2023 전기 졸업과제 중간보고서

멀티모달 기반 스팸 필터링 플랫폼

지도교수 : 최 윤 호
분 과 명 : 지능형융합보안
팀 명 : 스 팸 도 시 락
202055550 박 혜 경
202055578 이 승 현
202055608 천 영 채

목차

1. 요구조건 및 제약 사항 분석에 대한 수정사항

1.1 요구조건

- 데이터 수집
  - 텍스트 데이터
    - 온라인 상에 존재하는 스팸 데이터(Enron dataset, SMS spamdataset)를 수집한다.
  - 이미지 데이터
    - 온라인 상에 존재하는 Image Spam Dataset 을 수집한다.
    - 피싱 사이트에 존재하는 이미지를 수집한다.
  - 음성 데이터
    - 피싱 전화 사기 수법 패턴에 대해 수집한다.
- 수집한 데이터의 스팸 여부 분석
  - 텍스트 데이터
    - Python 을 이용하여 수집한 데이터를 전처리, 분석하여 모델에 넣을 수 있게 가공한다.
    - 가공한 데이터를 모델에 넣어 스팸 여부를 계산한다.
  - 이미지 데이터
    - Python 을 이용해 이미지의 텍스트를 추출하여 텍스트 토큰으로 만든다.
    - 만들어진 토큰을 연결하여 스팸 필터링 모델에 넣을 수 있는 데이터로 가공하여 모델에 넣어 스팸 여부를 계산한다.
  - 음성 데이터
    - Python 과 음성인식 API 를 활용하여 화자를 분리하여 대화 스크립트 형식으로 추출한다.
    - 추출된 문장을 모델에 넣을 수 있는 데이터로 가공하여 모델에 넣고 스팸 여부를 계산한다.
- 서비스 제공 형식
  - 웹 형식으로 제공하여 메일이나 이미지를 받았을 경우 데이터를 넣어 확인할 수 있게 한다.
  - 음성 서비스의 경우 파일을 넣는 것과 실시간 탐지 기능을 구분하여 제공하며, 모바일 웹을 통해 통화중에도 사용할 수 있게 하여 실시간 통화의 스팸 여부도 계산할 수 있도록 한다.

1.2 기존 제약 사항 및 추가 제약 사항

기존 제약 사항	대책
기존 필터링 시스템을 우회하기 위한 지능적인 스팸 메시지가 증가했다.	딥러닝을 기반으로 한 반복된 학습과 빅데이터 분석기술을 통해 교묘하게 단어를 바꾸어 보내는 메시지도 탐지할 수 있게 한다.
음성 스팸 메시지의 경우 기존 자료가 많지 않다.	직접 음성 스팸 데이터를 모은다. 기존 데이터와 직접 모은 데이터를 같이 활용할 예정이다
음성 스팸 메시지의 경우 스팸 음성 감지와 동시에 전화를 차단하기에 한계가 있다.	실시간으로 음성을 들려주면, 그 파일을 저장해 바로 스팸 필터링을 해줌으로써 사용자는 실시간으로 느낄 수 있게 할 예정이다.
추가 제약 사항	**대책**
모델 1 개를 사용할 경우 신뢰도가 떨어질 수 있다.	2 개의 모델을 쓰고 각각의 정확도를 보여줌으로써 다양성과 정확도를 높일 예정이다.
스팸 필터를 돌릴 때마다 결과는 동일하지만, 수치는 다르게 나온다.	데이터를 효율적으로 처리할 구조를 설계한다

