

# 05 PJT

# 회원제 게시판 구현과 소셜 로그인

## 챕터의 포인트

- [도전] 회원제 게시판 구현과 소셜 로그인
- 제출

**[도전]**  
**회원제 게시판 구현과**  
**소셜 로그인**

## | 공통 요구사항 (1/2)

- 추가 구현할 회원 관리 앱의 이름은 accounts 로 지정합니다.
- .gitignore 파일을 추가하여 불필요한 파일 및 폴더는 제출하지 않도록 합니다.
- 명시된 요구사항 이외에는 자유롭게 작성해도 무관합니다.


## | 공통 요구사항 (2/2)

- 게시판 기능이 구현된 기본 코드를 제공합니다.
  - 게시글 생성, 조회, 수정, 삭제 및 댓글 생성, 삭제 기능이 구현되어 있습니다.
- 기본 코드에 회원 기능을 추가합니다.
  - 회원가입, 로그인, 로그아웃 기능을 추가합니다.
  - 로그인 여부에 따라 다른 동작이 가능한 애플리케이션을 완성합니다.
  - [도전] 기본 로그인과 소셜 로그인 두 방식으로 로그인이 가능하도록 구성합니다.

## | Model

- 수정할 모델 클래스 목록
  - A. Board
  - B. Comment
- 새로 정의할 모델 클래스 목록
  - C. User

## | A. Board

- Board 모델에 다음과 같은 정보를 추가합니다.
-  표시된 부분을 추가하고 수정한 내용을 데이터베이스에 반영합니다.

필드명	데이터 유형	역할
author	integer	외래 키(User 클래스 참조)
title	varchar(100)	게시글 제목
content	text	게시글 내용
created_at	datetime	게시글 생성일
updated_at	datetime	게시글 수정일




## | B. Comment

- Comment 모델에 다음과 같은 정보를 추가합니다.
-  표시된 부분을 추가하고 수정한 내용을 데이터베이스에 반영합니다.

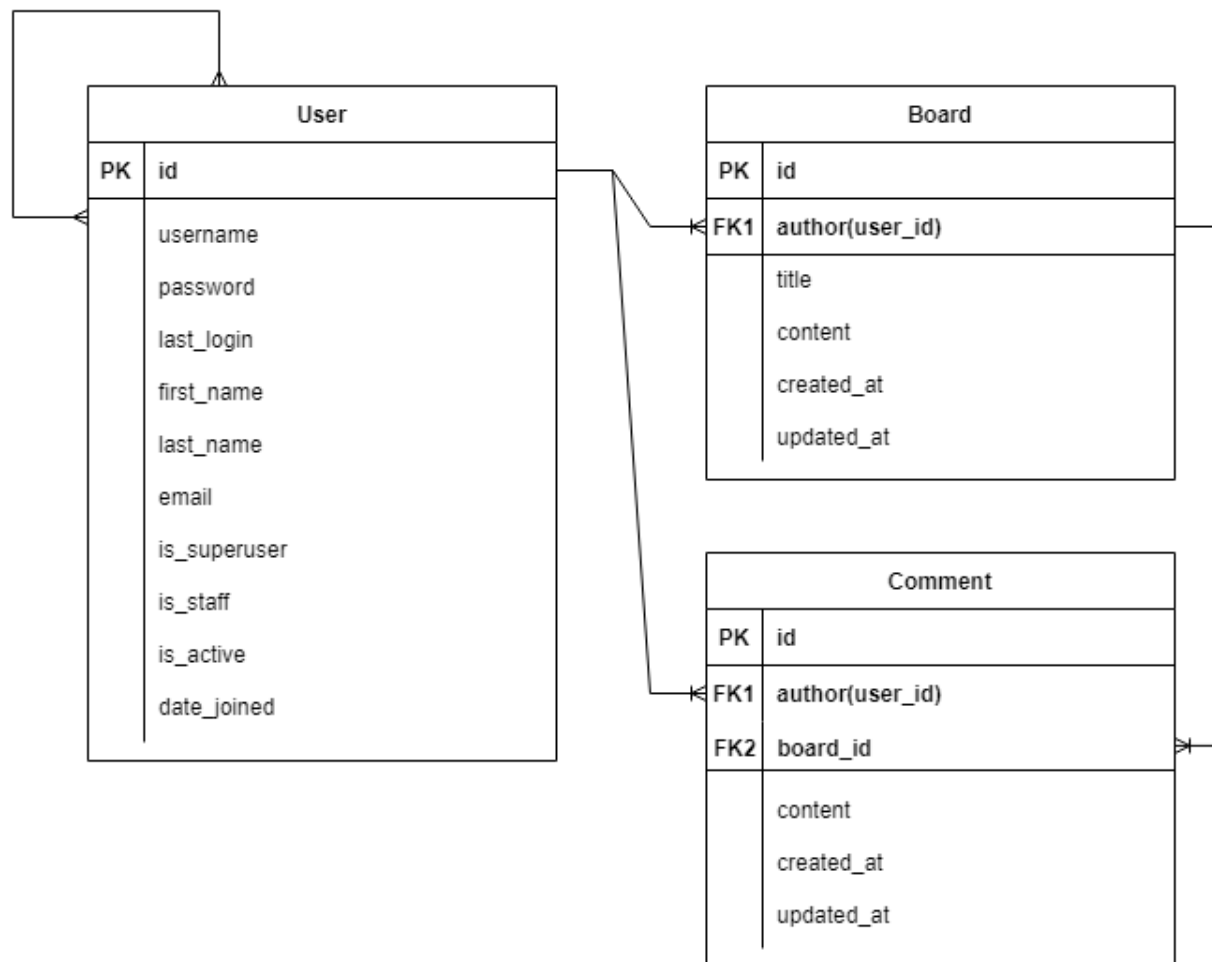
필드명	데이터 유형	역할
author	integer	외래 키(User 클래스 참조)
board	integer	외래 키(Board 클래스 참조)
content	text	댓글 내용
created_at	datetime	댓글 생성일
updated_at	datetime	댓글 수정일

## | C. User

- Django 에서 제공하는 AbstractUser 를 상속받는 User 클래스를 추가합니다.
- 수정할 모델 클래스 이름은 User 이며, 아래 필드를 추가합니다.
  -  표시된 부분을 추가하고 수정한 내용을 데이터베이스에 반영합니다.

필드명	역할
followings	M:N 중개 테이블 생성 (User 클래스 비대칭 참조)
나머지 필드들은 기존에 있던 필드들을 그대로 활용합니다.	

## [참고] ERD (Entity-Relationship Diagram)



## | URL (1/2)

- boards 앱은 다음 URL 요청에 맞는 역할을 가집니다.

URL 패턴	역할
/boards/	전체 게시글 목록 페이지 조회
/boards/create/	게시글 생성 페이지 조회 & 게시글 작성 및 저장
/boards/<board_pk>/	단일 게시글 상세 페이지 조회 & 댓글 작성 및 저장
/boards/<board_pk>/update/	단일 게시글 수정 페이지 조회 & 게시글 수정 및 저장
/boards/<board_pk>/comment/	댓글 저장
/boards/<board_pk>/comment/<comment_pk>/	댓글 삭제

## | URL (2/2)

- accounts 앱은 다음 URL 요청에 맞는 역할을 가집니다.

