[캡스톤디자인]

스미싱 방지 어플리케이션 결과보고서

대표 학생 :
지도 교수 :

2022. 12.

1. 서론

1. 제안 배경

스미싱(smishing)은 문자메시지(SMS)와 피싱(Phising)의 합성어로 악성 앱 주소가 포함된 휴대폰 문자(SMS)를 대량으로 전송 후 이용자가 악성 앱을 설치하도록 유도하여 금융정보 등을 탈취하는 신종 사기수법이다.

구분	발생 건수	피해 (억원)	피해자 현황					
			20대 이하	30대	40대	50대	60대 이상	계
2016년	17,040	1,468	3,209	3,735	4,542	3,834	1,720	17,040
2017년	24,259	2,470	5,273	4,887	6,473	5,412	2,214	24,259
2018년	34,132	4,040	4,480	6,483	9,842	9,313	4,014	34,132
2019년	37,667	6,398	3,855	6,041	10,264	11,825	5,682	37,667
2020년 (2016년 대비 증감률)	31,681 (▲85.9%)	7,000 (▲376.8%)	5,323 (▲65.9%)	4,406 (▲18%)	7,704 (▲69.6%)	9,217 (▲140.4%)	5,031 (▲192.5%)	31,681 (▲86%)
계	144,779	21,376	22,140 (15.3%)	25,552 (17.6%)	38,825 (26.8%)	39,601 (27.4%)	18,661 (12.9%)	144,779 (100%)

그림 1 최근 5년간 보이스피싱 피해 현황

최근, 코로나로 인한 방역지침이나 확진자추이 등을 미끼로 이러한 신종 사기 수법이 증가하고 있다. 그림1에 따르면 2016년 1468억 원이었던 보이스피싱 피해액은 2017년 2470억 원, 2018년 4040억 원, 2019년 6398억 원, 2020년 7000억 원으로 점진적으로 증가하였고, 50,60대에서 140%이상 피해가 증가하는등 정보의 취약계층인 중장년층의 피해가 크다.

정보취약계층의 보호와 범죄피해를 방지하기 위해 AI를 이용하여 스미싱 방지 기능을 탑재한 어플리케이션을 구현하고자 한다.

2. 프로젝트의 목표

작품의 목표는 문자필터링 시스템을 구현하는 것이다.
SMS를 수신시에 SMS의 내용의 스미싱 여부를 판단하고 첨부된 URL을 검사하여 악성 사이트인지 판단/정보제공/차단을 하는 어플리케이션의 구현이다. 이 어플리케이션을 통해 스미싱 피해를 줄이는 것이 목표이다.

url검사의 경우 바이러스 토탈에서 url유효성 검사, 네트워크 검사, 바이러스 검사 기능등 여러 백신 회사의 엔진을 통하여 처리결과를 반환하므로 정확도가 높게 구현되어 있다. 따라서 바이

러스토탈 API를 이용하여 처리한다.

스미싱 문자의 판독은 감성분석을 이용하며 이미 구현되어 있는 sklearn라이브러리를 이용한다.
한글 데이터의 백터화를 위하여 okt_tokenizer을 이용하여 tfidfvectorizer를 진행할 것이다.
감성분석의 정확도를 높이기 위해 KISA에서 제공하는 스미싱 및 악성 SMS데이터셋을 이용할 것이며, 정상적인 메시지는 공공데이터포털의 API를 통하여 데이터를 수집한다. 또한 사용자 sms를 통해 지속적으로 데이터셋을 업데이트하여 새로이 학습시켜 필터링이 가능하도록 한다.
데이터 부족으로 sms분석과 url검사에서도 정상적인 메시지로 판단된다 하더라도 사용자의 누적된 신고를 통해 임시적으로 차단할 수 있는 기능을 제공하여 실시간 대응할 수 있도록 한다.

1) tfidfvectorizer구현

tfidfVec모델을 이용하여 sms의 메시지가 정상적인 메시지인지 아닌지 분별 할 수 있도록 한다. 또한 지속적으로 학습하여 정확도를 높힌다.

2) url검사

바이러스 토탈의 url검사 api를 이용하여 sms에 포함되어 있는 url의 정상여부를 판단한다.

