|  |
| --- |
|  |



|  |
| --- |
| **Lighthouse** |
| **포팅 메뉴얼** |
|  |
|  |
|  |
|  |
|  |

목차

[I. 개요 2](#_Toc143176943)

[1. 프로젝트 사용 도구 2](#_Toc143176944)

[2. 개발환경 2](#_Toc143176945)

[3. 외부 서비스 2](#_Toc143176946)

[4. 환경변수 처리한 핵심 키들 2](#_Toc143176947)

[II. 빌드 3](#_Toc143176948)

[1. 환경변수 형태 3](#_Toc143176949)

[2. 빌드하기 4](#_Toc143176950)

[3. 배포하기 4](#_Toc143176951)

[4. 서비스 이용 방법 6](#_Toc143176952)

[가) Google 클라우드 플랫폼(GCP) 6](#_Toc143176953)

[나) S3와 CloudFront 연동 6](#_Toc143176954)

[III. 시연 시나리오 7](#_Toc143176955)

[1. 메인페이지 7](#_Toc143176956)

# 개요

## 프로젝트 사용 도구

이슈 관리 : JIRA

형상 관리 : Gitlab

커뮤니케이션 : Notion, Mattermost

디자인 : Figma

UCC : Movavi

CI/CD : Jenkins, Docker

## 개발환경

VS Code : 1.81.1,

IntelliJ : IDEA 2023.1.3 (Runtime version: 17.0.7+10-b829.16 amd64, VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o)

JVM : 1.8.0\_372

Node.js : 18.17.0

SERVER : AWS EC2 Ubuntu 20.04 LTS

DB : MariaDB (azure 10.3.23-MariaDB), MongoDB

Kafka : 3.1.0

Docker : 24.0.5

## 외부 서비스

Google OAuth : application.properties 에 해당 내용 있음

AWS S3 : application.properties 에 해당 내용 있음

AWS CloudFront : .env 에 해당 내용 있음.

## 환경변수 처리한 핵심 키들

React: .env

Spring: 서버 내 ~/config/.env 파일에 존재

# 빌드

## 환경변수 형태

**.env:**

// MariaDB

DATASOURCE\_URL={DB 주소}

DATASOURCE\_USERNAME= {DB 호스트 유저 이름}

DATASOURCE\_PASSWORD= {{DB 비밀번호}

// MondoDB

MONGODB\_URI= {DB 주소}

SALT= {JWT Salt}

// S3

AWS\_ACCESSKEY= {S3 접근 키}

AWS\_SECRETKEY= {S3 암호 키}

// Google Oauth

GOOGLE\_CLIENT\_ID= {구글 클라이언트 아이디}

GOOGLE\_CLIENT\_SECRET= {구글 시크릿 키}

REDIRECT\_URI={리다이렉트 주소}

// React

REACT\_APP\_API\_URL= {백엔드 서버 주소}

REACT\_APP\_GOOGLE\_CLIENT\_ID={구글 클라이언트 아이디}

KAFKA\_BROKER= {카프카 서버 주소}

SSL\_PASSWORD= {SSL 비밀번호}

## 빌드하기

1) Front: React

npm i

npm start

2) Back: Spring Boot

Gradle 실행

Bootjar 실행

## 배포하기

**Jenkinsfile**

pipeline {

agent any

environment {

VERSION = "latest"

DOCKERHUB\_REPOSITORY\_BACK = {도커허브이름/백엔드저장소이름}

DOCKERHUB\_REPOSITORY\_FRONT = {도커허브이름/프론트저장소이름}

DOCKERHUB\_CREDENTIAL = credentials('dockerhub-imsong')

CONTAINER\_NAME\_BACK = {백엔드 컨테이너 이름}

CONTAINER\_NAME\_FRONT = {프론트 컨테이너 이름}

SSH\_CONNECTION = {서버 주소}

ENV\_DIR = {서버 내 환경변수 저장 위치}

PORT\_BACK = {백엔드 포트}

PORT\_FRONT = "{프론트 포트}"

}

stages {

stage('Build Backend') {

steps {

dir('backend/lighthouse') {

sh "chmod +x gradlew"

sh "./gradlew clean compileJava bootJar"

}

}

}

stage("Build Images") {

steps {

sh "docker compose build"

}

}

stage('Push Images'){

steps {

sh "echo $DOCKERHUB\_CREDENTIAL\_PSW | docker login -u $DOCKERHUB\_CREDENTIAL\_USR --password-stdin"

sh "docker push $DOCKERHUB\_REPOSITORY\_BACK:$VERSION"

sh "docker push $DOCKERHUB\_REPOSITORY\_FRONT:$VERSION"

}

}

stage('Deploy Backend Server') {

steps {

sshagent(credentials: ['ec2']) {

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker rm -f $CONTAINER\_NAME\_BACK'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker rmi -f $DOCKERHUB\_REPOSITORY\_BACK:$VERSION'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker pull $DOCKERHUB\_REPOSITORY\_BACK:$VERSION'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'echo y | docker image prune'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker images'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker run -d --name $CONTAINER\_NAME\_BACK --env-file $ENV\_DIR -p $PORT\_BACK:8080 $DOCKERHUB\_REPOSITORY\_BACK:$VERSION'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker ps'"

}

}

}

stage('Deploy Frontend Server') {

steps {

sshagent(credentials: ['ec2']) {

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker rm -f $CONTAINER\_NAME\_FRONT'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker rmi -f $DOCKERHUB\_REPOSITORY\_FRONT:$VERSION'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker pull $DOCKERHUB\_REPOSITORY\_FRONT:$VERSION'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'echo y | docker image prune'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker images'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker run -d --name $CONTAINER\_NAME\_FRONT --env-file $ENV\_DIR -p $PORT\_FRONT:3000 $DOCKERHUB\_REPOSITORY\_FRONT:$VERSION'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker ps'"