URL 패턴	역할
/accounts/login/	로그인 페이지 조회 & 세션 데이터 생성 및 저장(로그인)
/accounts/logout/	세션 데이터 삭제(로그아웃)
/accounts/signup/	회원 생성 페이지 조회 & 회원 데이터 작성 및 저장(회원가입)
/accounts/<user_pk>/follow/	팔로우 데이터 저장 및 팔로우 데이터 삭제

## | View (1/2)

- boards 앱은 다음 역할을 가지는 view 함수를 가집니다.

View Method	역할	허용 HTTP Method
index	전체 게시글 조회 및 index.html 렌더링	GET
create	create.html 렌더링 유효성 검증 및 게시글 저장 후 detail.html 리다이렉트	GET, POST
detail	게시글 상세 데이터 조회 및 detail.html 렌더링 게시글 삭제	GET, POST
update	수정 대상 게시글 조회 및 update.html 렌더링 유효성 검증 및 게시글 수정 후 detail.html 리다이렉트	GET, POST
comment	유효성 검증 및 댓글 저장 후 detail.html 리다이렉트	POST
comment_detail	댓글 삭제 후 detail.html 리다이렉트	POST

## | View (2/2)

- accounts 앱은 다음 역할을 가지는 view 함수를 가집니다.

View Method	역할	허용 HTTP Method
login	login.html 렌더링 및 회원 로그인	GET, POST
logout	DB 와 클라이언트 쿠키에서 세션 데이터 삭제	POST
signup	signup.html 렌더링 유효성 검증 및 회원 데이터 저장 후 index.html 리다이렉트	GET, POST
follow	회원 팔로우 데이터 저장 및 팔로우 데이터 삭제	POST

## | Admin

- 모델 Board, Comment, User를 Admin site에 등록합니다.
- Admin site에서 데이터의 생성, 조회, 수정, 삭제가 가능해야 합니다.



## | Form

- Board 모델과 Comment 모델의 데이터 검증, 저장, 에러 메시지, HTML을 모두 관리하기 위해 적절한 **ModelForm**을 사용합니다.
- User 모델의 데이터 검증, 저장, 에러 메시지, HTML을 모두 관리하기 위해 적절한 Form 과 **ModelForm** 그리고 커스텀 **ModelForm**을 사용합니다.

## | 세부 요구사항 - (1/2)

### A. 회원 관리 기능 구현

1. 회원 가입
2. 로그인 & 로그아웃

### B. Boards 앱 기능

1. 게시글 CRUD(생성, 조회, 수정, 삭제) 기능 수정
2. 댓글 생성, 삭제 기능 수정

## | 세부 요구사항 - (2/2)

C. 프로필 페이지 구현

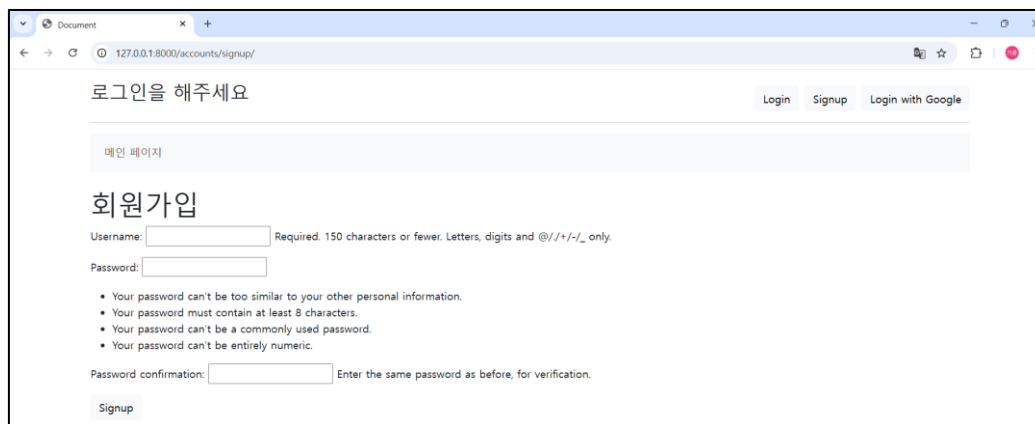
D. 유저 팔로우 기능

E. [도전] 소셜 로그인 구현하기

## A. 회원 관리 기능 구현

### 1. 회원 가입

- Django의 UserCreationForm 을 적절히 수정하여 커스텀 ModelForm을 구현합니다.
- 가입 정보를 입력할 수 있는 적절한 UI를 제공합니다.
- 회원 가입 버튼 클릭 시 입력한 회원 정보로 DB에 저장될 수 있도록 구현합니다.



Document x +

127.0.0.1:8000/accounts/signup/

로그인을 하주세요 Login Signup Login with Google

메인 페이지

### 회원가입

Username:  Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation:  Enter the same password as before, for verification.

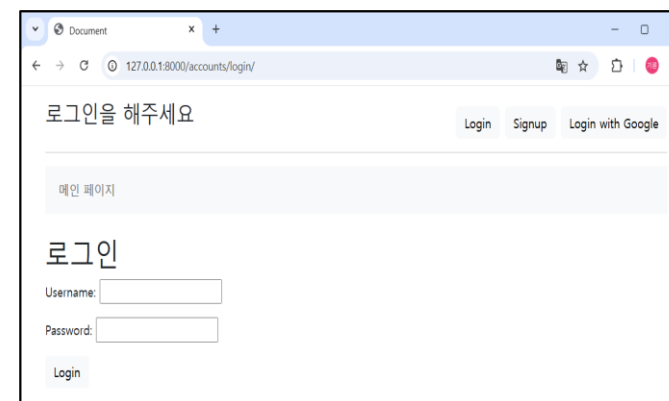
Signup

결과 예시 화면

## A. 회원 관리 기능 구현

### 2. 로그인 & 로그아웃

- 비로그인 시 로그인 버튼을, 로그인 된 사용자에게는 로그아웃 버튼을 제공합니다.
- 로그인 버튼 클릭 시 로그인을 할 수 있는 페이지를 제공합니다.
  - Django가 제공하는 로그인 Form 과 메서드를 활용합니다.
- 로그아웃 버튼 클릭 시 DB와 클라이언트 쿠키에서 세션 데이터를 삭제할 수 있도록 구현합니다.
  - Django 가 제공하는 로그아웃 메서드를 활용합니다.



Document x +  
127.0.0.1:8000/accounts/login/  
로그인을 해주세요 Login Signup Login with Google  
메인 페이지  
로그인  
Username:   
Password:   
Login

결과 예시 화면

## | B. Boards 앱 기능 (1/6)

### 1. 게시물 CRUD(생성, 조회, 수정, 삭제) 기능

- 조회 기능: 로그인 여부와 관계없이 모두 제공합니다.
- 생성 기능: 로그인 된 사용자만 생성 가능하도록 구성합니다.
- 수정, 삭제 기능
  - **본인이 작성한 게시물만** 수정 및 삭제가 가능하도록 UI와 로직을 구성합니다.
  - 수정 페이지에서 내용 작성 및 수정이 가능하도록 적절한 UI를 제공합니다.

