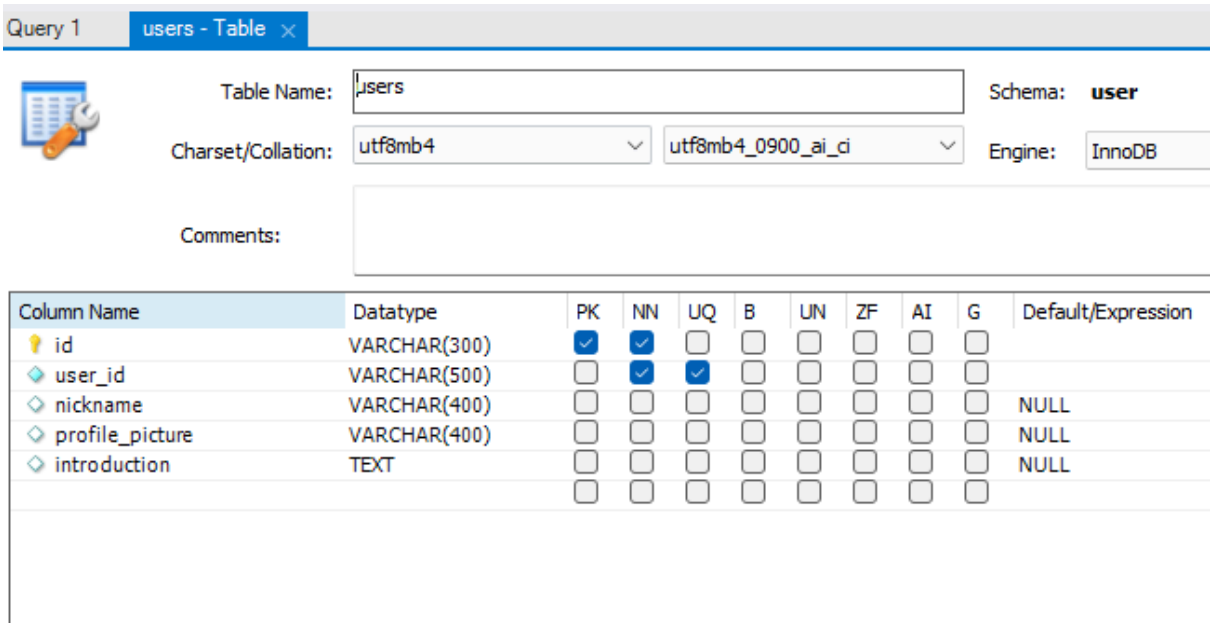


이어서 feed 테이블에 아래와 같은 추가 속성을 추가합니다.



수정 후 'apply' 를 눌러 변경사항 적용 후 수정 창을 닫고 이어서 'users' 테이블의 수정 버튼을 클릭합니다.

이어서 users 테이블에 아래와 같이 다른 3개의 추가 속성을 추가합니다.

