PhishingGuard Chat

Team 7. BlackList

김민형 김수민 김원준 오준석
유주현 이재홍 주혜련

Detects and alerts malicious URLs

Contents

프로젝트 개요

프로젝트 주제 및 선정 배경 기획 의도	악성 URL 피싱 공격의 급증에 따라 이를 예방하기 위해 AI 기반 악성 URL 탐지 및 대안 웹 사이트 안내 챗봇 시스템 기획
프로젝트 내용	입력된 URL의 피싱 여부를 분석하고 악성일 경우 대안 사이트 및 AI 보안 어시스턴트 구현
프로젝트 구조	사용자 입력 → AI 어시스턴트 URL 분석 → 블랙리스트 확인 → ML 모델 예측 → AI 어시스턴트가 대안 사이트 제공하는 답변 생성
활용 방안 및 기대 효과	1. 피싱 사고 예방과 정보보안 전문 지식 접근성 향상 2. 고객 상담, 보안 교육, 내부 자가 진단 등

프로젝트 팀 구성 및 역할

훈련생	구현 팀	기능 구현 내용
김민형	AI 어시스턴트	Responses API 기반 통합 구조 구현, File Search 구현, 전체 코드 병합, 대본 작성
오준석		Responses API 기반 통합 구조 구현, File Search 구현, PPT 제작
김원준	머신러닝 모델	피싱 탐지 모델 구축, Streamlit을 활용한 UI 구성, 블랙리스트 저장 구현, 대본 작성
이재홍		피싱 탐지 모델 구축, Streamlit을 활용한 UI 구성, 블랙리스트 저장 구현, 대본 작성
유주현		피싱 탐지 모델 구축, Streamlit을 활용한 UI 구성, PPT 제작, 시연 영상 제작
김수민	Function Call	Function Tool 설계 및 연결, PPT 제작
주혜련		Function Tool 설계 및 연결, PPT 제작

프로젝트 수행 절차 및 방법

구분	기간	활동	비고
사전 기획 데이터 수집	**7/1(화)**	- 기획서 작성, 팀 역할 분담, 일정 수립 - 악성 URL 및 정상 URL 수집, 데이터 전처리 - 모델 개별 학습 및 비교	기획 방향 수립 데이터 : Kaggle - Malicious URLs dataset URL feature 기반 전처리
데이터 전처리 모델링 기능 구현	**7/2(수) ~ 7/3(목)**	- 모델 통일 및 정확도 조정 - Responses API 통합 구조 설계 - Function tool 구현 - 블랙리스트 저장 기능 구축	성능 기준, 머신러닝 실습 적용 에이전트 기반 구조 설계 자동화 흐름 구현
서비스 구축	**7/4(금)**	- UI 구성 : Streamlit - 예외 처리 - 종합 시나리오 검증	Streamlit 연동 확인 발표용 시나리오 구성
	7/5(토) ~ 7/6(일)	- 디버깅, 테스트 - 발표 준비	실무 상황 시뮬레이션
발표	**7/7(월)**	발표	실시간 챗봇 시연 역할별 구현 내용 소개

프로젝트 수행 경과

프로젝트
수행 경과

01 **Dataset 구성, 모델 선정, 데이터 전처리 및 모델 학습**
머신러닝을 통한 데이터 기반 학습, XGBoost 모델 정확도 77% 구현

02 **Responses API 기반 통합 구조**
AI 어시스턴트, 블랙리스트, Function_call, Web Search, File Search 등

03 **UI 구성**
Streamlit을 활용하여 사용자 친화적인 UI 구성

04 **디버깅, 리팩토링, 발표 준비**

프로젝트 수행 경과

XGBoost

Extreme Gradient Boosting

피싱 URL 탐지 모델 선정

- **최종 선택 모델**
  - Gradient Boosting 확장 형태의 앙상블 학습 모델
- **주요 특징**
  - 과적합 방지, 정확도, 속도 측면 개선
  - 회귀 및 분류 문제 모두 사용 가능
  - 대규모 Dataset에서도 뛰어난 성능 발휘
- **작동 원리**
  - 잔차 기반 학습 : 이전 모델의 잔차를 학습 → 정확도 높임
  - 경사하강법 : 손실 함수 최소화 방향으로 모델 업데이트
  - 정규화 : 복잡한 모델에 패널티 → 과적합 방지
  - 병렬 처리 : 빠른 학습 속도
- **학습 방식** : 각 트리는 이전까지의 예측이 틀린 부분을 **"집요하게 물고 늘어져서"** 전체 성능을 개선함