3) 감성 분석

sms의 메시지가 유사한지를 판단하여 해당 메시지가 스미싱 문자인지 아닌지를 판단하는데 사용한다

2. 관련자료

1. 문자 필터링 시스템(Message Filtering System)

피싱문자/스미싱문자에 주로 사용되는 문자 키워드 DB만을 모아서 피싱/스미싱 문자가 도착함과 동시에 이를 사용자에게 알려주는 기능, 하지만 단어 하나로 단정할 수 없는 포괄적인 기능이라 기능이 매우 불안전하다. (ex,카드, 은행, 대출, 보안, 결제 등)

이러한 문제점을 해결하기 위해 AI를 이용한 시스템을 도입하기로 결정하였으며 TF-IDF를 이용하여 군집분석을 통해 유사도로 문자의 스팸가능성을 판단한다.

2. TF-IDF : (Term Frequency - Inverse Document Frequency)는 정보 검색과 텍스트 마이닝에서 이용하는 가중치로, 여러 문서로 이루어진 문서군이 있을 때 어떤 단어가 특정 문서 내에서 얼마나 중요한 것인지를 나타내는 통계적 수치이다. 문서의 핵심어를 추출하거나, 검색 엔진에서 검색 결과의 순위를 결정하거나, 문서들 사이의 비슷한 정도를 구하는 등의 용도로 사용할 수 있다.TF(단어 빈도, term frequency)는 특정한 단어가 문서 내에 얼마나 자주 등장하는지를 나타내는 값으로, 이 값이 높을수록 문서에서 중요하다고 생각할 수 있다. 하지만 단어 자체가 문서군 내에서 자주 사용되는 경우, 이것은 그 단어가 흔하게 등장한다는 것을 의미한다. 이것을 DF(문서 빈도, document frequency)라고 하며, 이 값의 역수를 IDF(역문서 빈도, inverse document frequency)라고 한다. TF-IDF는 TF와 IDF를 곱한 값이다.IDF 값은 문서군의 성격에 따라 결정된다.

예를 들어 '원자'라는 낱말은 일반적인 문서들 사이에서는 잘 나오지 않기 때문에 IDF 값이 높아지고 문서의 핵심어가 될 수 있지만, 원자에 대한 문서를 모아놓은 문서군의 경우 이 낱말은 상투어가 되어 각 문서들을 세분화하여 구분할 수 있는 다른 낱말들이 높은 가중치를 얻게 된다.

3. NPL

자연어 처리(NLP)는 컴퓨터가 인간의 언어를 이해, 생성, 조작할 수 있도록 해주는 인공 지능(AI)의 한 분야입니다. 자연어 처리는 자연어 텍스트 또는 음성으로 데이터를 상호 연결하는 것으로 '언어 입력(language in)'이라고도 한다. 대부분의 소비자는 자신도 모르는 사이에 NLP와 상호 작용을 하고 있다. 예를 들어, NLP는 Oracle Digital Assistant(ODA)나 Siri, Cortana, Alexa와 같은 가상 도우미의 핵심 기술이다. 이러한 가상 도우미에게 질문을 하면 NLP를 통해 사용자의 요청을 이해할 수 있을 뿐만 아니라 자연어로 응답할 수 있다. NLP는 서면 텍스트와 음성 모두에 적용되며 모든 인간 언어에 적용될 수 있다.

4. 클라우드 활용

사용자에게 전송해줄 DB도, 제공자가 일방적으로 수집하는 것 뿐만 아니라, 클라우드 개념을 활용하여 실제 사용자로부터 스미싱 정보를 쌍방향으로 모집하고, 모집된 DB를 타 사용자에게 전달한다.