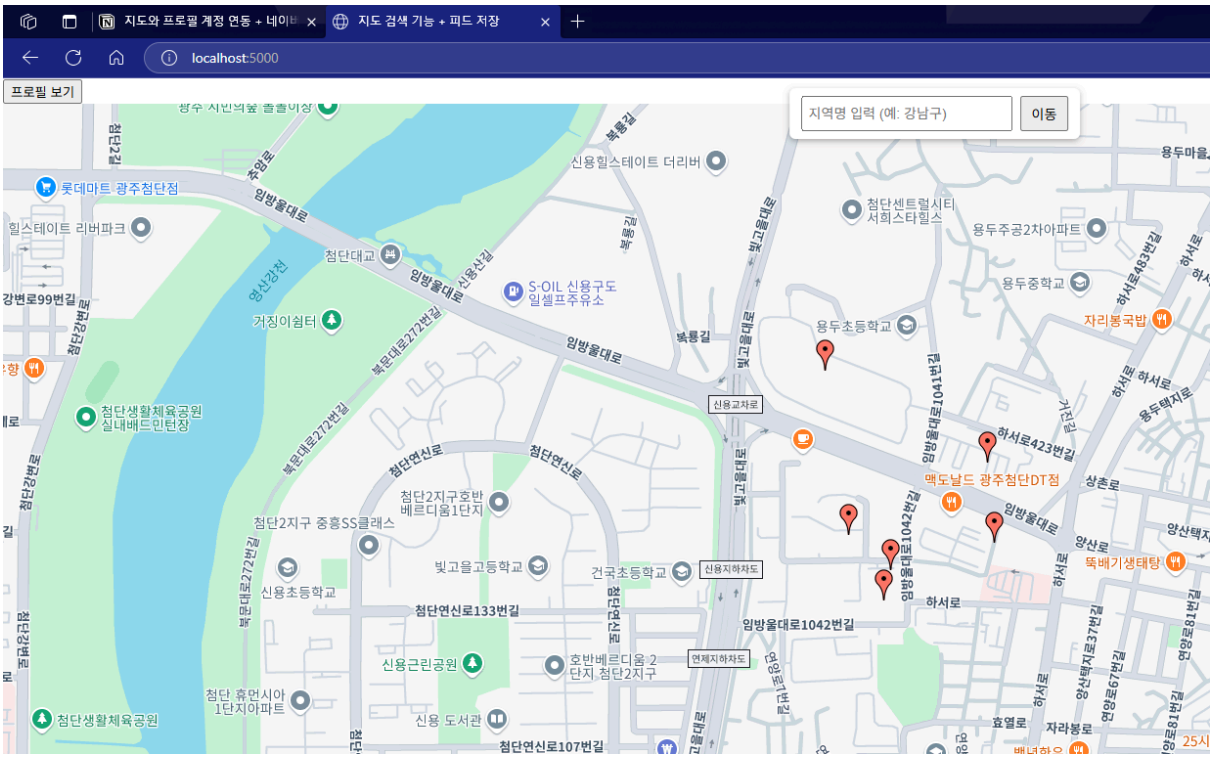


수정 사항을 apply 버튼을 눌러 저장하고 MySQL workbench 를 켜둔 채로 다음 단계로 넘어갑니다(결과 확인을 위한 것입니다).

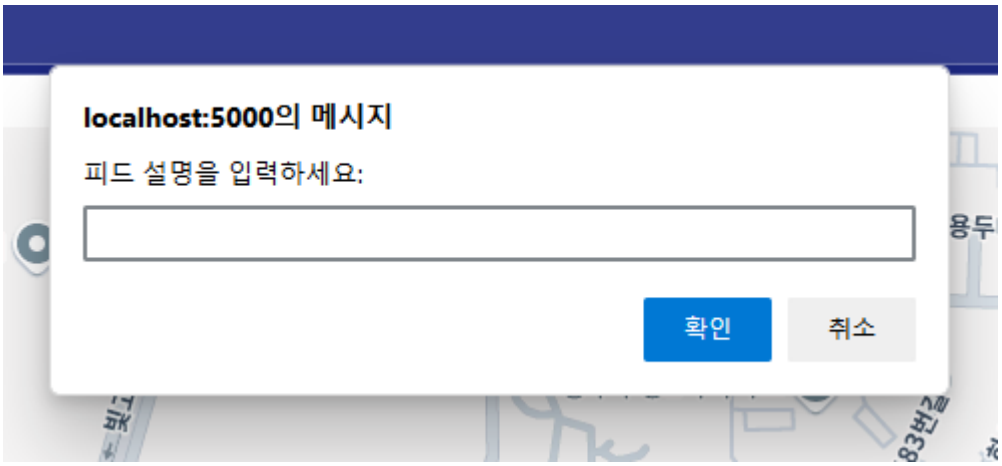
—

<4> backend.py 를 ctrl+f5 를 클릭해 flask 서버와 함께 실행합니다. flask 서버 실행 후, 웹 브라우저를 열고http://localhost:5000 에 접속합니다.