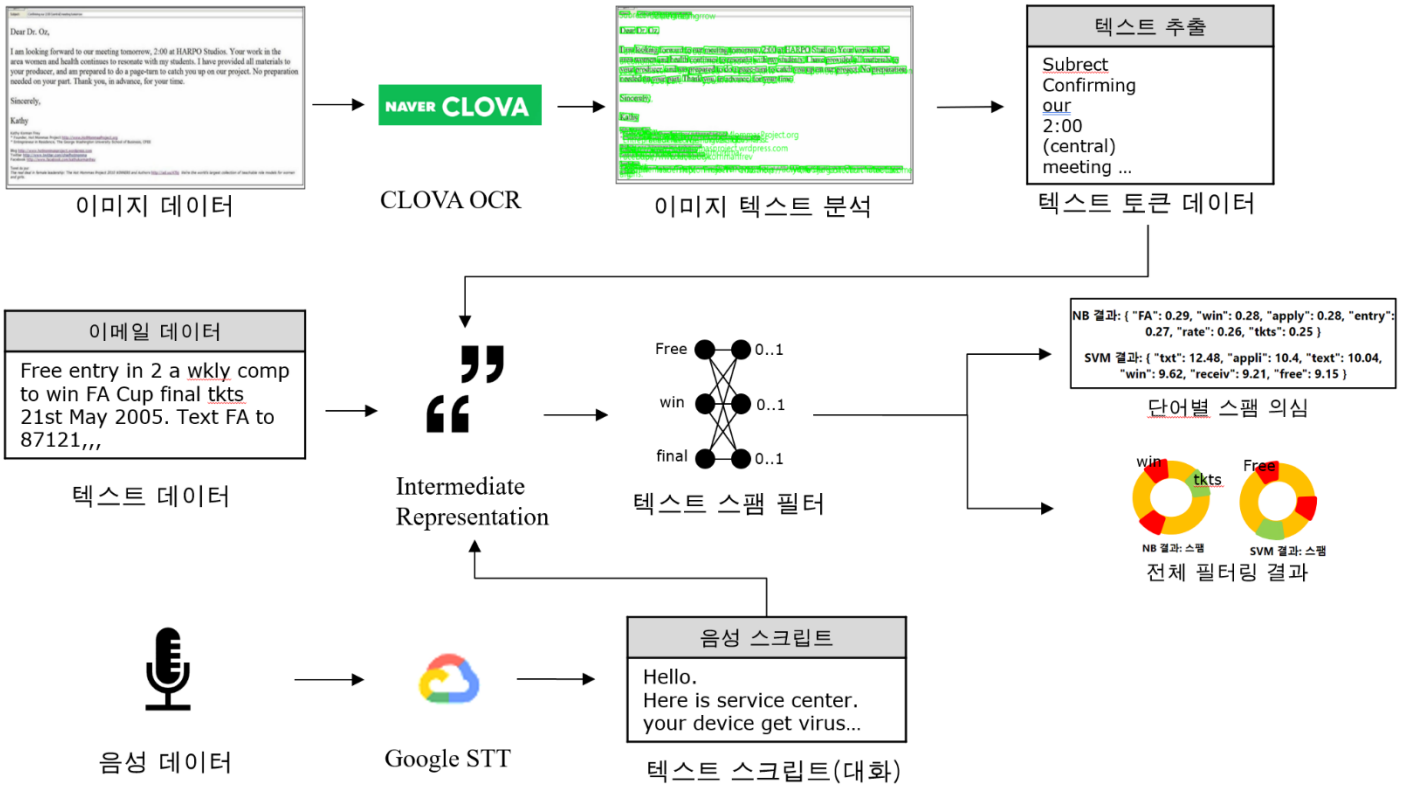
2. 설계 상세화 및 변경 내역

2.1 모델 설계 및 구조

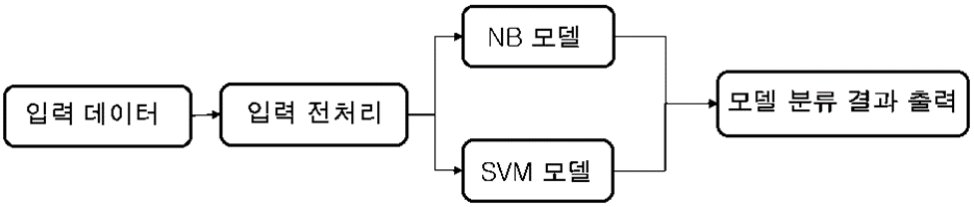
1. 학습 모델 구상 : 이미지, 텍스트, 음성에서의 텍스트 형식이 다르다고 판단하여 각각 다르게 데이터를 추출하고 이를 스팸 필터에 넣을 수 있는 IR(Intermediate Representation)으로 가공한다.

2. IR : content based 스팸 분류이므로, 각각의 input text 를 모델에 넣기 적합한 형태로 가공한다. 의미 없는 단어 및 문장 기호 삭제, 문법적으로 이어지도록 문장을 합치는 등 적절하게 문장을 가공하고, 벡터화 한다.

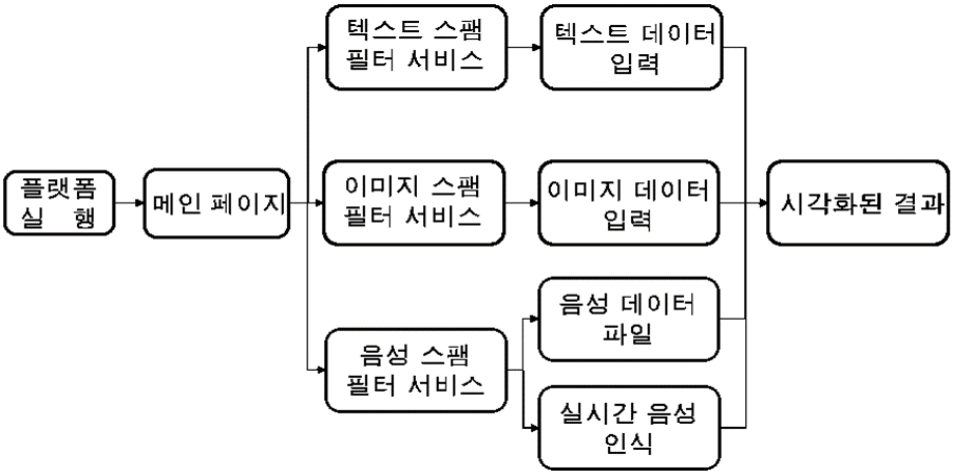
이미지는 네이버 클로바 OCR API 를, 음성은 구글 STT API 를 사용하여 변환한다.