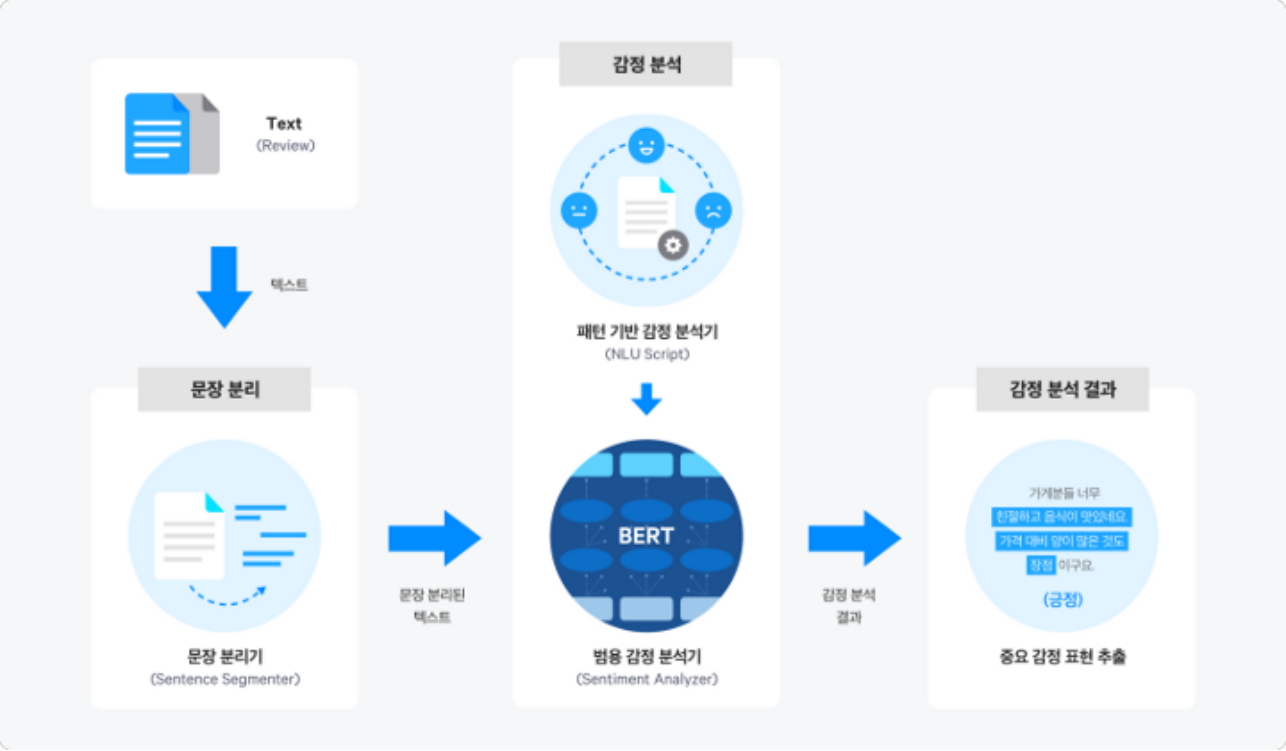
5. 악성 이메일 링크(URL) 검사

스미싱 공격 기법 중 하나로, 공격자가 만든 악성 링크를 사용자에게 전송하여 이를 실행하게 끔 유도한다. 이를 위험하게 연결하여 사이트에 접속하지 않고, VirusTotal과 같이 해당 URL을 입력하면 해당 도메인에 대해 국내외 보안 업체에서 탐지한 내역을 확인할 수 있고, 도메인에서 다운로드 된 파일 중 다수가 악성 파일이라는 정보도 확인할 수 있으며, 실제 허가된 서비스에서 제공하는 URL 주소가 맞는지를 알 수 있다. 이렇게 사용자에게 해당URL에 위험성을 경고함으로써 악성URL로 인한 피해가 방지되도록 한다.

URL검사를 위한 엔진목록은 다음과 같다

- CloudStat
- ADMINUSLabs
- AegisLab WebGuard
- AlienVault
- Antiy-AVL
- malwares.com URL Checker
- Etc.
- MAX

6. 감성 분석 : 감정 분석은 자연어 처리, 텍스트 분석, 컴퓨터 언어학 및 생체 인식을 사용하여 정서적 상태와 주관적 정보를 체계적으로 식별, 추출, 수량화 및 연구하는 것을 의미한다.