## | B. Boards 앱 기능 (2/6)

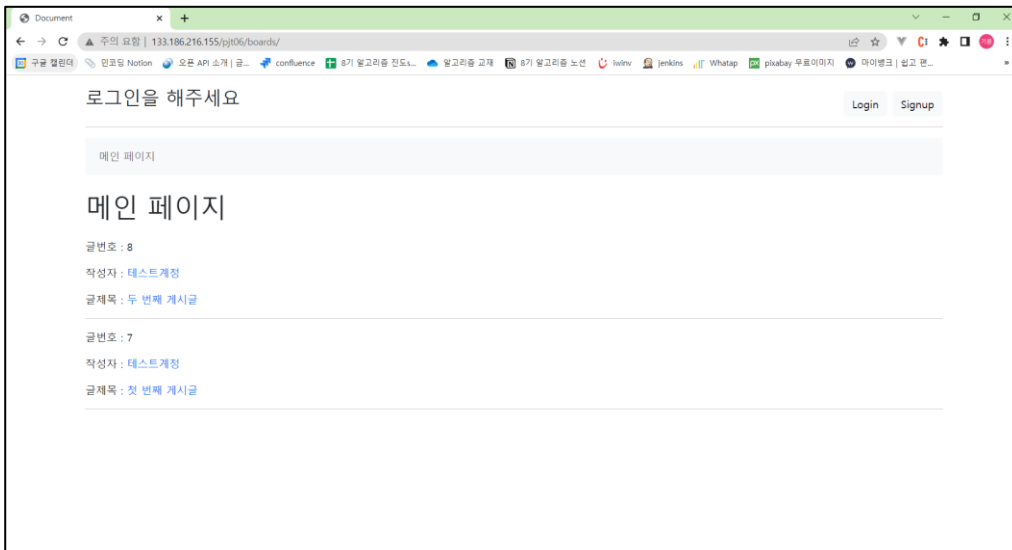
### 2. 댓글 생성, 삭제 기능

- 생성 기능
  - 인증된 사용자에게만 댓글 작성이 가능하도록 UI와 로직을 구현합니다.
- 삭제 기능
  - 본인이 작성한 댓글만 삭제가 가능하도록 UI와 로직을 구현합니다.

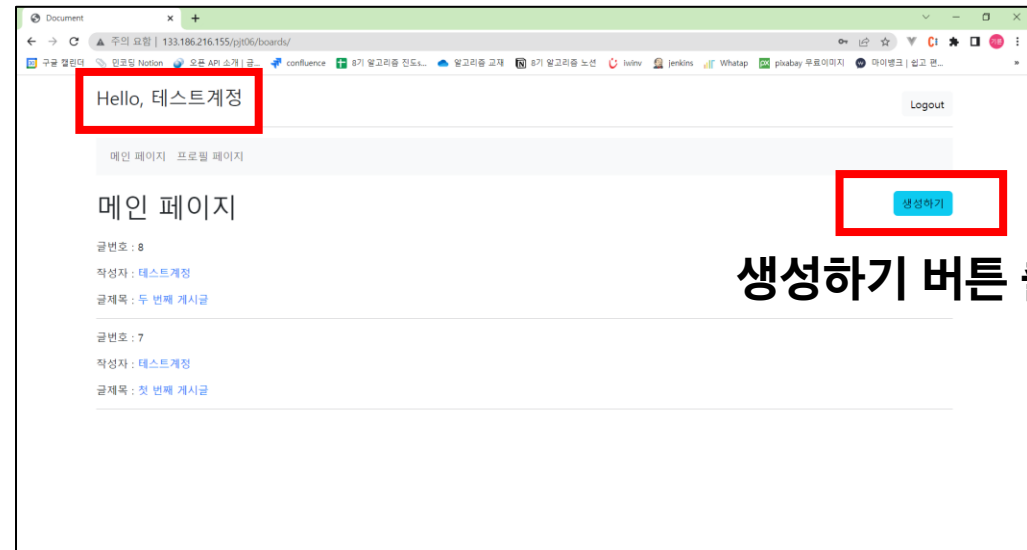
## B. Boards 앱 기능 (3/6)

- 전체 게시물 조회 페이지 예시 화면

### 로그인 전



### 로그인 후



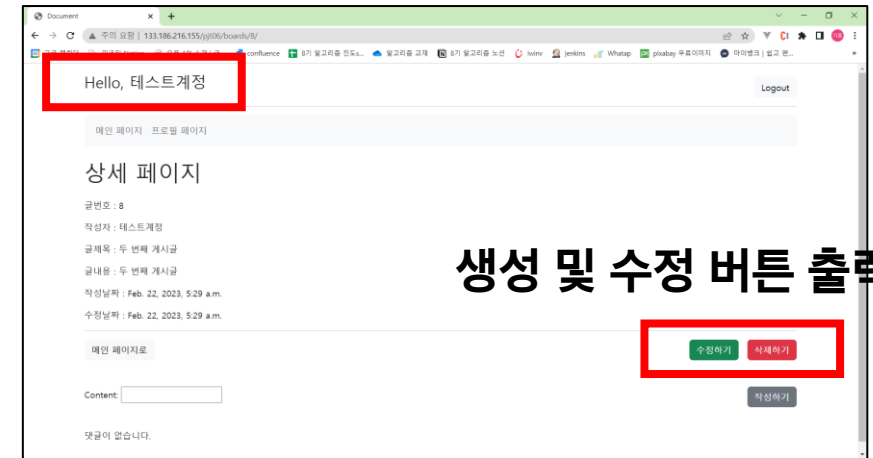
생성하기 버튼 출력



## B. Boards 앱 기능 (4/6)

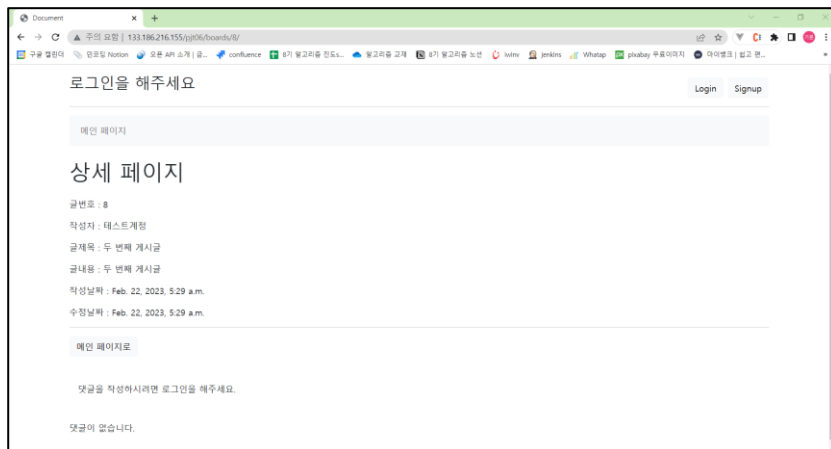
- 상세 게시물 조회 페이지 예시 화면
- 내가 작성한 댓글만 삭제 가능하도록 구현합니다.