}

}

}

}

}

## 서비스 이용 방법

### Google 클라우드 플랫폼(GCP)

1. console.cloud.google.com에 접속 → 새 프로젝트 생성

2. OAuth 동의 화면에서 User Type 외부로 설정

3. 앱 정보 등록(앱 이름, 이메일)

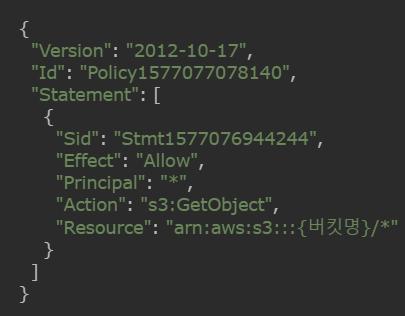
4. 사용자 인증 정보 → 사용자 인증 정보 만들기 → OAuth 클라이언트 ID

5. 승인된 자바스크립트 원본에 localhost, localhost:3000, 프론트 서버 주소 추가

6. 승인된 리다이렉션 URI에 local callback API 주소와 백엔드 서버 주소 추가

### S3와 CloudFront 연동

1. AWS 회원가입
2. IAM 생성하기
   1. 사용자 그룹 - 그룹 생성 - AmazonEC2FullAccess 권한 정책 연결 – 그룹 생성
   2. 사용자 – 사용자 생성 – 그룹에 사용자 추가 – 사용자 생성
   3. 사용자 정보 – 보안 자격 증명 – 액세스 키 만들기 – 기타 - Access Key 및 Secret Key 저장
3. S3 버킷 생성
   1. 퍼블릭 액세스 차단을 비활성화
   2. 버킷 정책 작성



1. 스프링 AWS 설정
   1. 의존성 추가 <https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-aws/2.2.1.RELEASE>
   2. AccessKey, Secret Key환경변수 설정
2. CloudFront 배포 생성
   1. 원본 도메인 선택 (Amazon S3) – 배포 생성
3. 생성된 도메인 이름으로 /frontend/.env 파일 내 REACT\_APP\_CLOUDFRONT\_DOMAIN\_URL 업데이트

### 카프카 설정

**./kafka/docker-compose.yml**

version: "3.8"

networks:

kafka-net:

driver: bridge

services:

zookeeper:

image: bitnami/zookeeper:3.7

networks:

- kafka-net

ports:

- '2181:2181'

environment:

- ALLOW\_ANONYMOUS\_LOGIN=yes

kafka:

image: bitnami/kafka:3

networks:

- kafka-net

ports:

- '9093:9093'

environment:

- KAFKA\_CFG\_ZOOKEEPER\_CONNECT=zookeeper:2181

- ALLOW\_PLAINTEXT\_LISTENER=yes

- KAFKA\_CFG\_LISTENER\_SECURITY\_PROTOCOL\_MAP=CLIENT:PLAINTEXT,EXTERNAL:PLAINTEXT

- KAFKA\_CFG\_LISTENERS=CLIENT://:9092,EXTERNAL://:9093

- KAFKA\_CFG\_ADVERTISED\_LISTENERS=CLIENT://kafka:9092,EXTERNAL://i9a409.p.ssafy.io:9093

- KAFKA\_CFG\_INTER\_BROKER\_LISTENER\_NAME=CLIENT

depends\_on:

- zookeeper

**명령어**

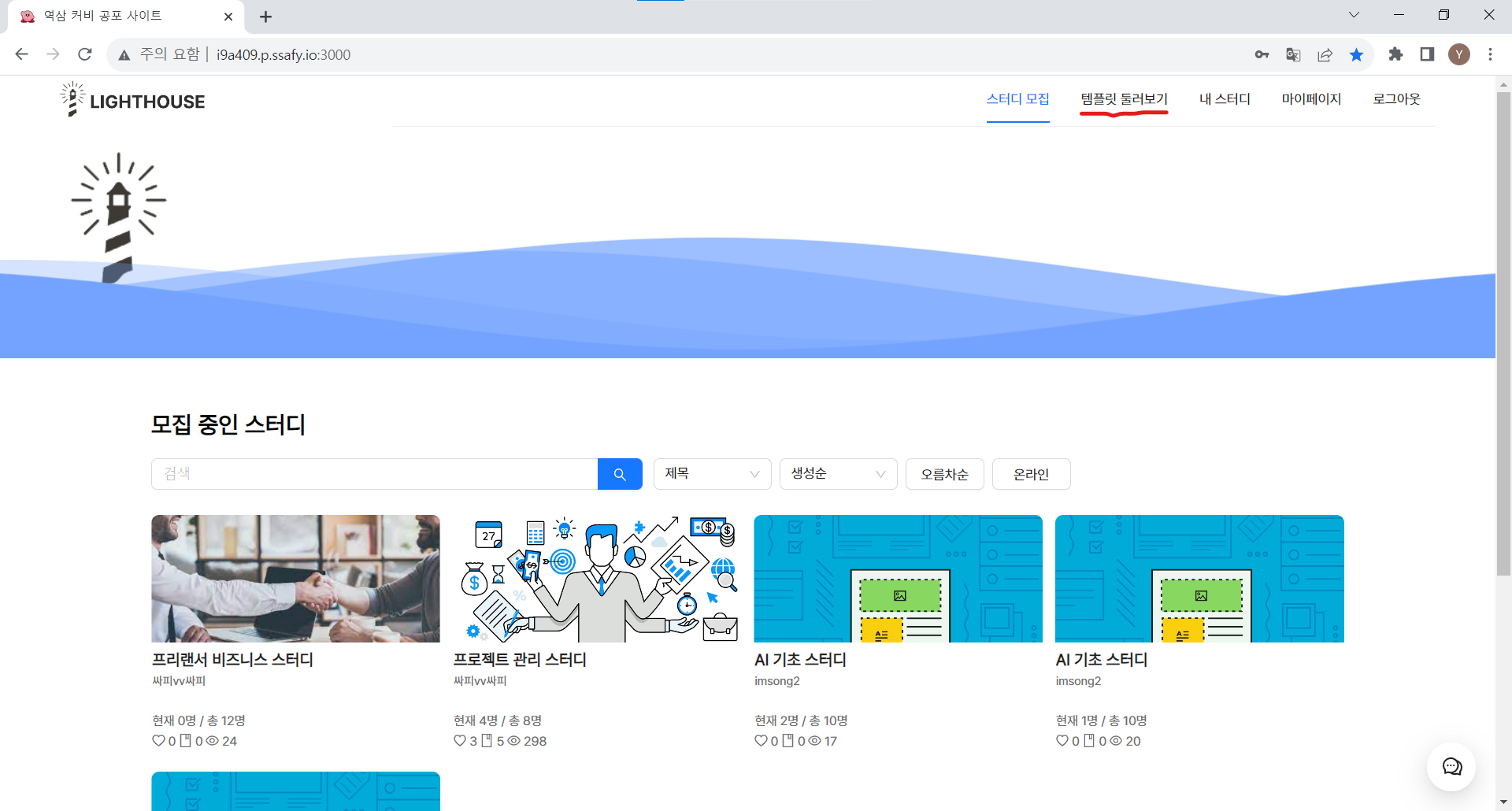
docker compose exec kafka kafka-topics --create --topic kafka-chat --bootstrap-server kafka:9092 --replication-factor 1 --partitions 1