3. 프로젝트

1. 프로젝트 소개

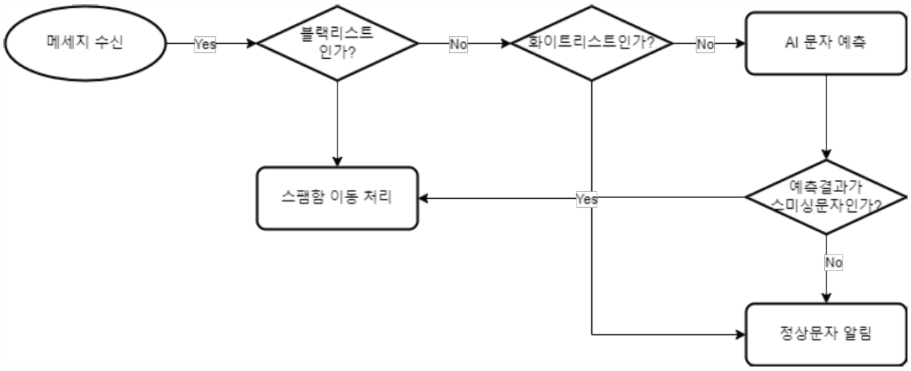


그림 2 초기 구성 플로우차트

스미싱 방지 어플리케이션은 설정한 목표를 이루기 위하여 구현한다. 먼저 메시지를 수신하고 블랙리스트인지 판단하고 AI와 바이러스 토탈의 검사를 통하여 문자를 악성여부를 예측한다. 이때, 악성문자일 경우 스팸메세지함으로 이동 처리하며, 정상적인 문자일 경우 정상적인 메시지 보관함에서 확인할 수 있다. 추가적으로 tfidifVec에 학습되지 않은 메시지가 수신될 경우 사용자들의 신고를 통하여 학습이 되고 추후 이와 유사한 메시지가 수신될 경우 필터링을 한다.

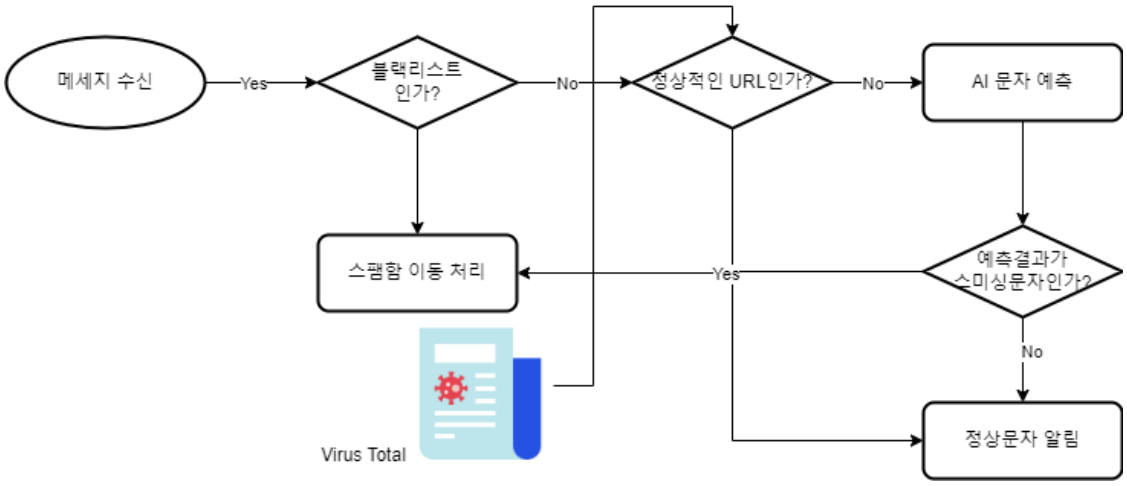


그림 6 변경된 플로우 차트

초기에는 블랙리스트와 화이트 리스트를 탐지하였으나 잘못된 화이트리스트의 위험성과 데이터 수집의 문제로 화이트리스트의 기능을 제거하였으며 메시지 내용은 정상적이나 sms속 url 또한 악성 url의 가능성이 있다고 판단하여 바이러스 토탈 API를 연동하여 url을 검사하는 기능을 추가하였다.

1. 감성분석

여러 가지 NLP분석기법중 감성분석기법을 사용하였으며 sklearn라이브러리를 사용하였다. 사용한 이유는 에 따라서 스미싱 문자가 될 수 있고, 스미싱 문자는 링크로 유도하는 메시지를 가지고 있기 때문이다.