### 로그인 후 - 내가 작성한 게시물

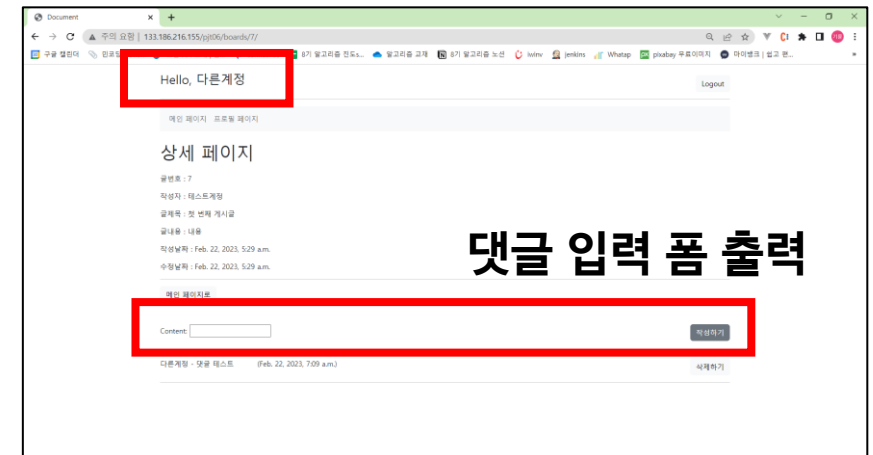


생성 및 수정 버튼 출력

### 로그인 전



### 로그인 후 - 다른 사람이 작성한 게시물



댓글 입력 폼 출력

## B. Boards 앱 기능 (5/6)

- 게시글 생성 페이지 예시 화면

Document x +

주요 요약 | 133.186.216.155/pjt06/boards/create/

구글 캘린더 민코딩 Notion 오픈 API 소개 | 금... confluence 8기 알고리즘 진도s... 알고리즘 교재 8기 알고리즘 노션 lwinv jenkins Whatap

Hello, 다른계정 Logout

메인 페이지 프로필 페이지

### 생성 페이지

Title:

Content:

생성하기

## B. Boards 앱 기능 (6/6)

- 수정 페이지 예시 화면

Document x +

주요 요약 | 133.186.216.155/pjt06/boards/8/update/

구글 캘린더 | 링크딩 Notion | 오픈 API 소개 | confluence | 8기 알고리즘 진도s... | 알고리즘 교재 | 8기 알고리즘 노션 | iwinv | jenkins | Whatap | pixabay 무료이미지 | 마이뱅크 | 업고 현...

Hello, 테스트계정 Logout

메인 페이지 | 프로필 페이지

수정 페이지

Title: 두 번째 게시물

두 번째 게시물

Content:

수정하기

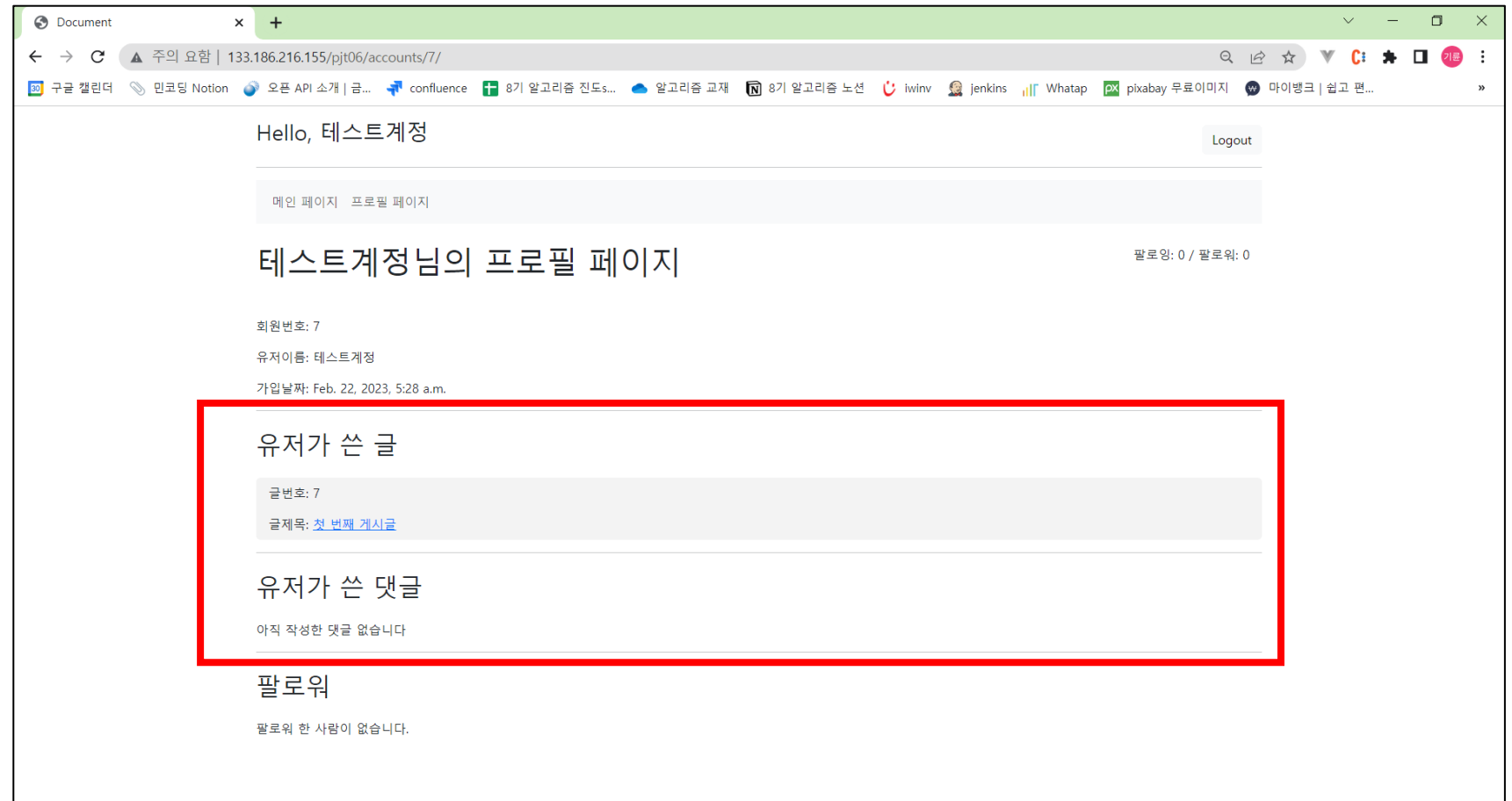
기존 내용 출력

## | C. 프로필 페이지 (1/2)

- 기본적인 회원 정보(회원번호, 회원ID, 가입날짜 등)를 출력합니다.
- 적절한 UI를 활용하여 **본인이 작성한 게시글 목록을** 제공합니다.
  - 각 게시글의 글 번호, 글제목을 출력합니다.
  - 게시글 제목 클릭 시 해당 게시글 상세 페이지로 이동합니다.
- 적절한 UI를 활용하여 **본인이 작성한 댓글 목록을** 제공합니다.
  - 각 댓글이 작성된 게시글 제목, 댓글 내용을 출력합니다.
  - 게시글 제목 클릭 시 해당 게시글 상세 페이지로 이동합니다.