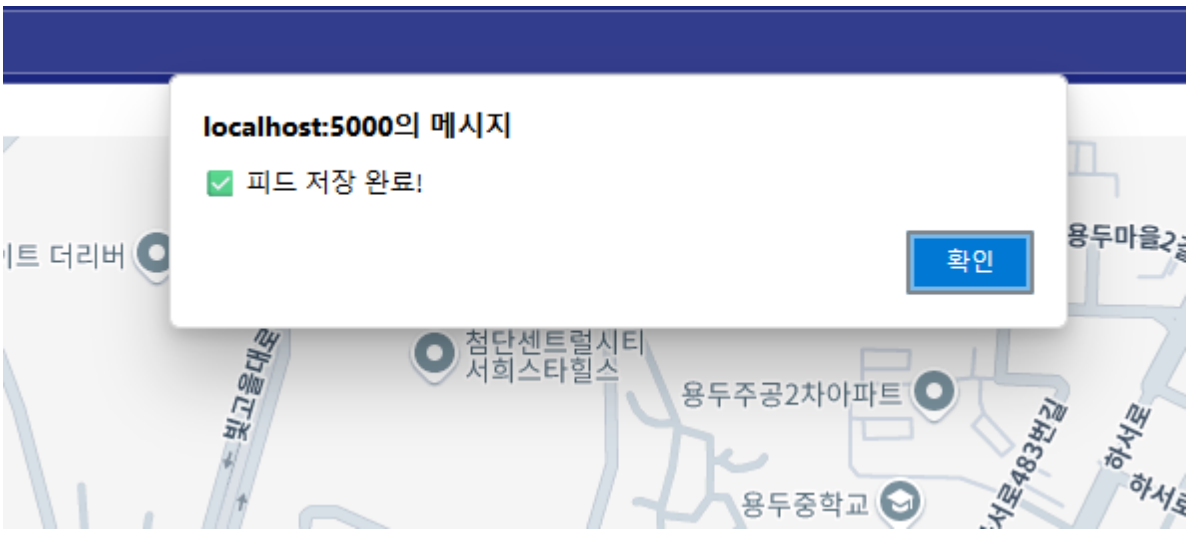
그럼 아래처럼 구글 지도가 보일 텐데, 여기서 피드 생성을 원하시는 위치에 대고 마우스 오른쪽 버튼을 클릭합니다.



그럼 아래의 화면처럼 '피드 설명을 입력하세요.' 라는 창이 뜹니다. 원하는 피드의 소개문구를 작성한 다음 확인 버튼을 누르세요.

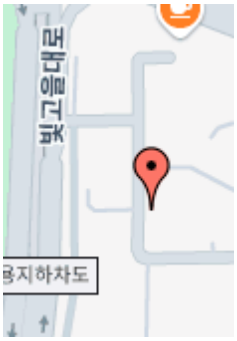


그럼 아래와 같이 '피드 저장 완료!' 프롬프트가 뜰 텐데, 확인을 클릭해 줍니다.

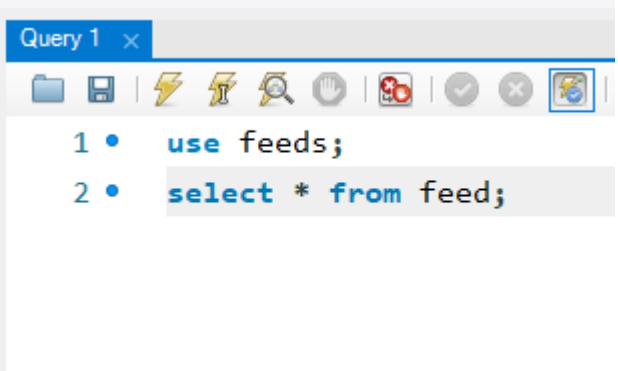


그럼 아래와 같이 정상적으로 피드 마커가 내가 클릭한 지점에 추가되는 것을 확인할 수 있습니다.