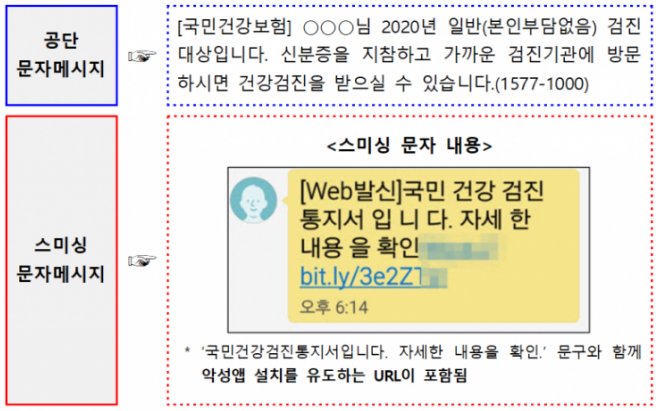


그림 3 스미싱 문자 이미지

예를 들어 보자면, 그림3과 같이 스미싱 문자 메시지랑 비슷하지만 유도하지 않는 메시지를 아래와 같이 AI에 입력해 보았다.

```
['국민 건강 검진 통지서 입니다 자세한 내용은 가까운 검진기관에 방분하세요'] ->> 일반적인 문자
```

반대로 입력해보면 다음과 같은 결과가 나온다.

```
['국민 건강 검진 통지서 입니다 자세한 내용을 확인'] ->> 스미싱 의심문자
```

이와 같은 특성을 이용하여 긍정/부정을 판단하고, 이를 스미싱 판단 기준으로 사용하였다.

```
1 best = grid_cv.best_estimator_
2 prd = best.predict(res)
3 from sklearn.metrics import accuracy_score
4
5 print('score : ', round(accuracy_score(df['1'], prd), 4))

  score :  0.9994
```

테스트 케이스기준 99.94%의 정확도를 가지고 있다.

2. Tokenization

TfidfVectorizer은 관련연구 파트에서 언급했었던 tf-idf를 이용하는 구현하는 방식 중의 하나이다. 학습을 위해 KISA와 데이터공유센터에서 데이터를 수집하여 정제 및 tokenization하여 dataset을 만들었다. 이 dataset을 통해 TfidfVectorizer모델을 학습하였다. 이 모델에 학습된 스미싱 문자와 정상적인 메세지를 이용하여 스미싱, 정상문자 군집을 대표하는 벡터로 이용하였다.

2. 요구 사항

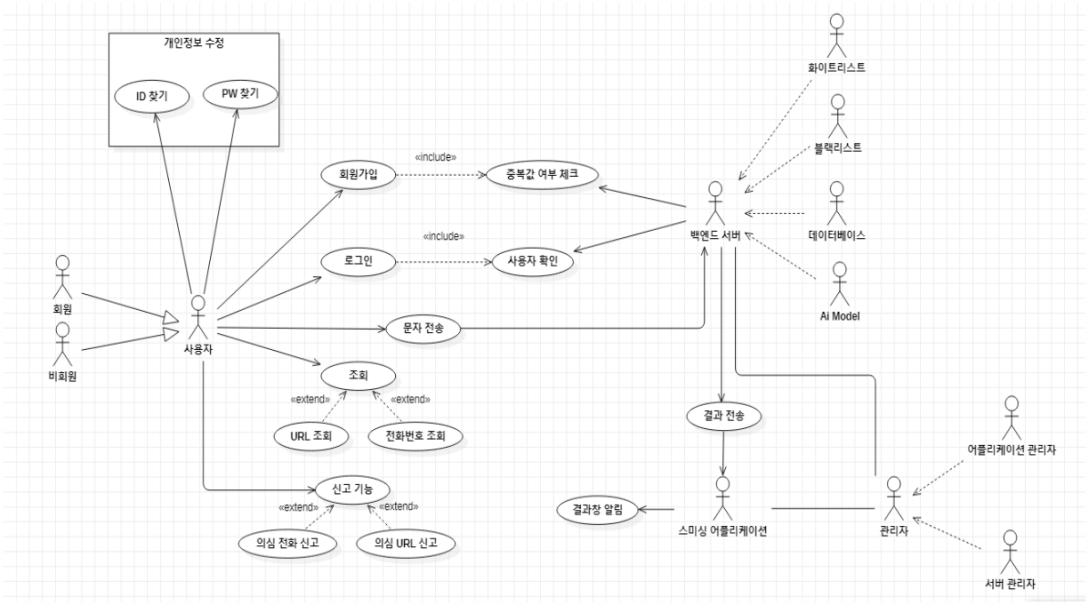


그림 8 프로그램 유스케이스

[유스케이스 명세서]