# 시연 시나리오

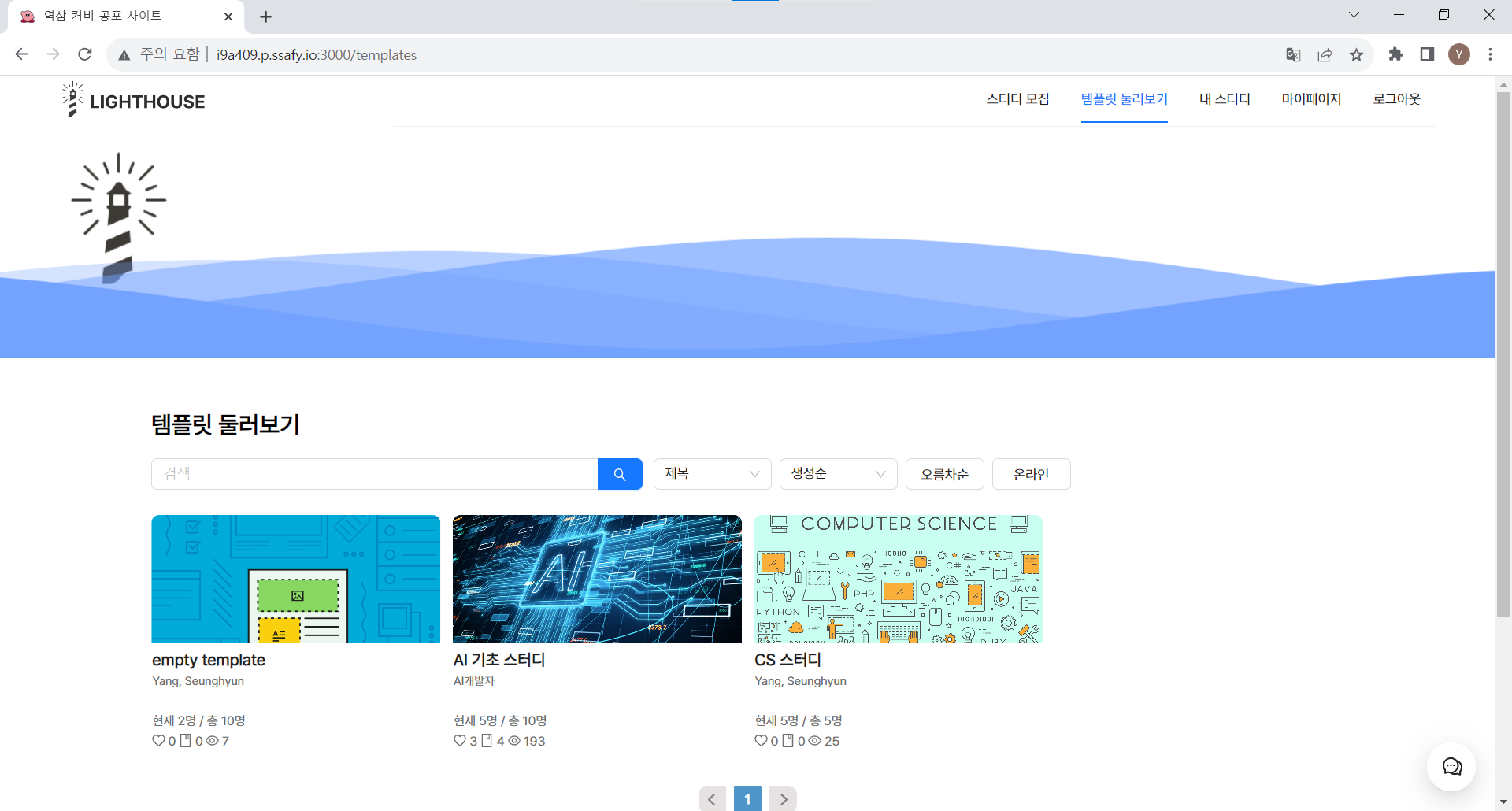
## 시나리오 1 (스터디장)