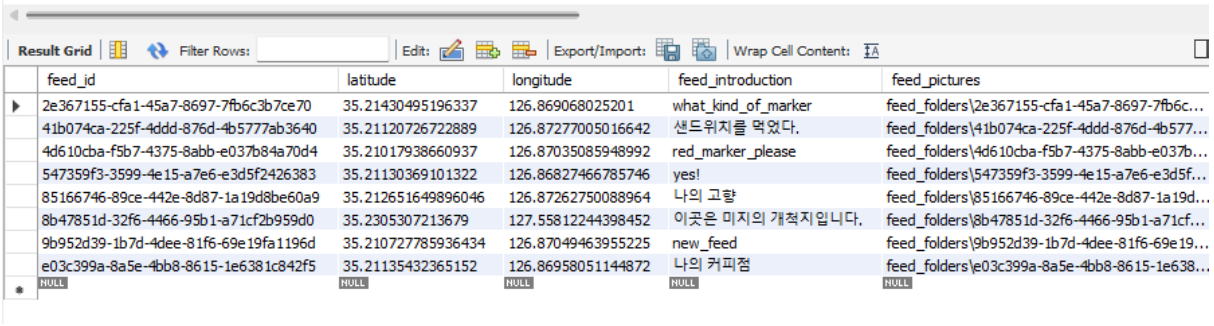
페이지를 여러 번 새로고침해도 해당 피드 마커는 유지될 겁니다 😊



다음으로 MySQL workbench 를 열어서 root 사용자를 쓰는 임의의 connection 으로 접속한 다음 다음의 쿼리문을 작성해 실행해 줍니다.



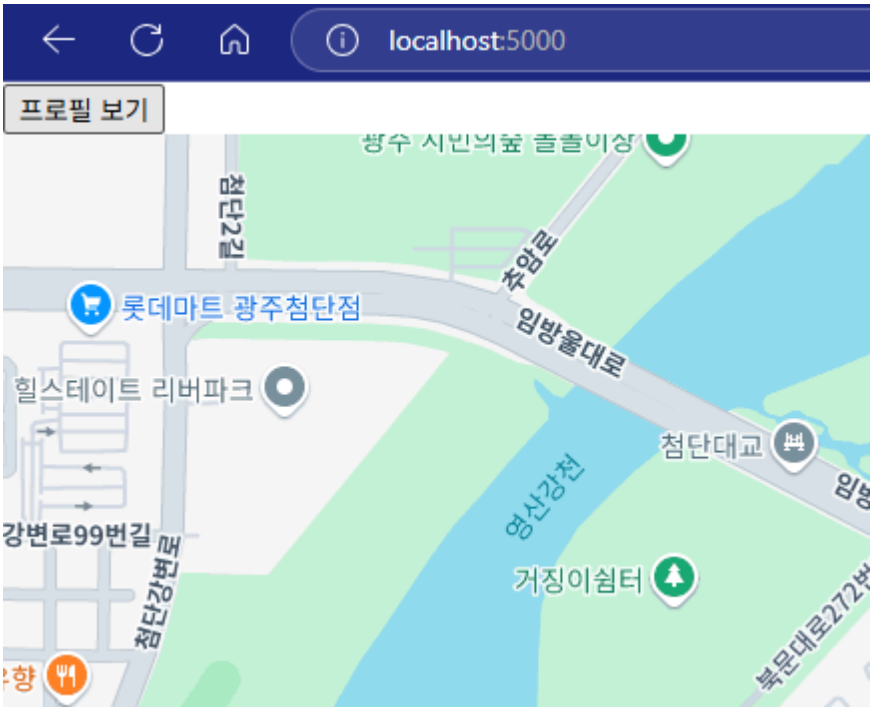
그러면 아래와 같이 내가 만든 데이터 베이스 feeds 의 feed table 에 내가 생성한 피드의 경도(latitude), 위도(longitude), 피드의 고유 아이디(feed_id), 내가 작성한 피드 소개문구(feed_introduction), 내가 생성한 피드의 사진들을 저장할 폴더의 상대경로 주소 값(feed_pictures)가 저장된 것을 확인할 수 있습니다! 🙂