1. Server



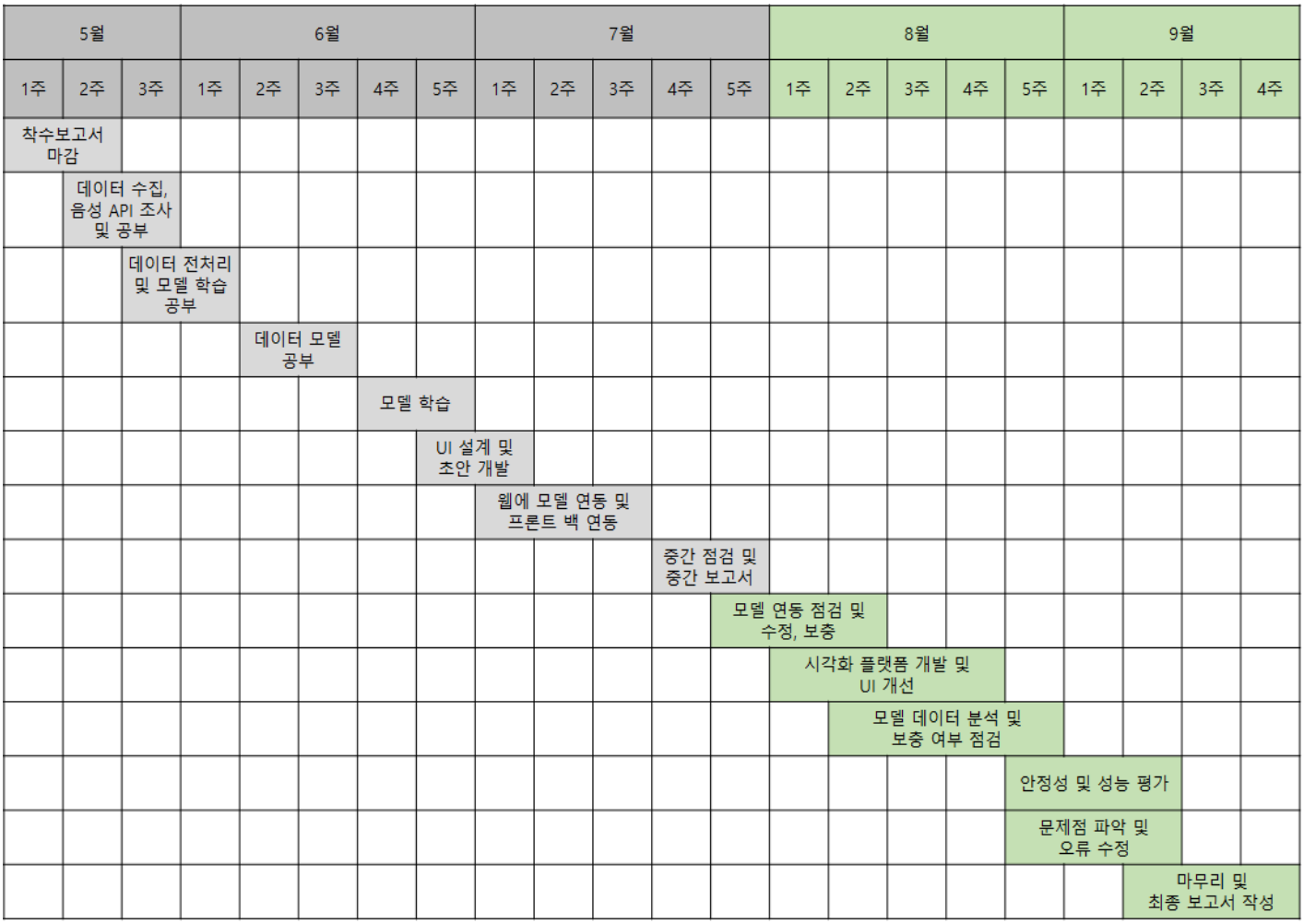
2. Client



3. 갱신된 과제 추진 계획

스팸 이미지를 판별하기 위해 이미지에서 텍스트를 추출하는 과정이 필요한데 이는 CLOVA OCR API 를 사용하여 구현할 예정이며, 스팸 음성을 분류하기 위해 음성을 텍스트로 변환하는 과정에서는 Google STT 를 사용하여 구현한다.