유스케이스 명	ID 찾기
개요	사용자가 자신이 회원가입이 되어 있는지 확인하거나 아이디를 잊어버린 경우에 사용자의 일부 개인 정보와 본인 인증을 통해 갖고자 하는 아이디를 찾을 수 있도록 한다
관련액터	사용자, 스미싱 어플리케이션
선행조건	서버에 등록된 사용자이어야 한다.
후행조건	요청한 사용자에게 ID의 값을 반환한다
이벤트 흐름	
정상 흐름	1. 사용자는 ID 찾기를 선택한다 -회원 가입 시 입력했던 정보를 입력 후, “찾기” 혹은 “취소”를 선택 2. ID 값을 받는다. -“ID는 xxxxx입니다” 메시지 출력
대안 흐름	1. 사용자가 ID 찾기 과정을 취소 -정보를 입력하지 않고, “취소”를 선택하여 이전 페이지로 돌아간 후, “취소되었습니다” 메시지 출력
비기능적 요구사항	사용자의 개인정보를 바탕으로 옳바른 ID값이 출력되어야 한다

유스케이스 명	회원가입
개요	사용자는 자신의 개인정보(이름, 이메일, 전화번호등)를 입력하여 DB에 ID와 PW를 등록한다
관련액터	사용자, 스미싱 어플리케이션
선행조건	어플리케이션이 정상적으로 설치되어 있어야 한다.
후행조건	정상적으로 DB에 ID와 PW가 등록되어 로그인에 사용이 가능하다
이벤트 흐름	
정상 흐름	1. 사용자는 회원가입을 선택한다. -회원가입 페이지에서 개인정보 및 ID, PW를 입력한다. 2. ID와 PW의 사용이 가능하다. -“회원가입이 성공하였습니다” 메시지 출력
대안 흐름	1. 사용자가 회원가입 과정을 취소 -정보를 입력하지 않고, “이전”을 선택하여 이전 페이지로 돌아간 후, “취소되었습니다” 메시지 출력
비기능적 요구사항	사용자의 옳바른 개인정보및 ID, PW가 DB에 저장되어야 한다.

유스케이스 명	PW 찾기
개요	사용자가 자신이 회원가입이 되어 있는지 확인하거나 비밀번호를 잊어버린 경우에 사용자의 일부 개인 정보와 본인 인증을 통해 갖고자 하는 비밀번호를 찾을 수 있도록 한다
관련액터	사용자, 스미싱 어플리케이션
선행조건	서버에 등록된 사용자이어야 한다.
후행조건	요청한 사용자에게 PW의 값을 반환한다
이벤트 흐름	
정상 흐름	1. 사용자는 PW 찾기를 선택한다 -회원 가입 시 입력했던 정보 및 ID를 입력 후, “찾기” 혹은 “취소”를 선택 2. PW 값을 받는다. -“PW는 xxxxx입니다” 메시지 출력
대안 흐름	1. 사용자가 PW 찾기 과정을 취소 -정보를 입력하지 않고, “취소”를 선택하여 이전 페이지로 돌아간 후, “취소되었습니다” 메시지 출력
비기능적 요구사항	사용자의 개인정보를 바탕으로 옳바른 PW값이 출력되어야 한다