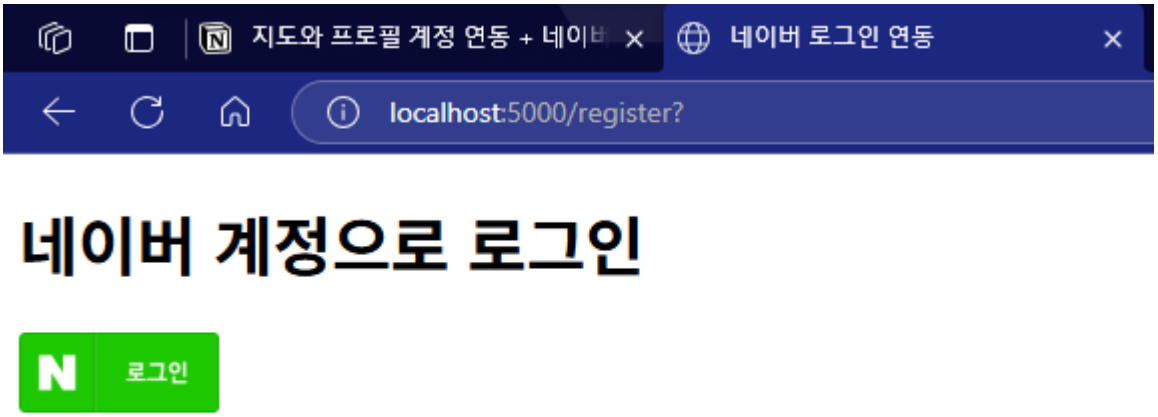
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

feed_id	latitude	longitude	feed_introduction	feed_pictures
2e367155-cfa1-45a7-8697-7fb6c3b7ce70	35.21430495196337	126.869068025201	what_kind_of_marker	feed_folders\2e367155-cfa1-45a7-8697-7fb6c...
41b074ca-225f-4ddd-876d-4b5777ab3640	35.21120726722889	126.87277005016642	샌드위치를 먹었다.	feed_folders\41b074ca-225f-4ddd-876d-4b577...
4d610cba-f5b7-4375-8abb-e037b84a70d4	35.21017938660937	126.87035085948992	red_marker_please	feed_folders\4d610cba-f5b7-4375-8abb-e037b...
547359f3-3599-4e15-a7e6-e3d5f2426383	35.21130369101322	126.86827466785746	yes!	feed_folders\547359f3-3599-4e15-a7e6-e3d5f...
85166746-89ce-442e-8d87-1a19d8be60a9	35.212651649896046	126.87262750088964	나의 고향	feed_folders\85166746-89ce-442e-8d87-1a19d...
8b47851d-32f6-4466-95b1-a71cf2b959d0	35.2305307213679	127.55812244398452	이곳은 미지의 개척지입니다.	feed_folders\8b47851d-32f6-4466-95b1-a71cf...
9b952d39-1b7d-4dee-81f6-69e19fa1196d	35.210727785936434	126.87049463955225	new_feed	feed_folders\9b952d39-1b7d-4dee-81f6-69e19...
e03c399a-8a5e-4bb8-8615-1e6381c842f5	35.21135432365152	126.86958051144872	나의 커피점	feed_folders\e03c399a-8a5e-4bb8-8615-1e638...
NULL	NULL	NULL	NULL	NULL

<5> 이번에는 홈 화면에서, 화면의 왼쪽 최상단의 '프로필 보기' 버튼을 클릭해 봅니다.