## C. 프로필 페이지 (2/2)

- 결과 예시 화면

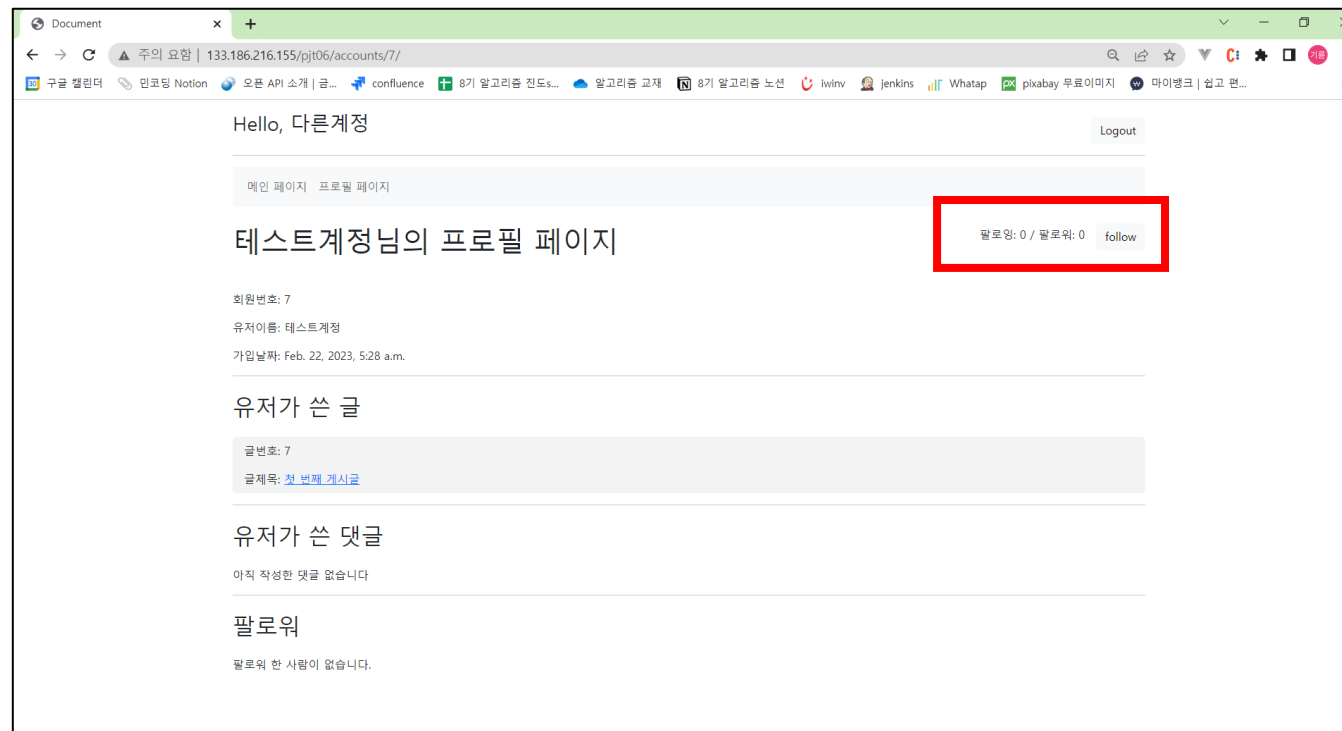


## | D. 유저 팔로우 기능 (1/2)

- 프로필 페이지에 팔로워 수와 팔로잉 수를 표시합니다.
- 프로필 페이지에 해당 사용자를 팔로우 할 수 있는 버튼을 표시합니다.
- 인증된 사용자만 다른 사용자를 팔로우 할 수 있으며, 사용자는 자기 자신을 팔로우 할 수 없습니다.
- 팔로우 버튼을 클릭하는 경우
  - DB에 팔로우 정보를 저장합니다.
  - 리다이렉션을 활용하여 해당 프로필 페이지에서 변경된 팔로우 수를 확인할 수 있도록 구현합니다.

## D. 유저 팔로우 기능 (2/2)

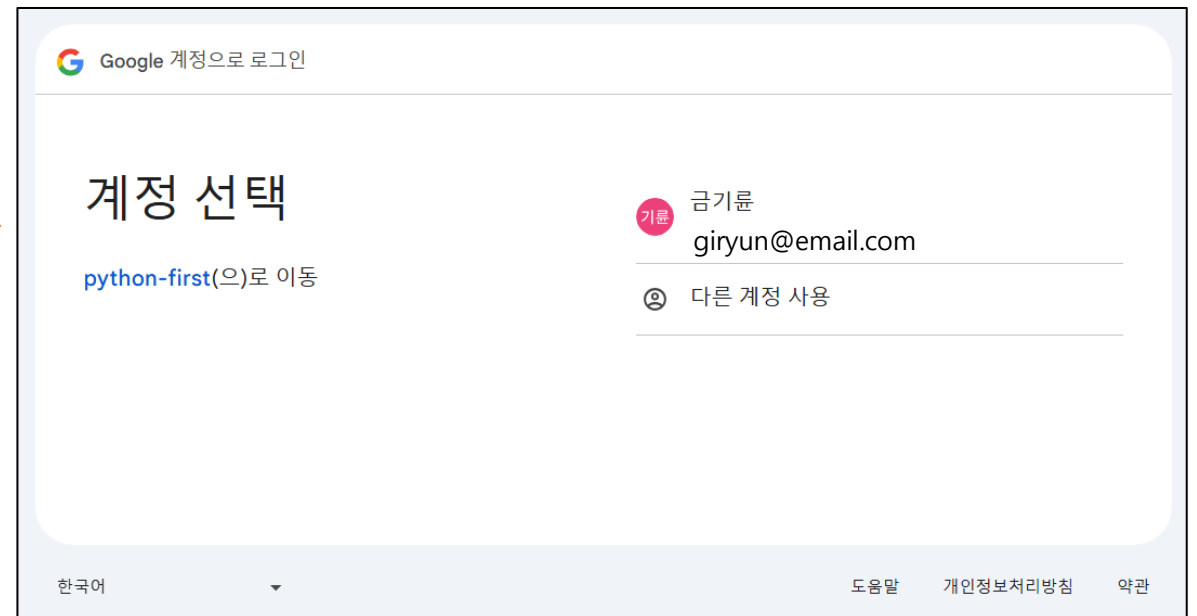
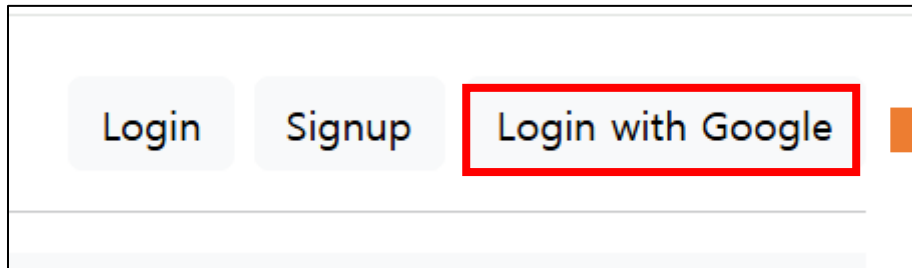
- 다른 사람의 프로필 페이지에 접근 시 팔로우를 할 수 있도록 구현합니다.



결과 예시 화면

## E. [도전] 소셜 로그인 구현하기

- Google 계정을 활용하여 로그인이 가능하도록 구현합니다.
- 버튼 클릭 시, 그림과 같이 Google 계정으로 로그인 페이지가 출력됩니다.