프로젝트 수행 경과

피싱 URL 탐지 모델 선정

```python
# 데이터 정규화
def normalize_url(url):
    url = url.strip()
    url = re.sub(r'^https?://', '', url)  # http:// 또는 https:// 제거
    return url

# 데이터 불러오기
df = pd.read_csv('url.csv', encoding='latin1')

# label 1 데이터 추출
label_1 = df[df['label'] == 1]

# label 0 데이터에서 label 1 개수만큼 랜덤 샘플링
label_0 = df[df['label'] == 0].sample(n=len(label_1), random_state=42)

# 합쳐서 섞기
balanced = pd.concat([label_1, label_0]).sample(frac=1, random_state=42).reset_index(drop=True)
print(balanced['label'].value_counts())

# 데이터 저장 및 새로운 데이터 불러오기
balanced.to_csv('data.csv', index=False, encoding='utf-8-sig')
df = pd.read_csv('data.csv')
```

새로 준비한 Dataset 정상 URL과
악성 URL 개수 2배 이상 차이 있음

데이터 전처리를 통해 개수 비율 맞추어
새로운 CSV 파일 생성

프로젝트 수행 경과

피싱 URL 탐지 모델 선정

✅ 결과는 다음과 같습니다:

- **label = 0** : 428,102개
- **label = 1** : 223,086개

데이터 전처리 전 개수 차이

전처리

✅ 결과는 다음과 같습니다:

- **label = 0** : 223,086개
- **label = 1** : 223,086개

전처리 후 데이터 개수 비율 맞춤

프로젝트 수행 경과

피싱 URL 탐지 모델 선정

```python
# Feature 추출
# 참고 자료 : https://archive.ics.uci.edu/dataset/327/phishing+websites
def extract_url_features(url) :
    feature = {}
    url = normalize_url(url)
    try:
        parsed = urlparse(url)
    except ValueError as e:
        print(f"[URL 파싱 오류] {url} → {e}")
        return {}  # 또는 기본값 반환

    # 기본 URL 기반 Feature
    feature['url_length'] = int(len(url) > 75)                          # URL 전체 문자열 길이
    feature['num_dots'] = np.log1p(url.count('.'))                      # URL 내 점(.)의 개수
    feature['has_ip'] = int(bool(re.search(r'\d+\.\d+\.\d+\.\d+', url)))  # IP 주소 포함 여부
    feature['num_special_chars'] = int(len(re.findall(r'[^\w]', url)) > 5)  # 특수문자 수 (@, ?, =, % 등)
    feature['has_at_symbol'] = int('@' in url)                          # @ 기호 포함 여부
    feature['path_length'] = np.log1p(len(parsed.path))                 # URL 경로 길이
    feature['num_digits'] = np.log1p(len(re.findall(r'\d', url)))       # 숫자 개수

    return feature
```

프로젝트 수행 경과

피싱 URL 탐지 모델 선정

```
# 순차적 특징 추출
features = [extract_url_features(str(url)) for url in tqdm(df['url'].fillna(''), desc='Extracting Features')]

# Feature, Target 설정
X = pd.DataFrame(features)
y = df['label'].astype(int)

# 학습 데이터 및 테스트 데이터 분할
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# XGBoost 모델 생성 및 학습
xgb_model = XGBClassifier(
    n_estimators=200,           # 부스팅할 트리 개수 : 많을수록 모델 복잡도와 과적합 가능성 증가
    max_depth=4,                # 각 트리의 최대 깊이 : 깊을수록 더 복잡한 패턴 학습 가능, 그러나 과적합 위험 있음
    learning_rate=0.5,          # 학습률 : 작을수록 학습이 느리지만 안정적임
    random_state = 42,          # 난수 고정
    use_label_encoder=False,    # 라벨 인코더 사용 여부
    eval_metric='logloss',      # 평가 지표 설정 : 로그 손실 함수, 작을수록 예측 확률이 정답에 가까움
    tree_method='hist',
    device='cuda'
    )
xgb_model.fit(X_train, y_train)

# 예측 및 성능 평가
y_pred = xgb_model.predict(X_test)
print("\n정확도:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

# 모델 저장
joblib.dump(xgb_model, 'XGBoost_model.pkl')
```