진행된 것 : 회색 / 진행 예정 : 녹색



4. 구성원별 진척도

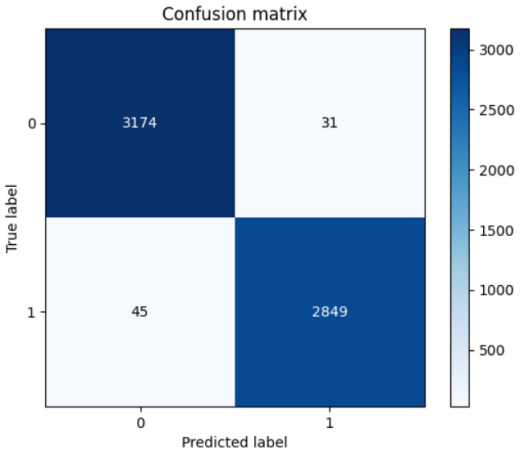
이름	역할 분담
박혜경	- SVM 스팸 필터링 모델 학습 - UI 설계 및 개발 - 학습 모델 추출 및 웹에 연동
이승현	- Naive Bayes 스팸 필터링 모델 학습 - UI 설계 및 개발 - Google STT 이용해서 텍스트 변환 및 스팸 분류
천영채	- KNN 스팸 필터링 모델 학습 - CLOVA OCR API 이용하여 텍스트 추출 및 스팸 분류 - 스팸 분류 웹 백엔드 개발

5. 보고 시점까지의 과제 수행 내용 및 중간 결과

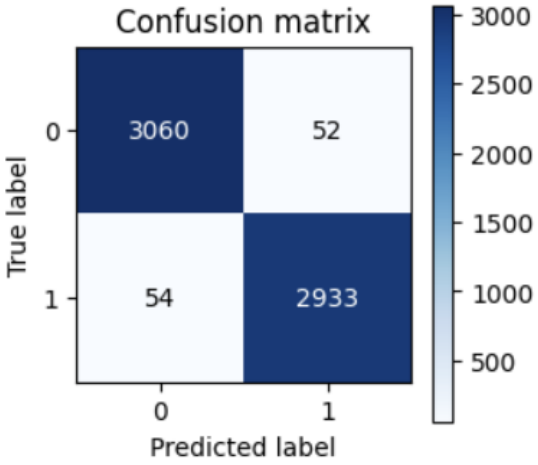
5.1 데이터 모델 학습

스팸 필터링으로 사용할 모델을 결정하기 위해 여러가지 머신러닝 기법을 학습시키고 가장 높은 정확도를 보이는 SVM 과 Naivce Bayes 기법을 사용하기로 정하였다.