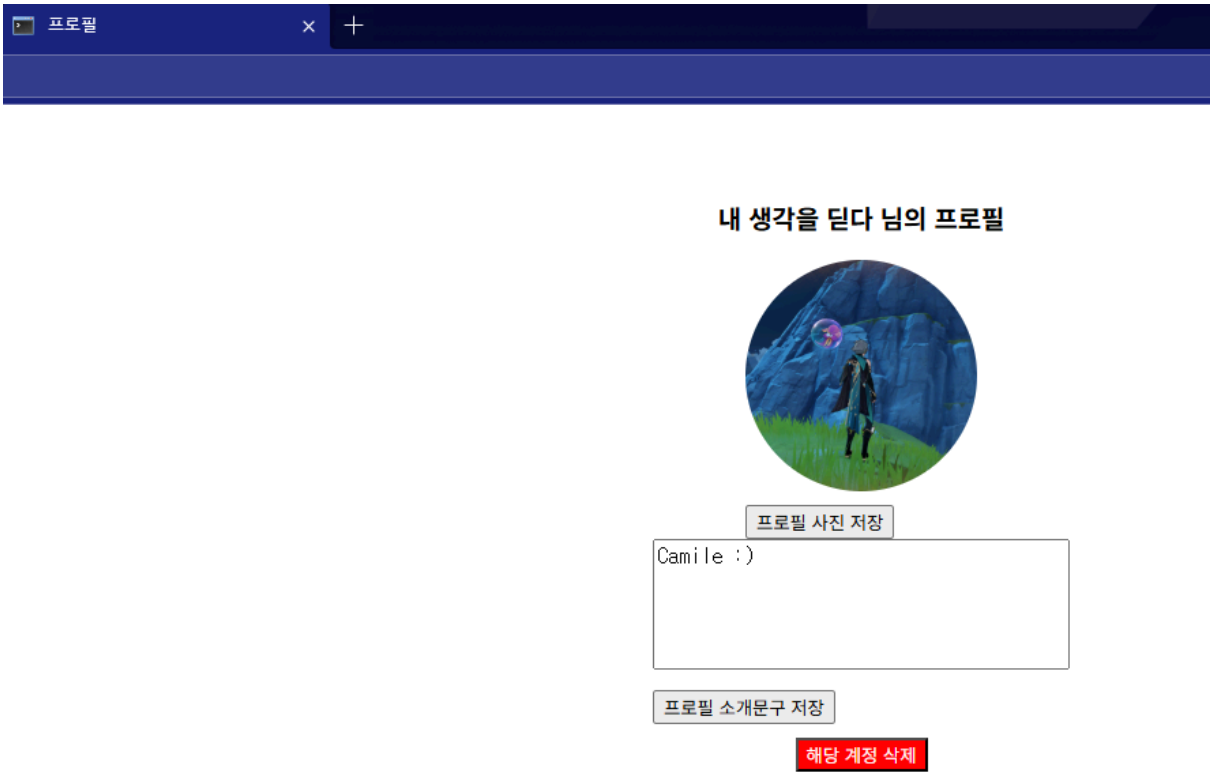
그럼 아래와 같이 내 네이버 계정으로 회원가입 및 로그인하는 버튼이 뜰 겁니다. 네이버 로그인 로고를 클릭해 줍니다.



처음 회원가입을 할 때는 '내 네이버 계정의 닉네임' 정보를 제공할지 아닐지에 대한 동의, 미동의 여부를 직접 선택하는 형태가 나올 겁니다.

'동의' 를 클릭하면 아래와 같이 '내 네이버 계정의 닉네임 님의 프로필' 이란 문구와 함께 화면의 중앙에 내 프로필 사진과 내 프로필 소개문구를 설정할 수 있는 화면이 보일 겁니다.

동그란 프로필 사진 폼을 클릭해 원하는 프로필 사진을 선택한 후 '프로필 사진 저장' 버튼을 클릭하면 내 프로필 사진이 업데이트됩니다.



또 프로필 소개문구도 원하는 문구로 작성 후 저장 버튼을 클릭하면 해당 문구가 해당 네이버 계정과 연동돼 데이터베이스에 즉각 영구적으로 저장됩니다 😊

아래의 쿼리문을 MySQL workbench 에서 입력하면 아래와 같이 회원가입을 통해 저장된 각 네이버 계정의 닉네임과 회원의 id(네이버 계정의 인증 토큰 값이 인코딩되어 있습니다.), 회원의 user_id 값, 내가 지정한 프로필 사진 파일(profile_picture) 의 상대 경로 값과 내가 작성한 프로필 문구(introduction) 의 값이 잘 저장되어 있는 걸 볼 수 있습니다 😊