## E. [도전] 소셜 로그인 구현하기

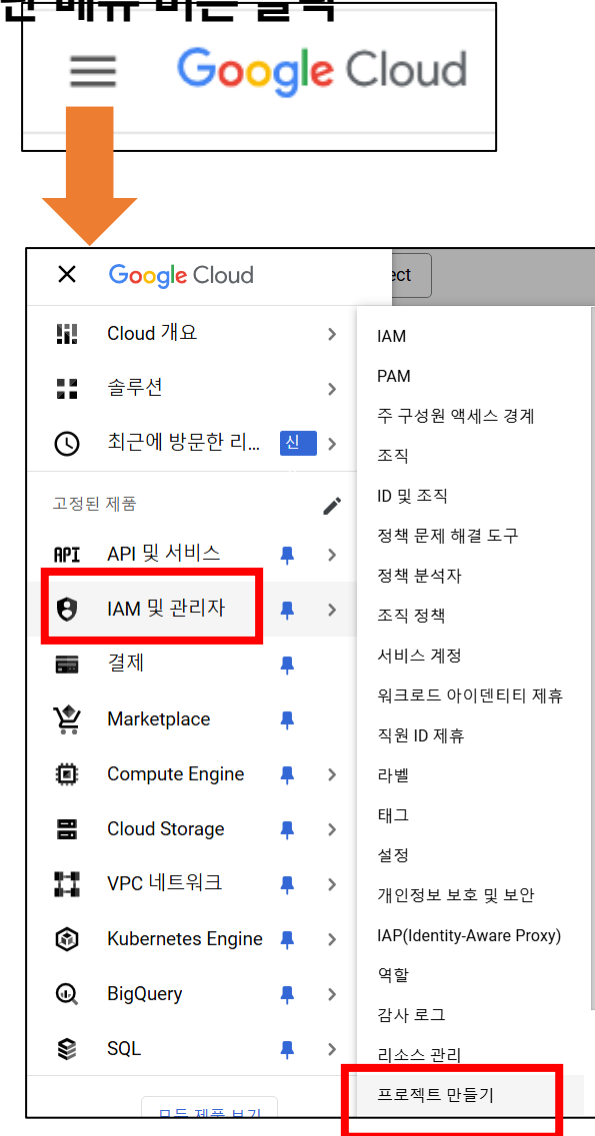
- 소셜 로그인은 인증 절차를 해당 소셜 서비스에 맡기는 것입니다.
- 소셜 로그인은 다음과 같은 과정을 통해 구현합니다.
  - Google Cloud Platform 에서 프로젝트 생성
  - OAuth 2.0 클라이언트 설정
  - 사용자가 Google 계정으로 로그인 요청
  - 인가 코드 수신 및 액세스 토큰 요청 & Django 에서 처리

## E. [도전] 소셜 로그인 구현하기

### 1. Google Cloud Platform 프로젝트 생성

- 좌측 상단 메뉴 버튼을 클릭합니다.
- IAM 및 관리자 -> 프로젝트 만들기 탭을 클릭합니다.

좌측 상단 메뉴 버튼 클릭



## E. [도전] 소셜 로그인 구현하기

### 1. Google Cloud Platform 프로젝트 생성

- [공식문서 바로가기](#)

기본값을 그대로 사용하거나  
맞춤 이름을 설정

무료 체험판  
사용자인 경우  
해당 목록은  
안 나옵니다.

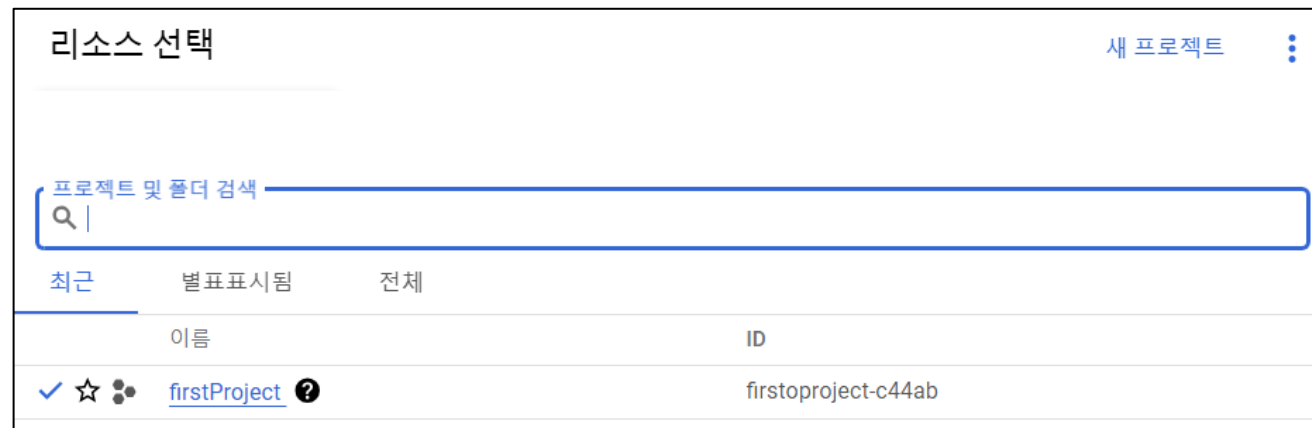
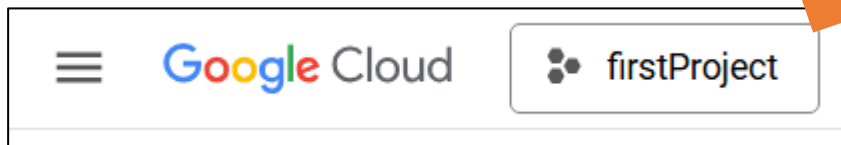
The screenshot shows the Google Cloud Project creation form with the following fields and annotations:

- 프로젝트 이름 \***: My Project 51901. An orange arrow points to this field with the text "기본값을 그대로 사용하거나 맞춤 이름을 설정".
- 프로젝트 ID**: careful-alloy-454807-a4. 나중에 변경할 수 없습니다. 수정
- 결제 계정 \***: Firebase 결제. An orange arrow points to this dropdown menu.
- 조직 \***: 조직 없음. An orange arrow points to this dropdown menu with the text "조직 없음 설정".
- 위치 \***: (empty). 찾아보기
- Buttons**: 만들기 (highlighted with a red box), 취소

## E. [도전] 소셜 로그인 구현하기

1. Google Cloud Platform 프로젝트 생성
  - 생성 완료 후 프로젝트 화면에 들어갑니다.

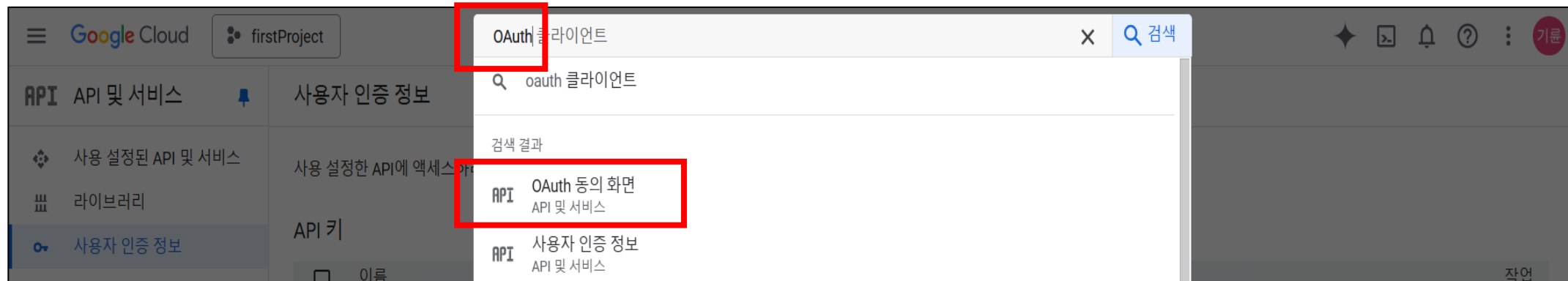
좌측 상단 네모 상자 클릭 후 이름 선택  
상자에 프로젝트명이 보이면 설정 완료



## E. [도전] 소셜 로그인 구현하기

### 2. OAuth 2.0 클라이언트 설정

- 상단 검색창에 “OAuth” 입력
- OAuth 동의 화면 탭 클릭



## E. [도전] 소셜 로그인 구현하기

### 2. OAuth 2.0 클라이언트 설정

- 시작하기 버튼 클릭



## E. [도전] 소셜 로그인 구현하기

### 2. OAuth 2.0 클라이언트 설정

- 프로젝트 4가지 구성을 완료해야 합니다.