## 스터디 모집 공고 생성

템플릿 둘러보기 페이지 이동



알맞은 템플릿 찾기



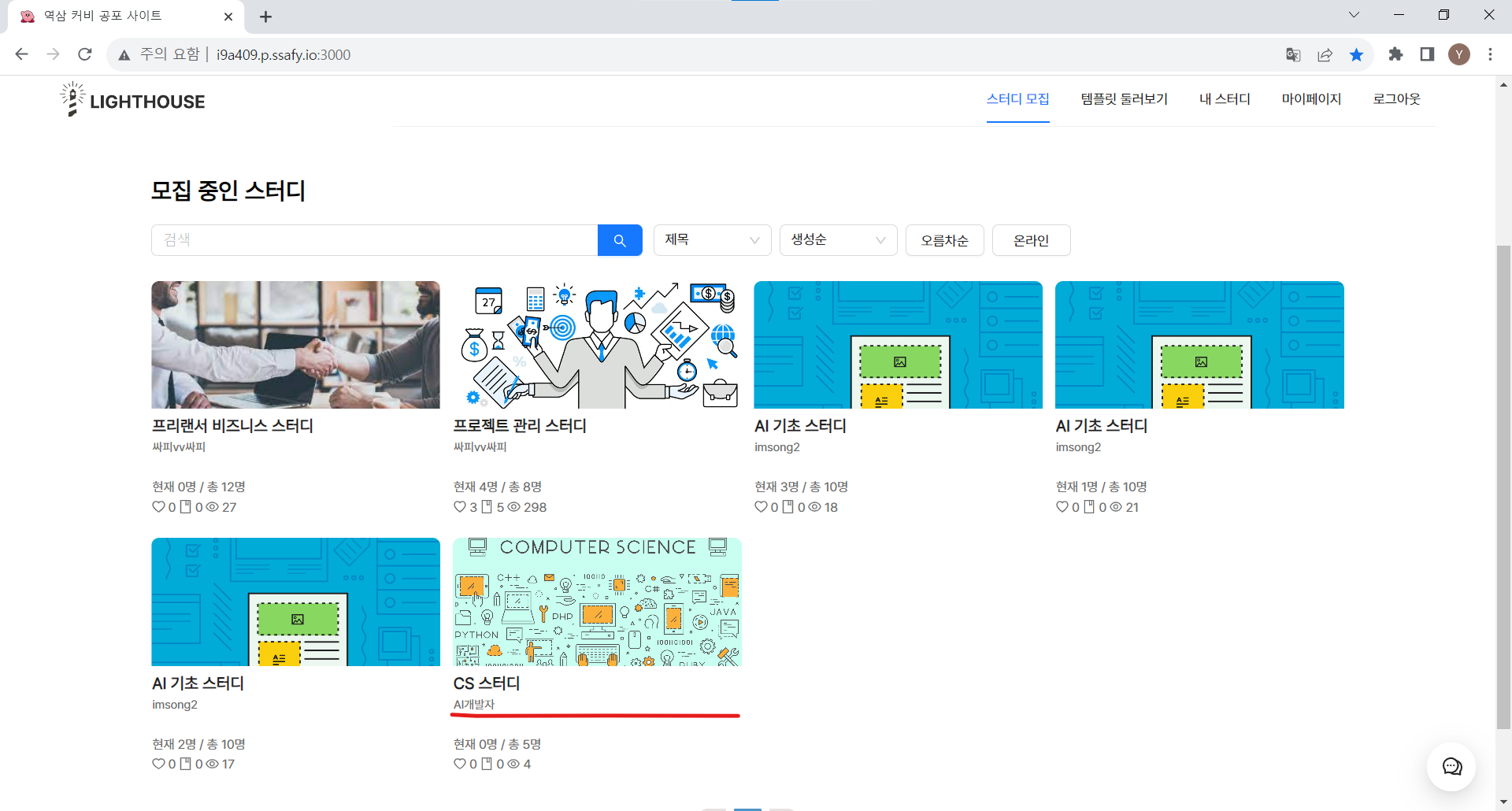
템플릿 상세보기 페이지



템플릿 사용하기 버튼 클릭 후 템플릿 수정 페이지 이동