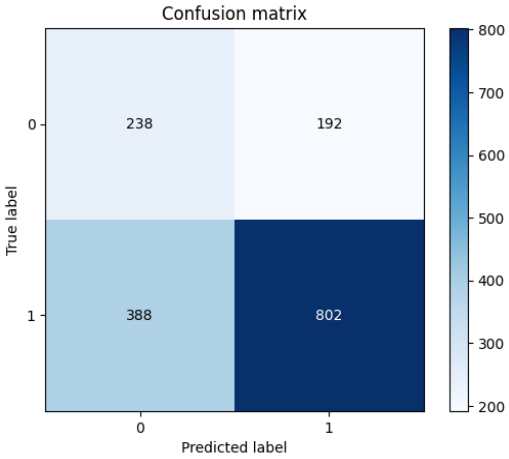
- Naive Bayes Confusion Matrix



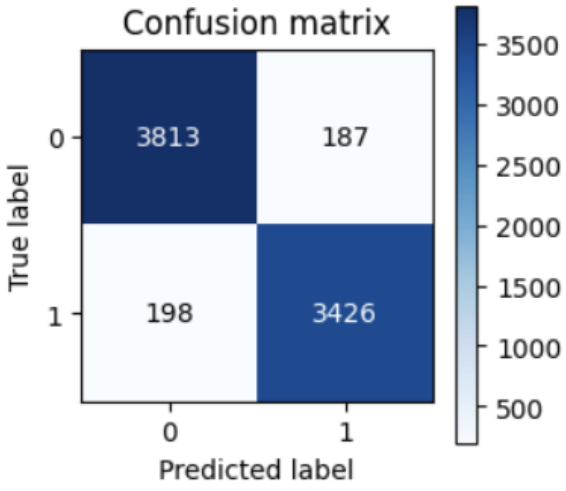
- SVM Confusion Matrix



- KNN Confusion Matrix



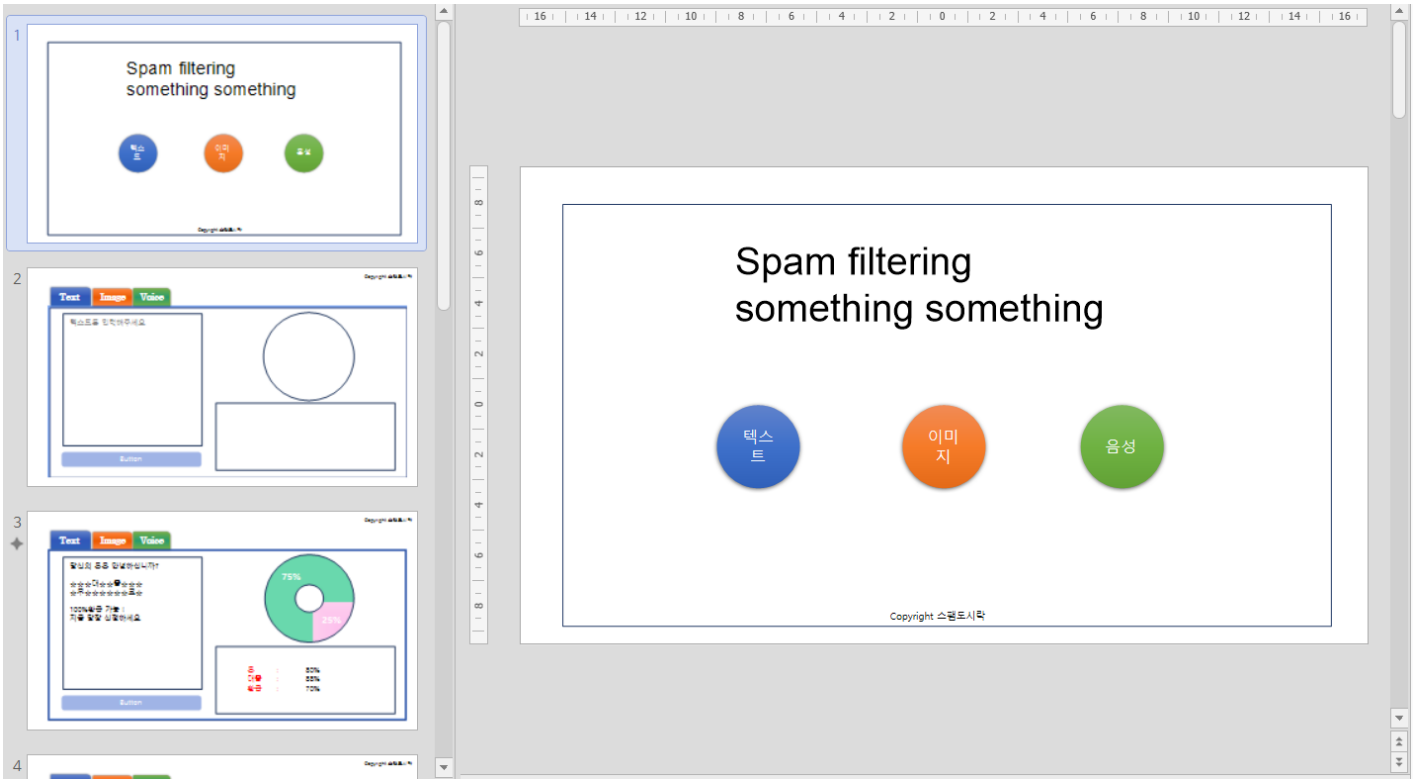
- Decision Tree Confusion Matrix



-

5.2 웹 UI 설계

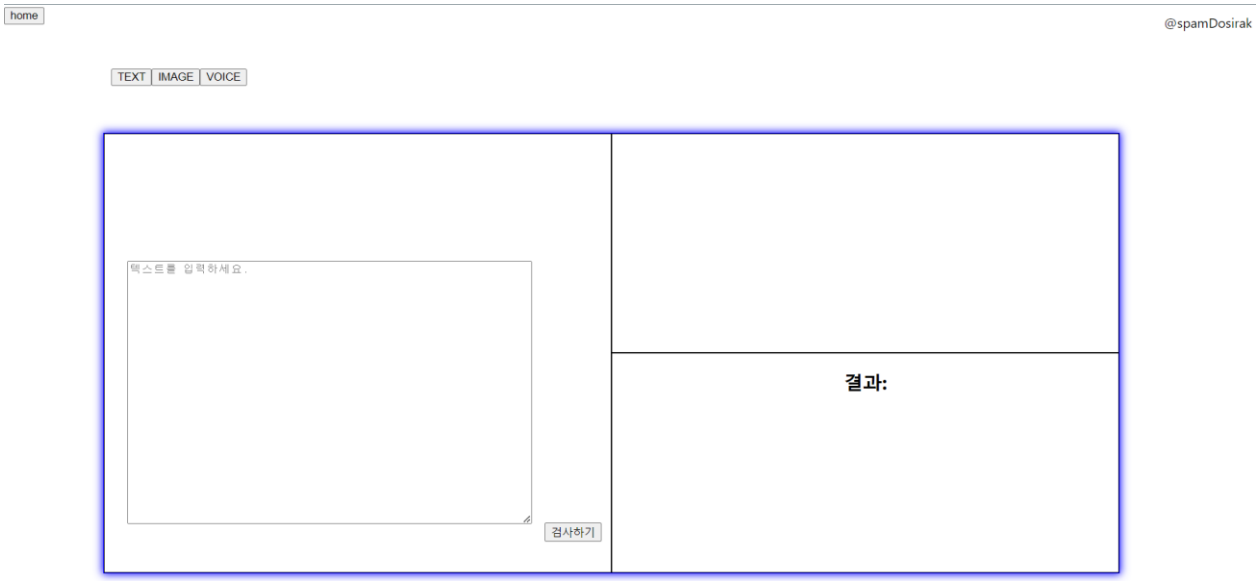
- 코드 작성 전 기본적인 틀과 UI 설계 과정



- React 를 이용한 웹페이지 설계 (미완성)
  - 메인 페이지

@spamDosirak

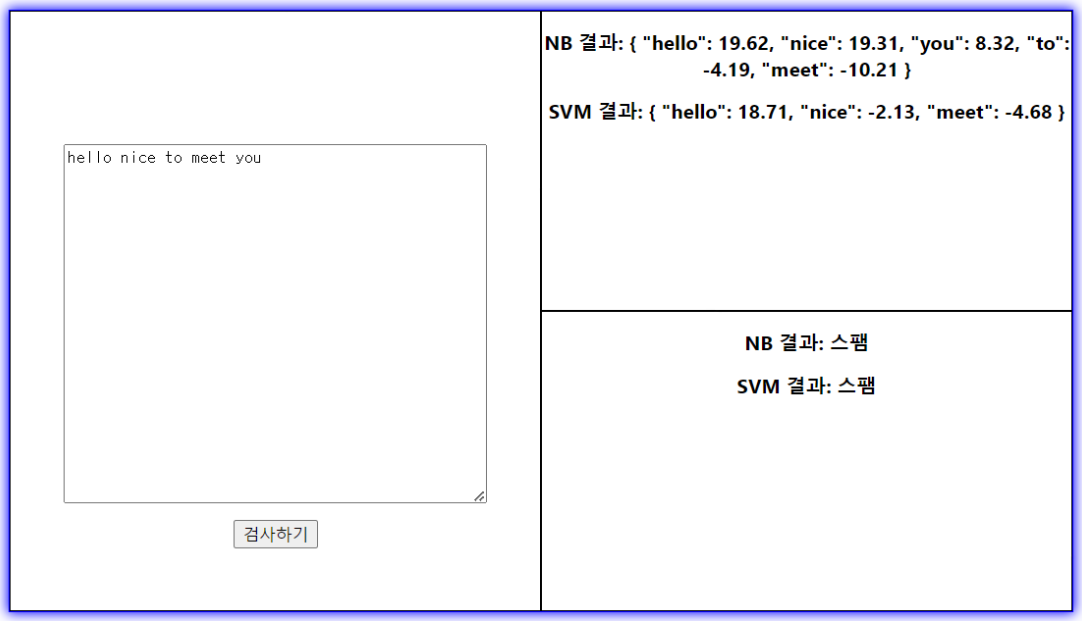
- 텍스트 페이지



5.3 백엔드/프론트엔드 연결

5.3.1 텍스트 스팸필터링 후 결과 확인

```python
#Text Spam 검사
@app.route('/predict',methods=['POST'])
def predict():


    if request.method == 'POST':
        data = request.get_json()
        section = data['section']
        value = data['value']
        #value = data['email']
        vect = cv.transform([value]).toarray()
        from sklearn.metrics import accuracy_score
        accuracy = accuracy_score(y_test,y_pred)

        #단어들
        exp = explainer.explain_instance(value, pipeline.predict_proba, num_features=6)
        print(exp.as_list())

        prediction = pipeline.predict_proba([value])[0,1]
        my_prediction = clf.predict(vect)
        print((exp).as_list())

        import json
        s = list(exp.as_list())
        l = {}
        for i in s :
            if i[0] not in l :
                l[i[0]] = round(i[1]*100,2)

    return jsonify({'result': '스팸' if my_prediction == 1 else '햄', 'vocabs' : json.dumps(l,indent=10)})
```



5.3.2 이미지 스팸필터링 후 결과 확인

```
payload = {'message': json.dumps(request_json).encode('UTF-8')}
files = [
('file', open(image_file,'rb'))
]
headers = {
'X-OCR-SECRET': secret_key
}
response = requests.request("POST", api_url, headers=headers, data = payload, files = files)

#print(response.text.encode('utf8'))

result = response.json()
with open('result.json','w',encoding='utf-8') as make_file:
    json.dump(result, make_file, indent='\t')

text = ""
for field in result['images'][0]['fields']:
    text += field['inferText'] + ' '

vect = cv.transform([text]).toarray()
my_prediction = clf.predict(vect)


# 처리 결과를 'result' 필드에 담아서 JSON 응답으로 반환
return jsonify({'text': text, 'result': '스팸' if my_prediction == 1 else '햄'})
```