유스케이스 명	로그인
개요	어플리케이션에 기능을 사용자들이 원활하게 이용하기 위해 로그인 기능을 지원한다
관련액터	사용자, 스미싱 어플리케이션
선행조건	1. 회원가입이 정상적으로 완료되어 있어야 한다. 2. 입력한 ID,PW가 회원가입 정보와 일치하는지 확인되어야 한다.
후행조건	어플리케이션의 기능(조회, 신고, 알림) 사용 가능
이벤트 흐름	
정상 흐름	1. 사용자는 ID와 PW를 입력한다 -회원가입 시 입력했던 ID와 PW를 입력하여 등록된 정보와 일치하면 로그인 가능 2. 로그인을 선택한다 -어플리케이션의 정상적으로 로그인되어 이용이 가능
대안 흐름	1. 등록되지 않은 로그인 -입력한 ID와 PW의 값이 서버에 존재하지 않는다면 회원가입으로 이동 2. 잘못된 정보의 로그인 -입력한 ID와 PW의 값이 서버에 등록된 것과 다르다면 ID / PW 찾기로 이동
비기능적 요구사항	서버에 등록된 사용자의 정보 값(ID와 PW)이 입력한 정보값과 같은지 혹은 없거나 다른지를 확인하고, 적절한 페이지로 이동이 가능하게 한다.

유스케이스 명	URL 조회
개요	확인을 원하는 URL를 어플리케이션을 통해 검색 및 조회를 가능하게 하고, 서버는 조회된 URL의 안전성을 확인한 후, 사용자에게 적절한 알림 메시지를 출력한다.
관련액터	사용자, 스미싱 어플리케이션, 백엔드 서버
선행조건	1. 로그인이 정상적으로 완료되어 있어야 한다. 2. 블랙리스트, 화이트리스트가 정상적으로 작동 가능하여야 한다.
후행조건	서버는 사용자에게 조회한 URL을 확인하여 안전성의 유무를 출력해주고, 스미싱 의심 URL이면 백엔드-블랙리스트 서버에 등록한다.
이벤트 흐름	
정상 흐름	1. 사용자는 확인이 필요한 URL을 URL 조회를 선택해 입력한다. -URL을 입력한 후, “검색” 버튼을 입력하여 조회를 한다. 2. 입력한 URL의 조회 결과를 확인할 수 있다. -“입력하신 URL www.xxx.xxxx는 스미싱 의심 URL입니다. / 의심 URL이 아닙니다 ” 메시지 출력
대안 흐름	1. 사용자가 조회 과정을 취소 -정보를 입력하지 않고, “이전”을 선택하여 이전 페이지로 돌아감
비기능적 요구사항	작성한 전화번호가 스미싱 의심 URL일 경우 블랙리스트 서버에 전송되어 저장되어야 한다.

유스케이스 명	전화번호 조회
개요	확인을 원하는 전화번호를 어플리케이션을 통해 검색 및 조회를 가능하게 하고, 서버는 조회된 전화번호의 안전성을 확인한 후, 사용자에게 적절한 알림 메시지를 출력한다.
관련액터	사용자, 스미싱 어플리케이션, 백엔드 서버
선행조건	1. 로그인이 정상적으로 완료되어 있어야 한다. 2. 블랙리스트, 화이트리스트가 정상적으로 작동 가능하여야 한다.
후행조건	서버는 사용자에게 조회한 전화번호를 확인하여 안전성의 유무를 출력해주고, 스미싱 의심 전화번호면 백엔드-블랙리스트 서버에 등록한다.
이벤트 흐름	
정상 흐름	1. 사용자는 확인이 필요한 전화번호를 전화번호 조회를 선택해 입력한다. -전화번호를 입력한 후, “검색” 버튼을 입력하여 조회를 한다. 2. 입력한 전화번호의 조회 결과를 확인할 수 있다. -“입력하신 전화번호 xxx-xxxx-xxxx/xxxx-xxxx는 스미싱 의심 번호입니다. / 의심 번호가 아닙니다 ” 메시지 출력
대안 흐름	1. 사용자가 조회 과정을 취소 -정보를 입력하지 않고, “이전”을 선택하여 이전 페이지로 돌아감
비기능적 요구사항	작성한 전화번호가 스미싱 의심 번호일 경우 블랙리스트 서버에 전송되어 저장되어야 한다.