프로젝트 수행 경과

피싱 URL 탐지 모델 선정

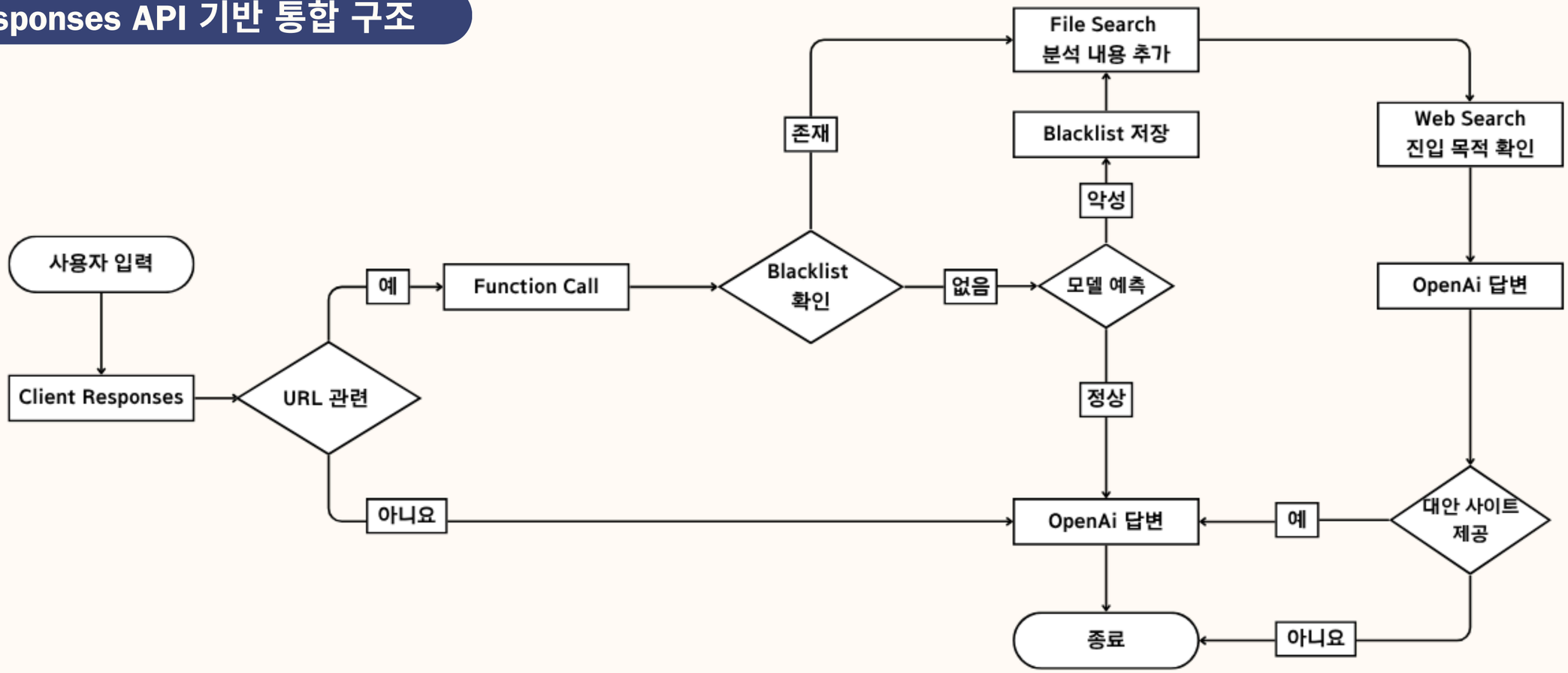
정확도: 0.7731607553090155

	precision	recall	f1-score	support
0	0.76	0.79	0.78	44705
1	0.78	0.75	0.77	44530
accuracy			0.77	89235
macro avg	0.77	0.77	0.77	89235
weighted avg	0.77	0.77	0.77	89235