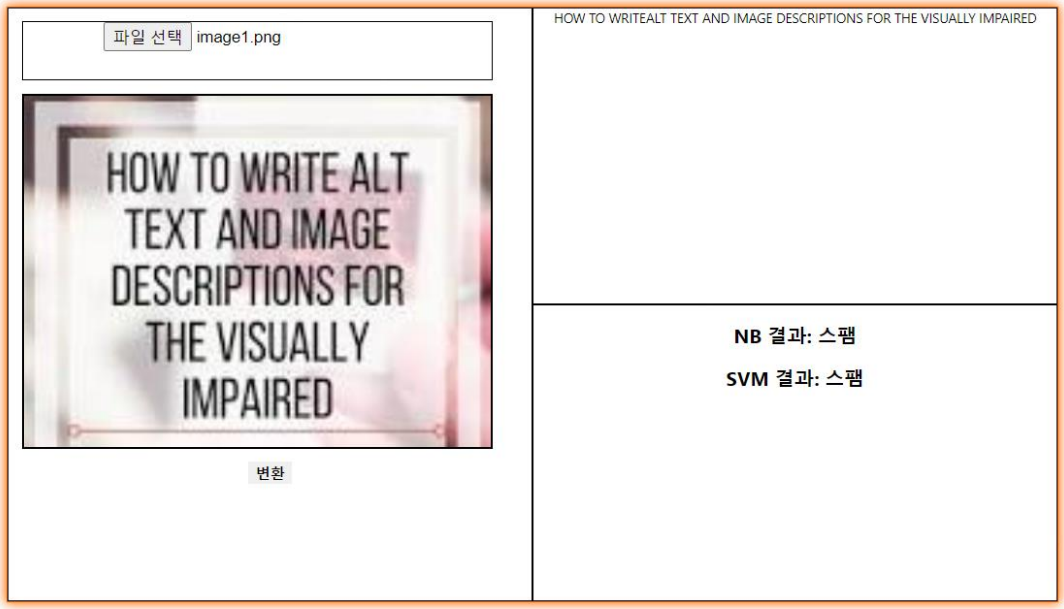
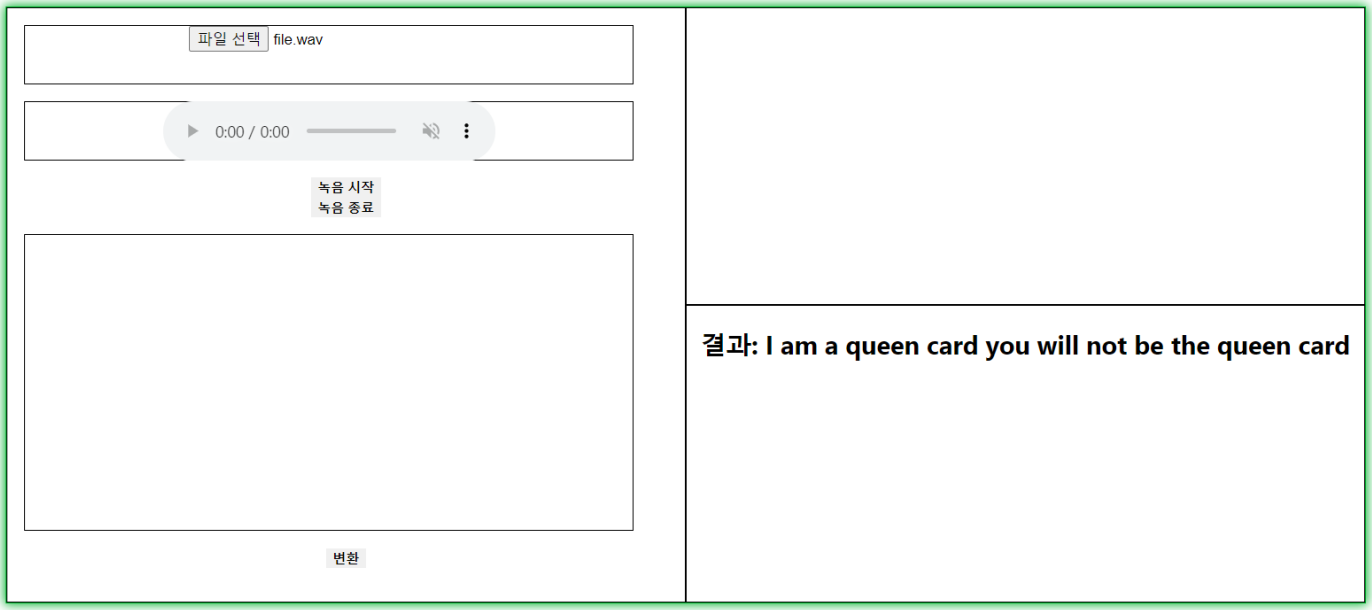