- 앱 정보
- 대상
- 연락처 정보
- 완료 (동의하기)

1 앱 정보

앱 이름 \*  
myfirstapp  
동의를 요청하는 앱의 이름

사용자 지원 이메일 \*  
[redacted]  
사용자가 동의에 대해 문의하는 용도입니다. [자세히 알아보기](#)

다음

2 대상

☐ 내부 ②  
조직 내 사용자만 사용할 수 있습니다. 인증을 위해 앱을 제출할 필요는 없습니다. [사용자 유형 자세히 알아보기](#)

☒ 외부 ②  
Google 계정이 있는 모든 테스트 사용자가 사용할 수 있습니다. 앱이 테스트 모드로 시작되고 테스트 사용자 목록에 추가된 사용자에게만 제공됩니다. 앱을 프로덕션에 출시할 준비가 되면 앱을 인증해야 할 수도 있습니다. [사용자 유형 자세히 알아보기](#)

다음

3 연락처 정보

이메일 주소 \*  
[redacted] x

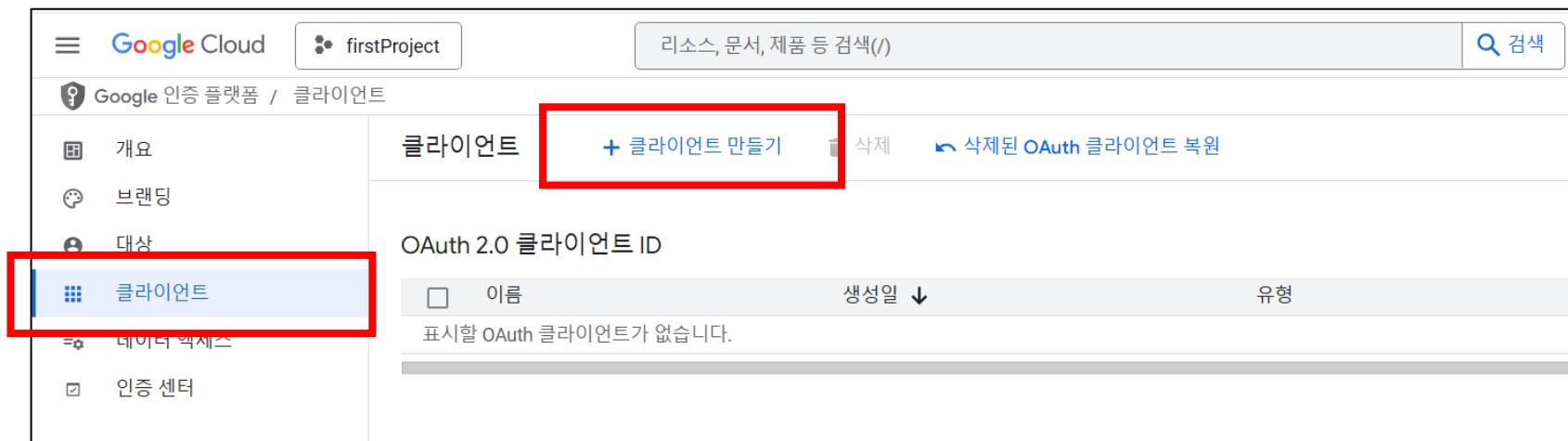
이 이메일 주소는 Google에서 프로젝트 변경사항에 대해 알림을 보내기 위한 용도입니다.

다음

## E. [도전] 소셜 로그인 구현하기

### 2. OAuth 2.0 클라이언트 설정

- 이 후 OAuth 2.0 클라이언트를 생성해줍니다.
- 프로젝트 좌측 클라이언트 탭을 클릭 한 후, 클라이언트 만들기 버튼을 클릭합니다.





## E. [도전] 소셜 로그인 구현하기

### 2. OAuth 2.0 클라이언트 설정

← OAuth 클라이언트 ID 만들기

클라이언트 ID는 Google OAuth 서버에서 단일 앱을 식별하는 데 사용됩니다. 앱이 여러 플랫폼에서 실행되는 경우 각각 자체 클라이언트 ID가 있어야 합니다. 자세한 내용은 [OAuth 2.0 설정](#)을 참조하세요. OAuth 클라이언트 유형을 [자세히 알아보세](#)

애플리케이션 유형 \*

웹 애플리케이션

Android

Chrome 확장 프로그램

iOS

TV 및 입력 제한 기기

데스크톱 앱

Universal Windows Platform(UWP)

애플리케이션 유형 :  
웹 애플리케이션 클릭

이름 \*

클라이언트1

이름은 자유롭게 지어도 됩니다.

OAuth 2.0 클라이언트의 이름입니다. 이 이름은 콘솔에서 클라이언트를 식별하는 용도로만 사용되며 최종 사용자에게 표시되지 않습니다.

## E. [도전] 소셜 로그인 구현하기

### 2. OAuth 2.0 클라이언트 설정

승인된 JavaScript 원본 ?  
브라우저 요청에 사용

URI 1 \*  Django 서버 주소 작성

+ URI 추가

승인된 리디렉션 URI ?  
웹 서버의 요청에 사용

URI 1 \*  Google 로그인 후 돌아오는 주소를 작성

+ URI 추가

#### 리디렉션 URI ?

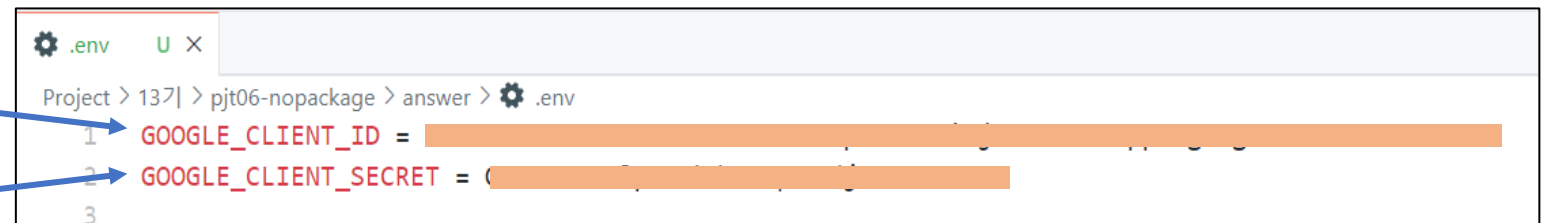
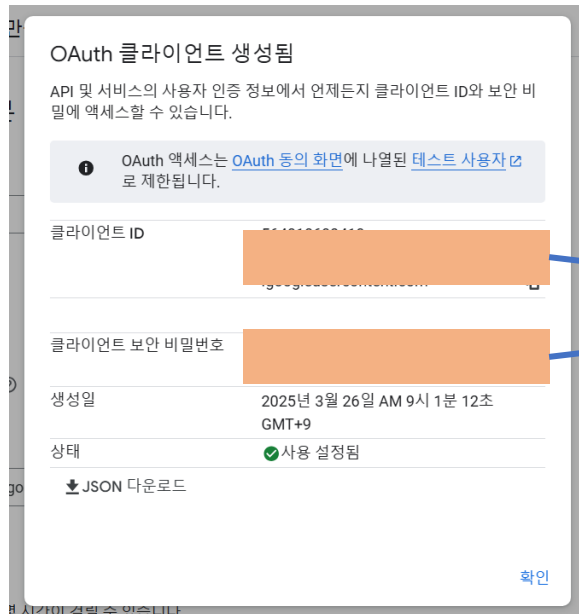
Google 에서 로그인 후  
사용자를 되돌려보낼 우리 웹사이트 주소

즉, Google 이 우리 서버에 로그인 결과를 알려줘야 하는데,  
결과를 보내 줄 주소를 의미한다.  
(작성한 URL 으로 인가 코드와 함께 요청이 들어온다.)