Classification Report 결과

프로젝트 수행 경과

Responses API 기반 통합 구조



전체 시스템 작동 로직 FlowChart

프로젝트 수행 경과

Responses API 기반 통합 구조

```
def agent_call(url):
    client = openai.OpenAI(api_key=OPEN_API_KEY)
    store_id = vector_store_with_file(
    "./악성url관련자료.pdf",
    "knowledge_base",
    client
    )

    if url:
        result = check_black_list(url)
        client = openai.OpenAI(api_key=OPEN_API_KEY)
        input = [{
            "role" : "system",
            "content" : "당신은 악성 url 결과를 잘 리포팅해서 상위 에이전트에게 전달하는 역할을 합니다.\
                당신은 블랙리스트에 존재한다는 문자나 0 또는 1을 받습니다. 블랙리스트에 존재하면 악성 url입니다.\
                    0이면 정상 url, 1이면 악성 url을 뜻하고 이는 ML모델에 넣은 결과입니다. 정상 url은 그냥 상위에이전트에게 정상이라고 전달하면 되고\
                        악성 url이면 벡터 스토어의 파일에는 악성 URL에 대한 자료가 있는데 이걸로 해당 URL에 대한 정보를 가져오는게 아니라\
                            왜 악성 URL인지에 대한 분석을 하는데 용이한 자료입니다. 해당 자료를 통해 악성 url이면 분석결과까지 리포팅해서 답변을 해주세요."
        },
        {
            "role" : "user",
            "content" : f"사용자 url: {url} 결과 : {str(result)}"
        }]
        response = client.responses.create(
            model="gpt-4o",
            input = input,
            tools=[{"type" : "file_search",
                    "vector_store_ids" : [store_id]}]
        )
        return response.output_text
```

프로젝트 수행 경과

Responses API 기반 통합 구조

```python
class Agent:
    def __init__(self, client):
        self.client = client
        self.tools = [{
            "type": "function",
            "name": "agent_call",
            "description": "사용자로부터 악성 url 관련 판단 질문을 받으면 url 검사와 리포팅을 위해 하위 에이전트를 호출하는 함수입니다.",
            "parameters": {
                "type": "object",
                "properties": {
                    "url": {"type": "string"}
                },
                "required": ["url"],
                "additionalProperties": False
            },
            "strict": True
        },
        {
            "type": "web_search_preview"
        }
        ]
```