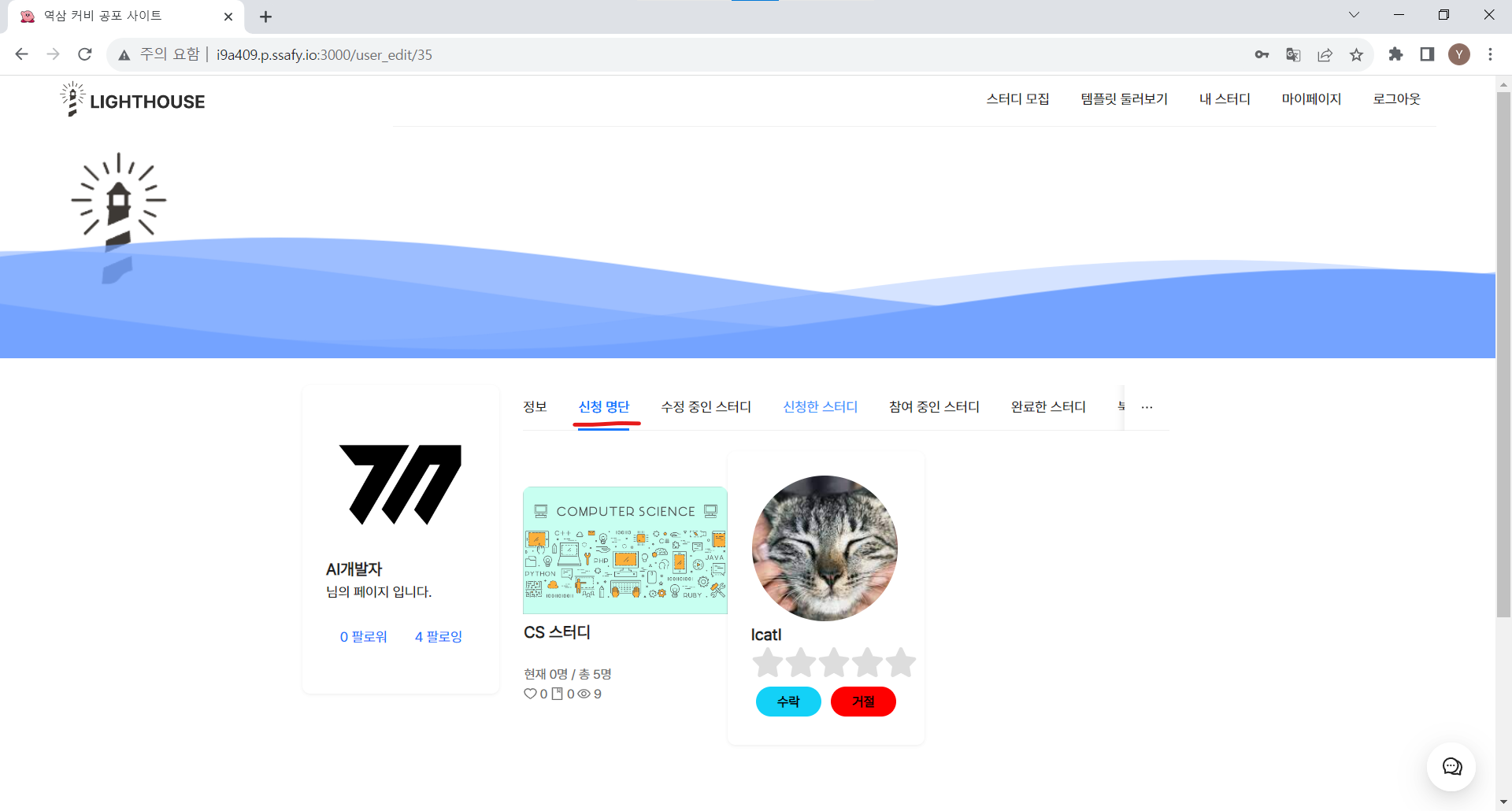


모집 공고 업로드

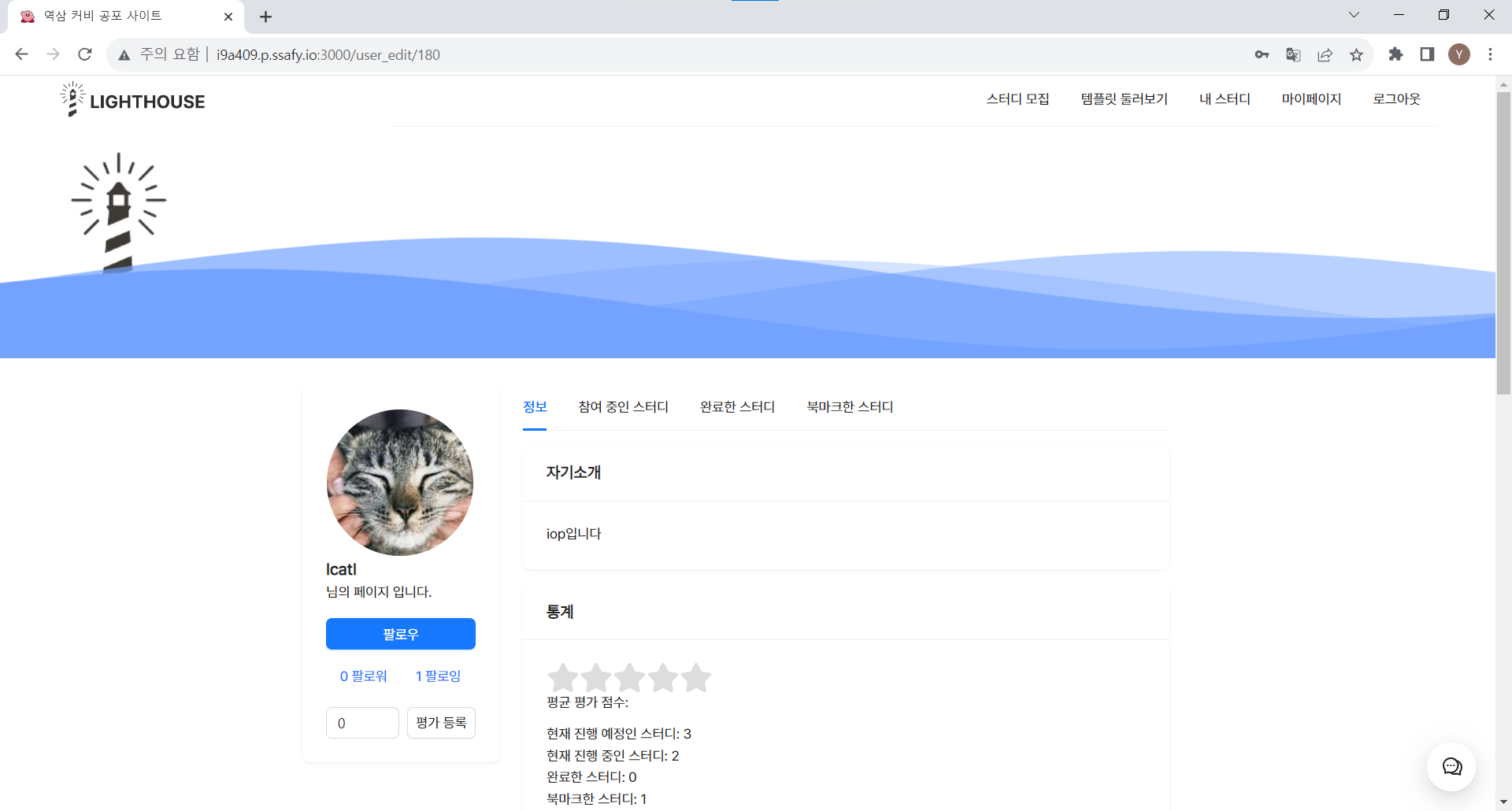


## 스터디 모집 과정

### 마이페이지의 신청 명단 조회



### 