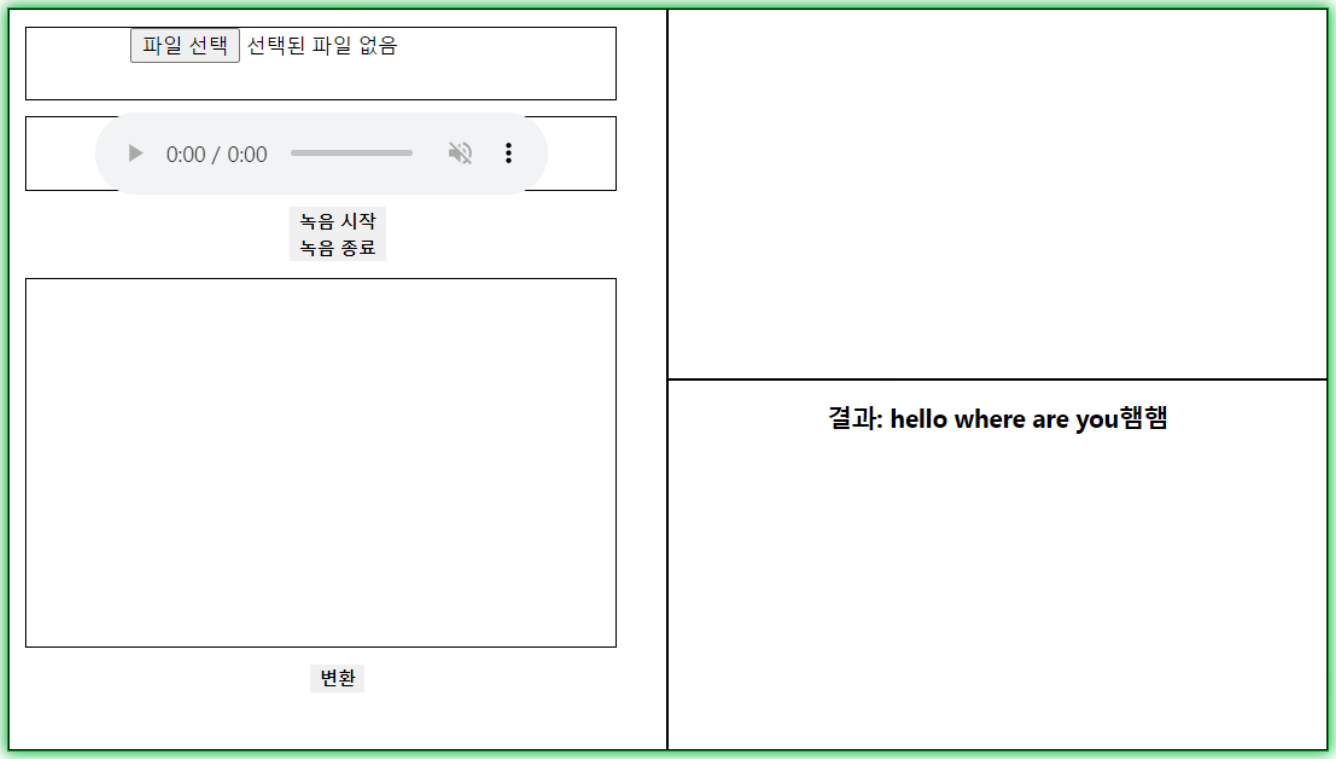
image to text 변환 및 스팸 분류 결과 웹에 올린 모습 (UI 미완성)

5.3.3 음성파일 스팸 필터링 후 결과 확인

```python
@app.route('/convert/audio', methods=['POST'])
def convertAudio():
    audio = request.files['audio']
    audio_file = os.path.join(app.config['UPLOAD_FOLDER'], 'audio.wav')
    print(audio_file)
    audio.save(audio_file)

    from google.cloud import speech
    os.environ["GOOGLE_APPLICATION_CREDENTIALS"]="C:\stt-test-key.json"
    client = speech.SpeechClient()

    #file_name = os.path.join(os.path.dirname(__file__), ".", "file.wav")

    with io.open(audio_file, "rb") as audio_file:
        content = audio_file.read()
        audio = speech.RecognitionAudio(content=content)

    config = speech.RecognitionConfig(
        encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
        #sample_rate_hertz=16000,
        sample_rate_hertz=44100,
        language_code="en-US",
        audio_channel_count = 2,
    )

    # Detects speech in the audio file
    response = client.recognize(config=config, audio=audio)
    result = ""
    for res in response.results:
        result += "{}".format(res.alternatives[0].transcript)

    return jsonify({'result' : result})
```



업로드 한 음성 파일을 텍스트로 변환하여 웹에 띄우기 (UI 미완성)



실시간 녹음한 음성을 텍스트로 변환 + 스팸 분류하여 웹에 올리기 (UI 미완성)