프로젝트 수행 경과

블랙리스트 확인

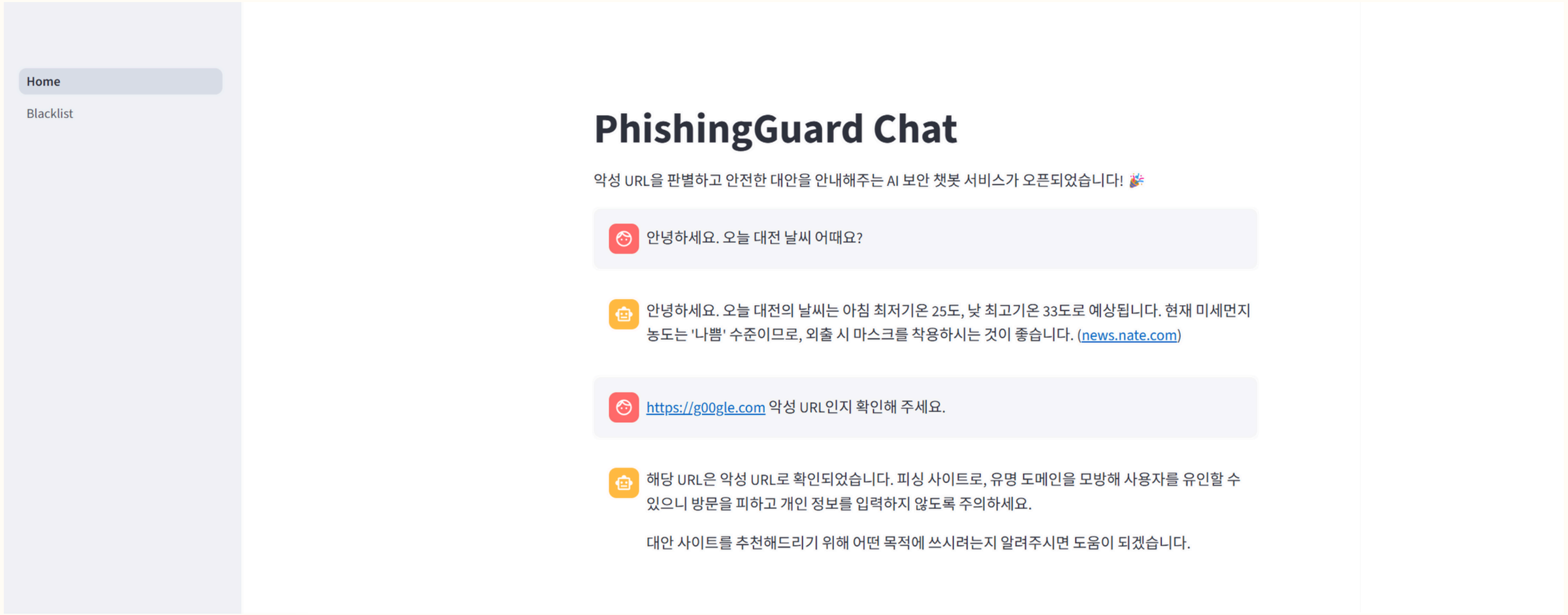
```python
def check_black_list(url):
    '''
    사용자로부터 URL을 받았을 때 블랙리스트를 검사하고
    블랙리스트에 없으면 ML 모델을 통해 검사하고 결과를 주는 함수입니다.
    '''
    black_list_df = load_blacklist() # 블랙리스트 불러오기

    # 블랙리스트가 비어있지 않다면
    if not black_list_df.empty:
        black_list = black_list_df['url'].tolist()
    # 비어있다면
    else:
        black_list = []

    # 블랙리스트에 존재하면 return
    if url in black_list:
        return "해당 URL은 블랙리스트에 존재하는 악성 URL입니다."
    # 블랙리스트에 존재하지 않으면 모델 부르기
    check_t_f = model_call(url)
    # 악성 url이면
    if check_t_f:
        # 블랙리스트에 추가
        black_list.append(url)
        # df로 만들어서
        df = pd.Series(black_list, name="url").to_frame()
        # csv로 저장
        save_csv(df)
        # 결과 반환하기
        return check_t_f
    # 정상이면
    else:
        return check_t_f
```

악성 URL 확인

```
url
http://iop.golzw.loan
g00gle.com
http://kgm.ilogekgvm.com
http://kfdcv.noex.social
skm.ilogenskk.com/
mk.logenkk.com/
https://www.cjlogisticm.com
http://bit.ly/3rcfQ0U
http://han.gl/WBTaOc
https://han.gl/uHvGDT
```