신청한 사용자 프로필 조회



### 신청 수락 및 스터디 시작



## 스터디 진행 및 마무리

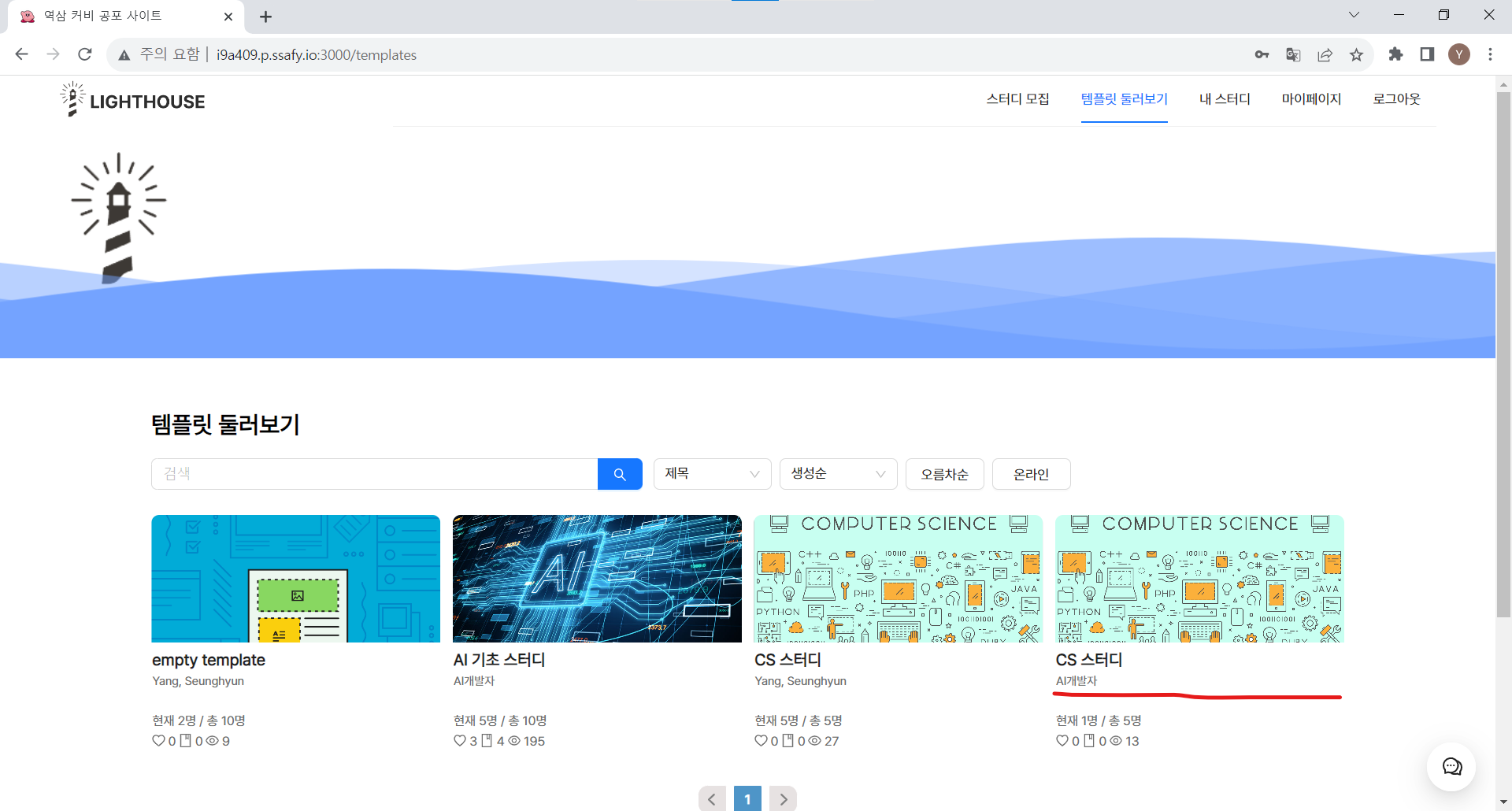
### 1. 스터디 종료



### 2. 