## E. [도전] 소셜 로그인 구현하기

### 2. OAuth 2.0 클라이언트 설정

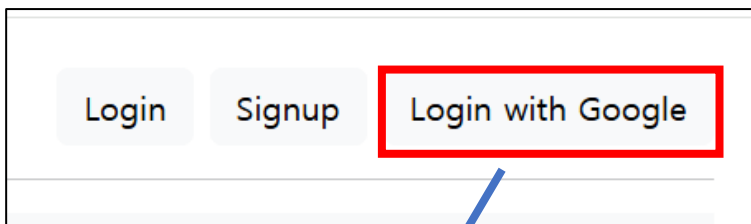
- 클라이언트 ID 와 클라이언트 보안 비밀번호를 복사하여 .env 파일에 작성합니다.
  - .env 파일은 환경 변수를 관리해주는 파일이며, 프로젝트 기본 경로에 생성합니다.
  - PPT 마지막 참고파트에 설명



## E. [도전] 소셜 로그인 구현하기

### 3. 사용자가 Google 계정으로 로그인 요청

- 요청을 보낼 수 있는 UI 를 만들고 요청을 보냅니다.



urls.py

```
path("google/login/", views.google_login, name="google-login"),
```

리디렉션 URI  
Google 클라이언트에  
작성한 주소와 동일해야 함

views.py

```
def google_login(request):
    client_id = os.getenv("GOOGLE_CLIENT_ID")
    redirect_uri = "http://127.0.0.1:8000/accounts/google/callback/"
    scope = "openid email profile"
    response_type = "code"

    google_auth_url = (
        "https://accounts.google.com/o/oauth2/v2/auth?"
        + urllib.parse.urlencode({
            "client_id": client_id,
            "redirect_uri": redirect_uri,
            "response_type": response_type,
            "scope": scope,
            "prompt": "select_account",
        })
    )

    return redirect(google_auth_url)
```

인가 코드 응답 요청

## E. [도전] 소셜 로그인 구현하기

4. 인가 코드 수신 & 액세스 토큰 요청 및 Django 에서 처리
  - 리디렉션 URI 와 매핑 되도록 URI 와 view 함수를 정의해줍니다.
  - google\_callback 함수에서는 아래 동작을 해야 합니다.
    - 인가 코드를 이용해서 Google 에 액세스 토큰 요청
    - 사용자 정보 요청
    - Django 에서 사용자 생성 또는 로그인

urls.py

```
path("google/callback/", views.google_callback, name="google-callback"),
```

## E. [도전] 소셜 로그인 구현하기

### 4. 인가 코드 수신 & 액세스 토큰 요청 및 Django 에서 처리

```
def google_callback(request):
    User = get_user_model()

    # Google 에서 준 인가 코드
    code = request.GET.get("code")
    if not code:
        print("No code provided")
        return redirect("/")

    # Google에서 액세스 토큰 요청
    access_token = get_access_token(code)

    # 사용자 정보 요청
    email, name = get_user_info(access_token)

    # 사용자 생성 또는 로그인
    user, created = User.objects.get_or_create(
        username=email,
        defaults={"first_name": name}
    )
    print(f'User: {user}')
    if created:
        print(f"User created: {user}")
    else:
        print(f"User exists: {user}")

    auth_login(request, user)

    return redirect("/boards/")
```

```
def get_access_token(code):
    token_url = "https://oauth2.googleapis.com/token"
    data = {
        "code": code,
        "client_id": os.getenv("GOOGLE_CLIENT_ID"),
        "client_secret": os.getenv("GOOGLE_CLIENT_SECRET"),
        "redirect_uri": "http://127.0.0.1:8000/accounts/google/callback/",
        "grant_type": "authorization_code",
    }
    response = requests.post(token_url, data=data)
    token_data = response.json()
    access_token = token_data.get("access_token")
    return access_token
```

```
def get_user_info(access_token):
    userinfo_url = "https://www.googleapis.com/oauth2/v3/userinfo"
    userinfo_response = requests.get(
        userinfo_url,
        headers={"Authorization": f"Bearer {access_token}"},
    )
    userinfo = userinfo_response.json()

    email = userinfo.get("email")
    name = userinfo.get("name")
    if not email:
        return redirect("/")

    return email, name
```

## 참고사항

## | 환경변수 관리

- API KEY 와 같이 외부에 노출하면 안되는 정보를 따로 관리합니다.
- django-environ 설치 - `(venv) $ pip install django-environ`
- .env 파일 작성

<주의사항> API\_KEY와 = 사이에 공백이 있으면 오류가 발생합니다.

```
1 API_KEY='<발급받은 API KEY 입력>'
```

- 발급받은 API KEY 를 따옴표로 묶어서 문자열 형태로 입력합니다.
- 수정 후 서버를 다시 시작해야 반영됩니다.



## | 환경변수 관리

- settings.py 에 아래 코드를 추가해 줍니다.

```
15 # Build paths inside the project like this: BASE_DIR / 'subdir'.
16 BASE_DIR = Path(__file__).resolve().parent.parent
17
18 √ import os
19 import environ
20
21 # 환경변수를 불러올 수 있는 상태로 설정합니다
22 env = environ.Env(DEBUG=(bool, True))
23
24 # 환경변수를 읽어올 파일을 설정합니다
25 √ env.read_env(
26     env_file=os.path.join(BASE_DIR, '.env')
27 )
28
29 # 환경변수를 읽어옵니다.
30 API_KEY = env('API_KEY')
```

## | 환경변수 관리

- settings 에 등록된 환경변수는 views.py 에서 다음과 같이 사용합니다.

```
from django.conf import settings  
  
API_KEY = settings.API_KEY
```

- .gitignore 파일에 .env 를 추가하여 API KEY 가 외부에 노출되지 않도록 설정합니다.

# 제출

## | 제출 시 주의사항

- 제출기한은 금일 18시까지입니다. 제출기한을 지켜 주시기 바랍니다.
- 반드시 README.md 파일에 단계별로 구현 과정 중 학습한 내용, 어려웠던 부분, 새로 배운 것들 및 느낀 점 등을 상세히 기록하여 제출합니다.
  - 단순히 완성된 코드만을 나열하지 않습니다.
- 위에 명시된 요구사항은 최소 조건이며, 추가 개발을 자유롭게 진행할 수 있습니다.
- <https://lab.ssafy.com/> 에 프로젝트를 생성하고 제출합니다.
  - 프로젝트 이름은 '프로젝트 번호 + pjt' 로 지정합니다. (ex. 05\_pjt)
- 반드시 각 반 담당 교수님을 Maintainer 로 설정해야 합니다.