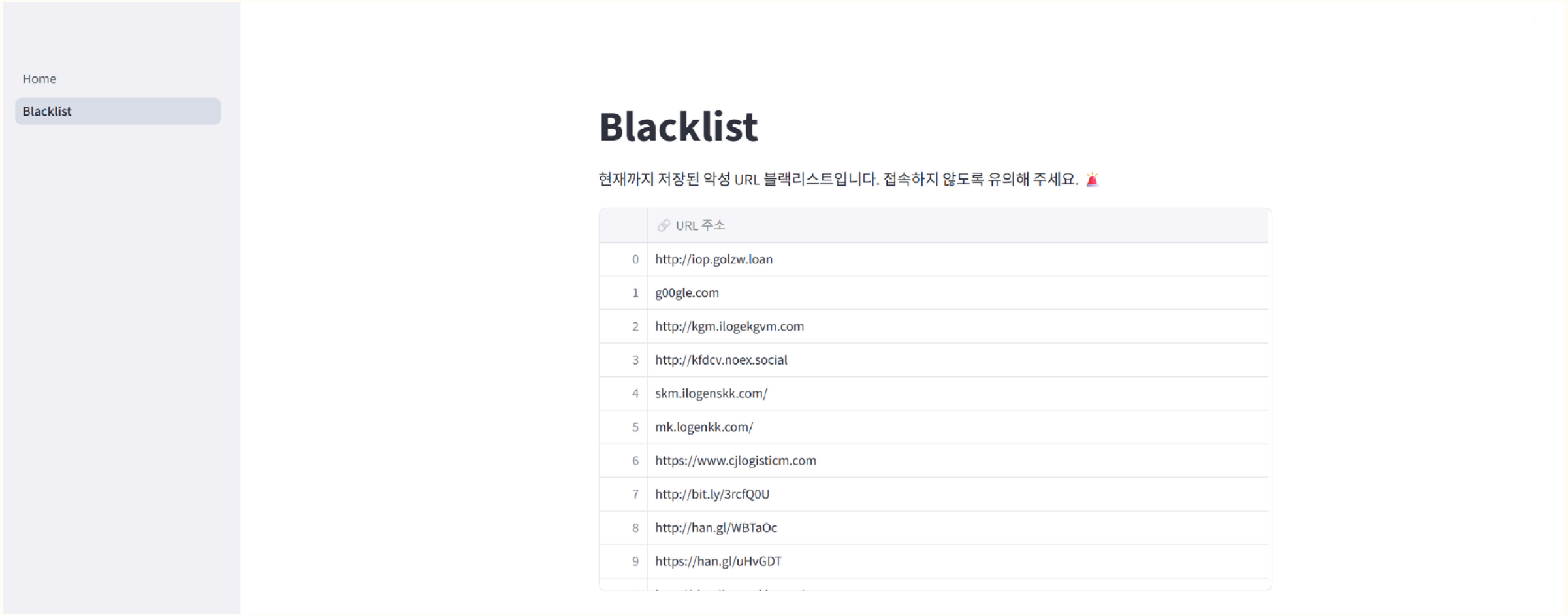
프로젝트 수행 경과

Streamlit 활용 UI



프로젝트 수행 경과

Streamlit 활용 UI



프로젝트 수행 경과

필수 요구사항 충족

OpenAI API 및 에이전트 플랫폼 활용	◦ Responses API 기반으로 URL 관련 응답 생성 ◦ Web Search Preview, File Search 호스팅 툴 연동
커스텀 함수(툴) 구현	◦ XGBoost 모델 학습 → .pkl 모델 파일 저장 ◦ Agent 클래스 및 커스텀 함수로 에이전트 연결 → 사용자 질문에 따라 추론 ◦ 구조 명확화 → 악성 URL 판단 질문 후 사용자 요청에 따라 대안 사이트 추천
멀티에이전트 활용	◦ 하위 에이전트 악성코드 탐지 → File Search 활용 분석 ◦ 상위 에이전트 전달 받음 → Web Search 결과 반환
통합 대화 UX	◦ 사용자 입장에서 자연어로 자유롭게 질문 또는 지시 가능