스터디 공유



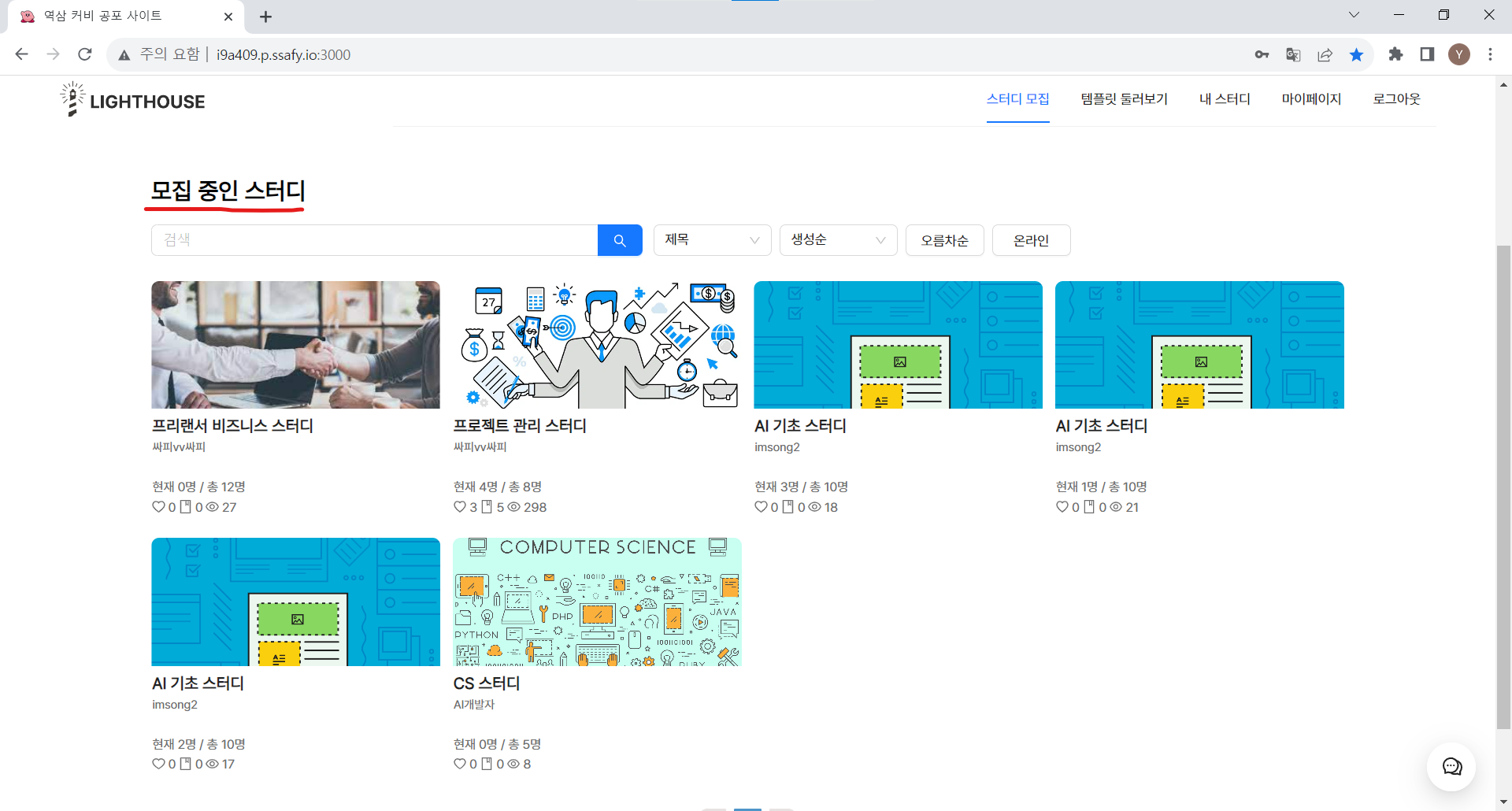
### 3. 템플릿 리스트에 추가됨



## 시나리오 2 (스터디원)

## 스터디 신청

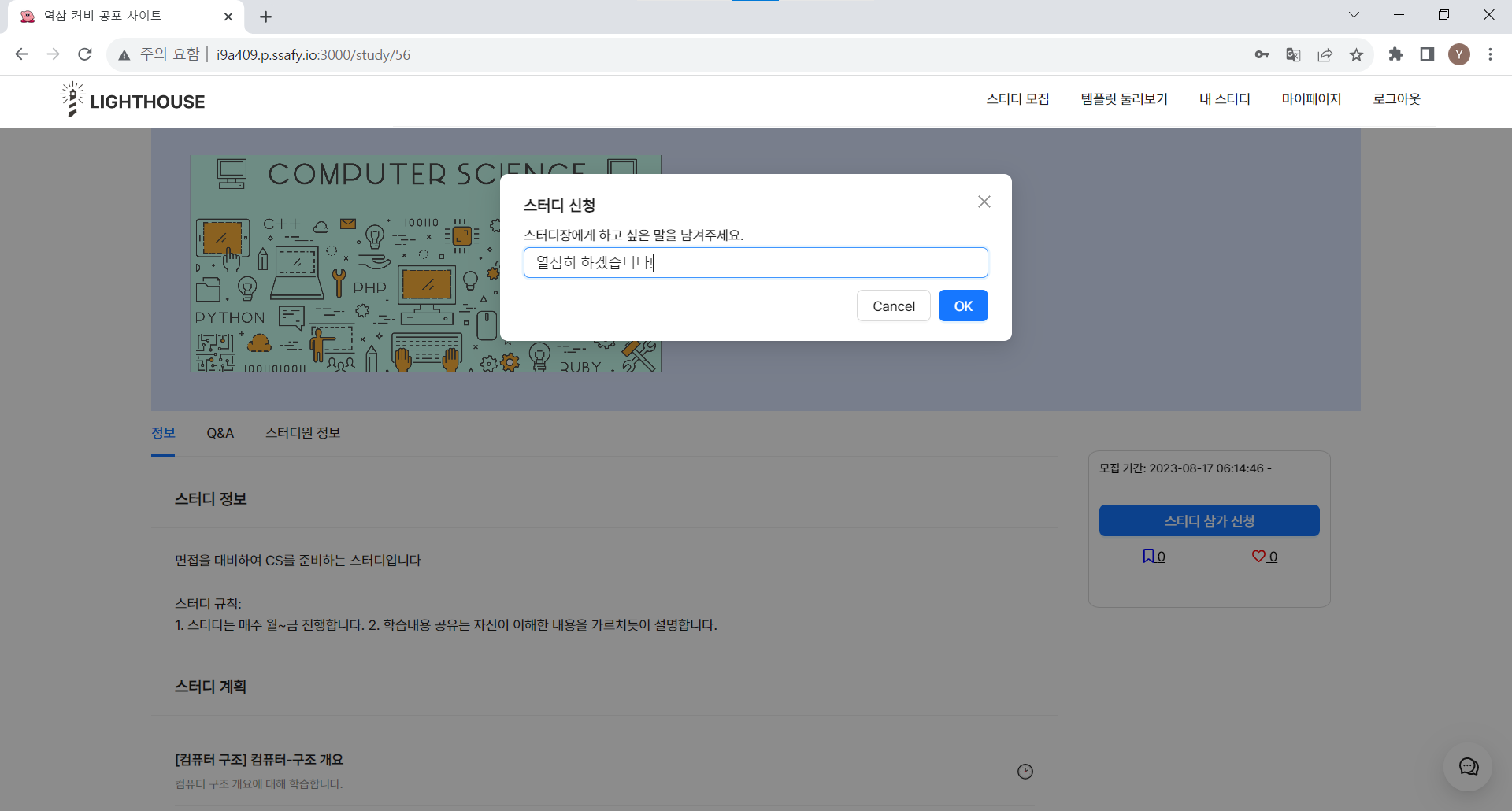