프로젝트 수행 경과

심화 요구사항 충족

01 **파일 분석** 반영 결과 자동 생성

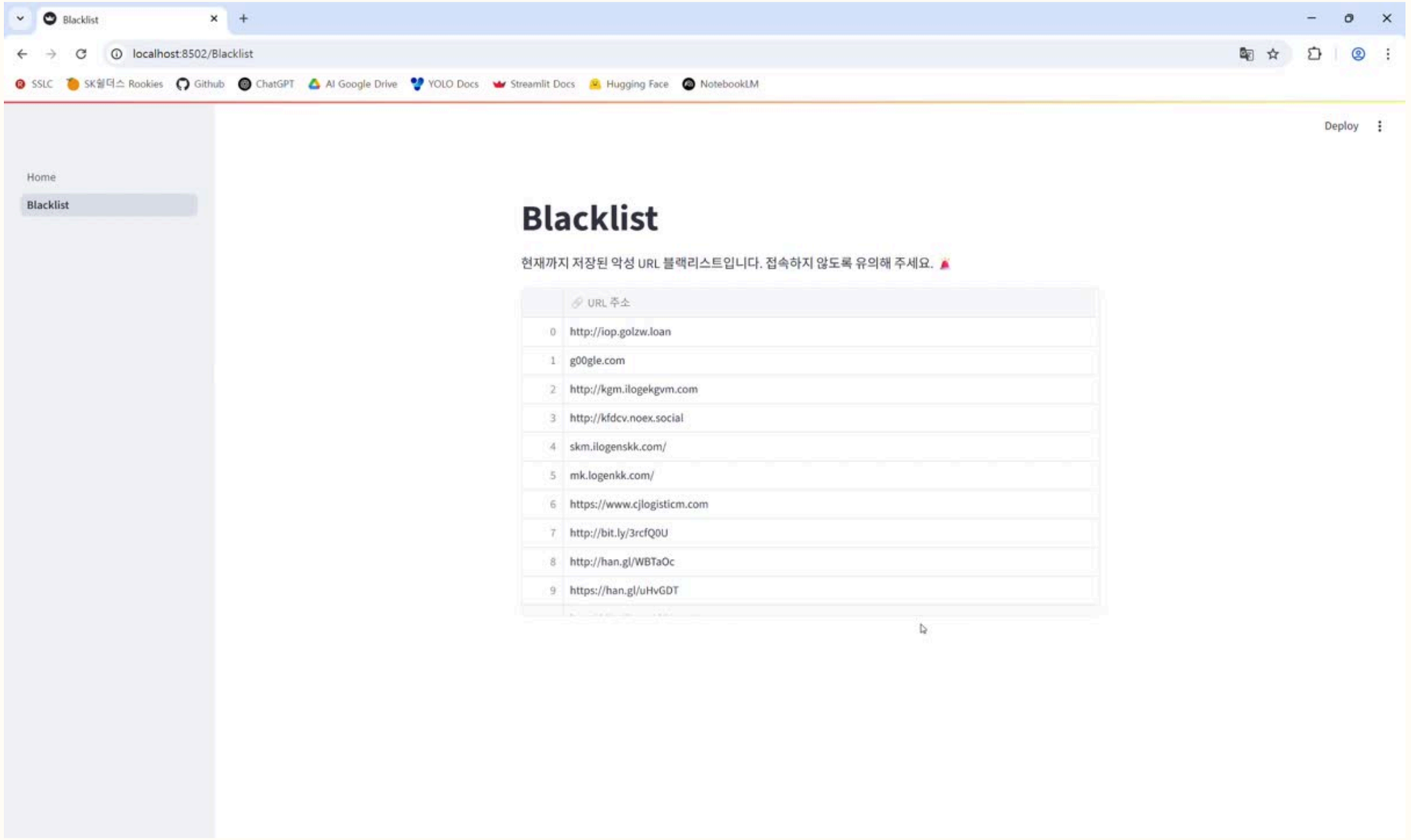
02 이전 대화를 저장하여 사용자별 **맞춤형 에이전트**

03 **블랙리스트** 실시간 로깅 대응

04 **다양한 실서비스 사례**에 맞는 시나리오 추가

프로젝트 수행 경과

시연 영상



자체 평가 의견

사전 기획의 관점

프로젝트 결과물 완성도 평가

★★★★★

계획한 기능 모두 구현

우리 팀의

칭찬할 점
아쉬운 점

★★★★★

팀워크

★★★☆☆

초기 낯가림

우리 팀의

프로젝트 개선 방안

★★★★☆

- **Feature 선정 제약**
- **모델 정확도**
- **AI 어시스턴트 응답 시간**

프로젝트 수행 중

느낀 점
경력 계획 관련 성과

★★★☆☆

- **Dataset 확보 어려움**
- **머신러닝 활용 난이도 실감**

★★★★☆

- **악성 URL에 대한 이해**
- **피싱 사고 위험 실감**

참고 자료

Dataset 출처 :

Malicious URLs dataset - Kaggle
https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset

모델 선정 근거 :

URL 주요특징을 고려한 악성URL 머신러닝 탐지모델 개발 - 한국정보통신학회논문지 - Vol. 26, No. 12: 1786~1793, Dec. 2022
https://koreascience.kr/article/JAKO202203255703372.pdf

Feature 선정 근거 :

UC Irvine Machine Learning Repository - Phishing Websites Donated on 3/25/2015
(University of California, Irvine)
https://archive.ics.uci.edu/dataset/327/phishing+websites

Vector Store 저장 파일:

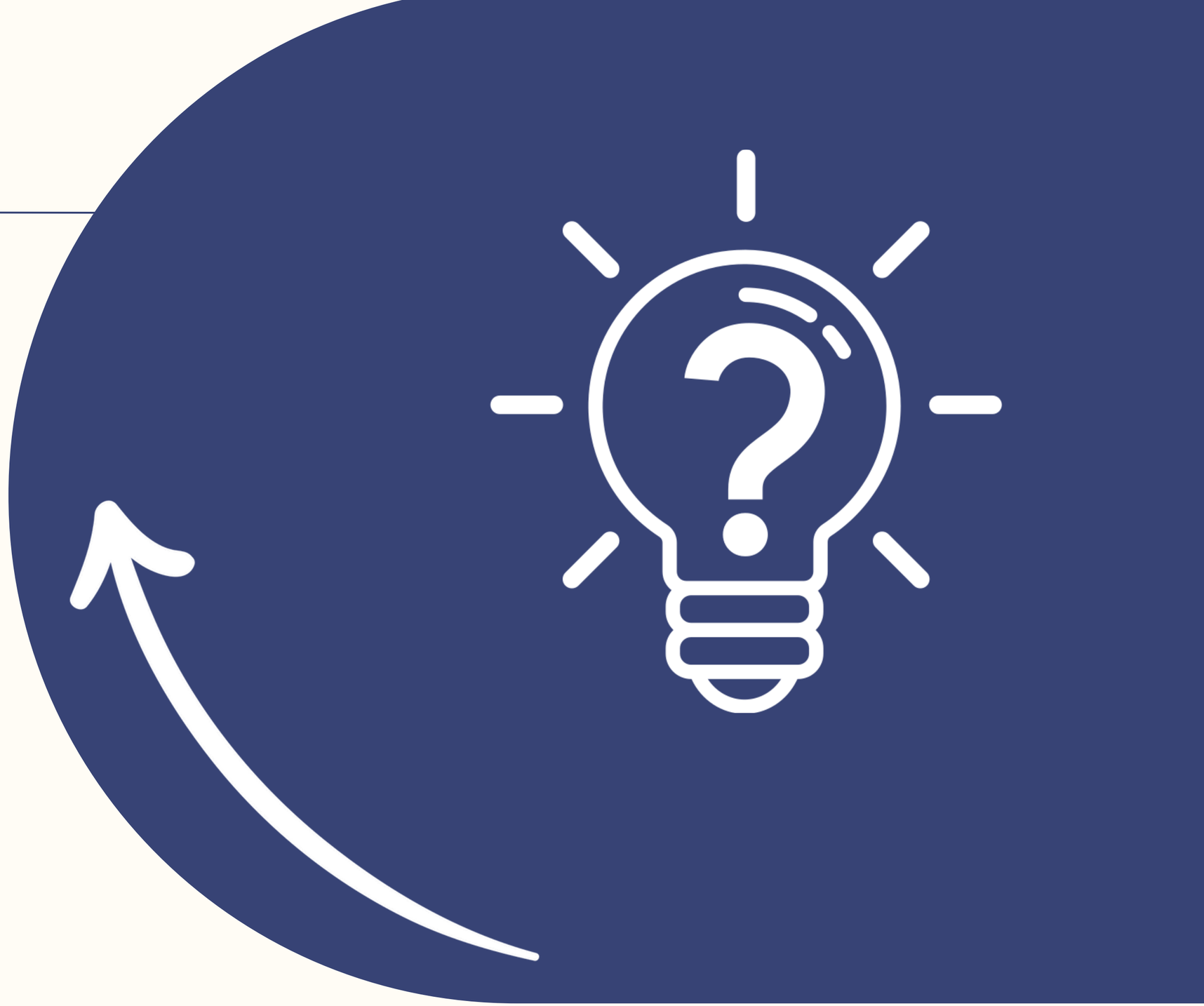
Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2011). *Learning to detect malicious URLs*. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3), Article 30. https://doi.org/10.1145/1961189.1961202
https://doi.org/10.1145/1961189.1961202

구현 작업물

Github 주소 :
https://github.com/minhyung-aram/black_list_team_project.git

Canva(PPT 제작):
https://www.canva.com/design/DAGsKPiO94U/eePHiwA0Wiy-Ad8jpjqV9g/edit?utm_content=DAGsKPiO94U&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Q&A



Thank you!

감사합니다!