### 스터디 모집 리스트 조회

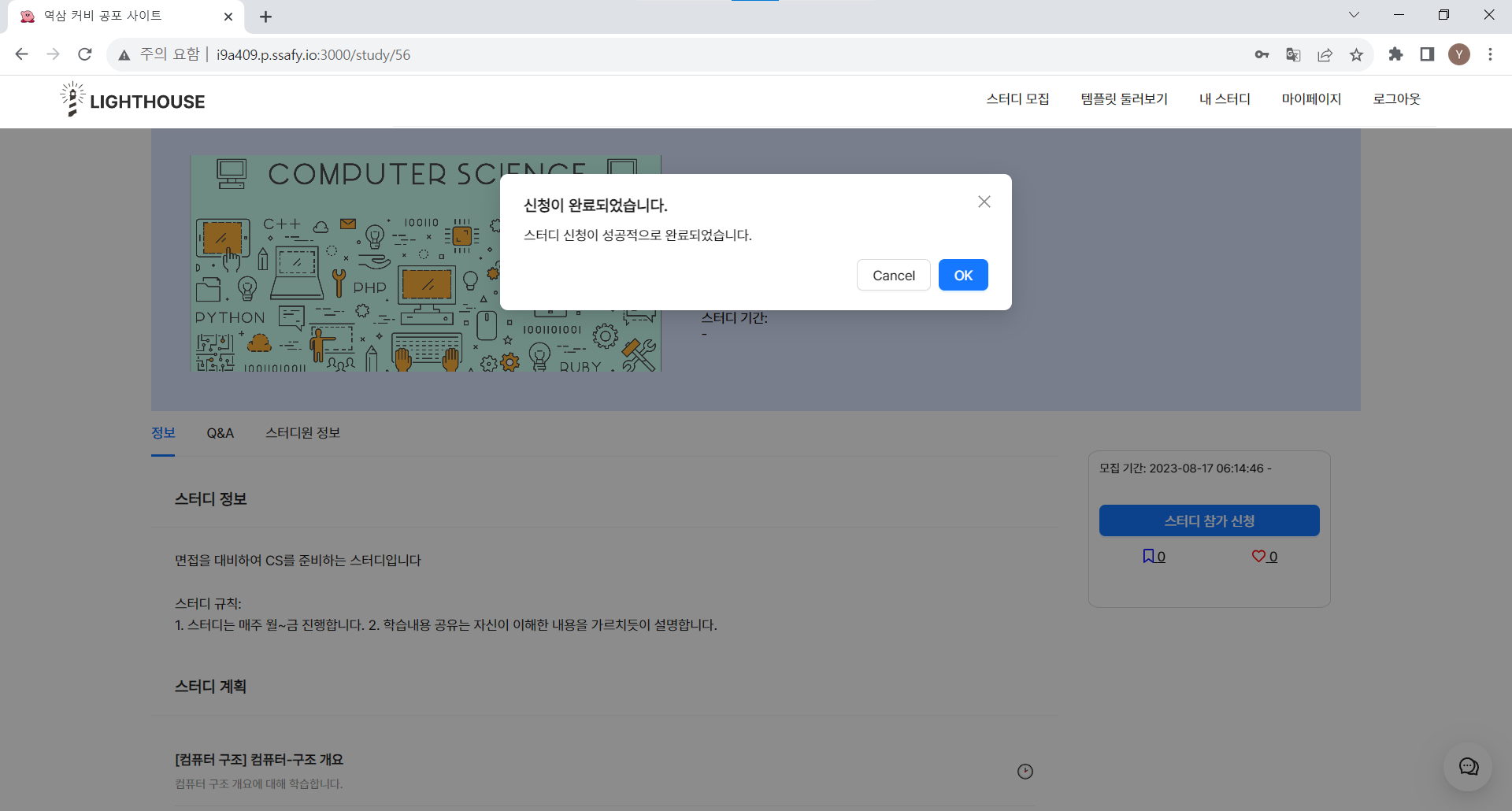


### 스터디 상세 조회



### 스터디장에게 메시지 입력 후 신청





## 스터디 진행

### 스터디 진행 페이지 조회



### 채팅 조회