유스케이스 명	의심 전화 신고
개요	사용자가 스미싱 관련 의심이 되는 전화번호를 직접 어플리케이션을 통하여 서버에 등록할 수 있다.
관련액터	사용자, 스미싱 어플리케이션, 백엔드 서버, 관리자
선행조건	1. 사용자 로그인이 되어있어야 한다. 2. 신고에 따른 근거 자료가 제시되어야 한다.
후행조건	관리자는 해당 신고 내역을 보고, 블랙리스트에 등록 여부를 결정한다.
이벤트 흐름	
정상 흐름	1. 사용자는 전화번호 신고 페이지로 이동한다. 2. 내용을 작성하고 “등록” 버튼을 통하여 신고 글을 등록한다. -신고할 전화번호 및 신고 근거(사진, 음성 파일 등)를 첨부하여 작성
대안 흐름	1. 사용자가 신고 과정을 취소 -정보를 입력하지 않고, “이전”을 선택하여 이전 페이지로 돌아감 2. 근거 미등록 -사용자가 근거 자료를 등록하지 않는 경우 “근거 사유가 필요합니다” 메시지 출력 후, 글 작성 불가
비기능적 요구사항	관리자는 신고 내역을 확인한 후, 안전성 유무를 검토하여 블랙리스트에 내용을 신규 기입한다.

3. 시스템 구성

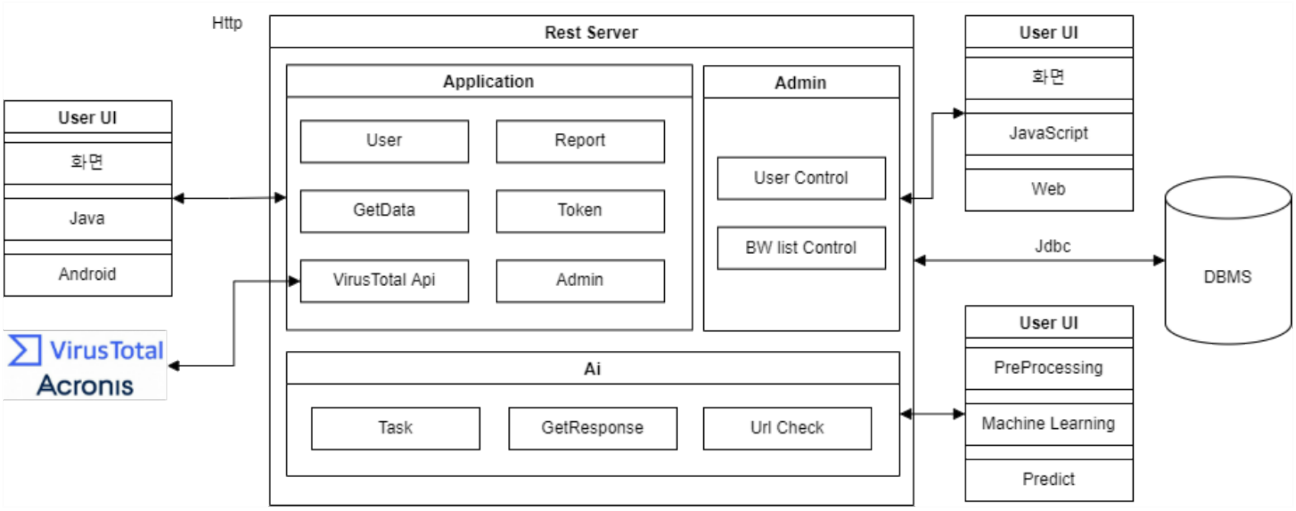


그림 6 시스템 구성도

전체 시스템 구성은 MVC모델을 이용하여 중앙집중 처리방식을 이용할 예정이며 RestApi를 이용한 Http통신을 통해 서버와 통신한다.

1. 백엔드 구성 :

백엔드 개발언어로는 Node.js를 기반으로한 Express프레임워크를 이용하여 개발 편의성을 높이고 RestApi를 구현한다.

데이터베이스는 MySql을 이용하고 단일 데이터 베이스를 이용한다.

2. AI :

웹페이지의 바이러스를 검사하는 방법은 VirusTotal의 OpenApi와 연동하여 바이러스 검사를 실시한다. AI 처리 시스템에서 URL 바이러스 검사와 메시지 판단을 동시에 처리 한다. flask를 이용하여 RestApi를 구현한다.

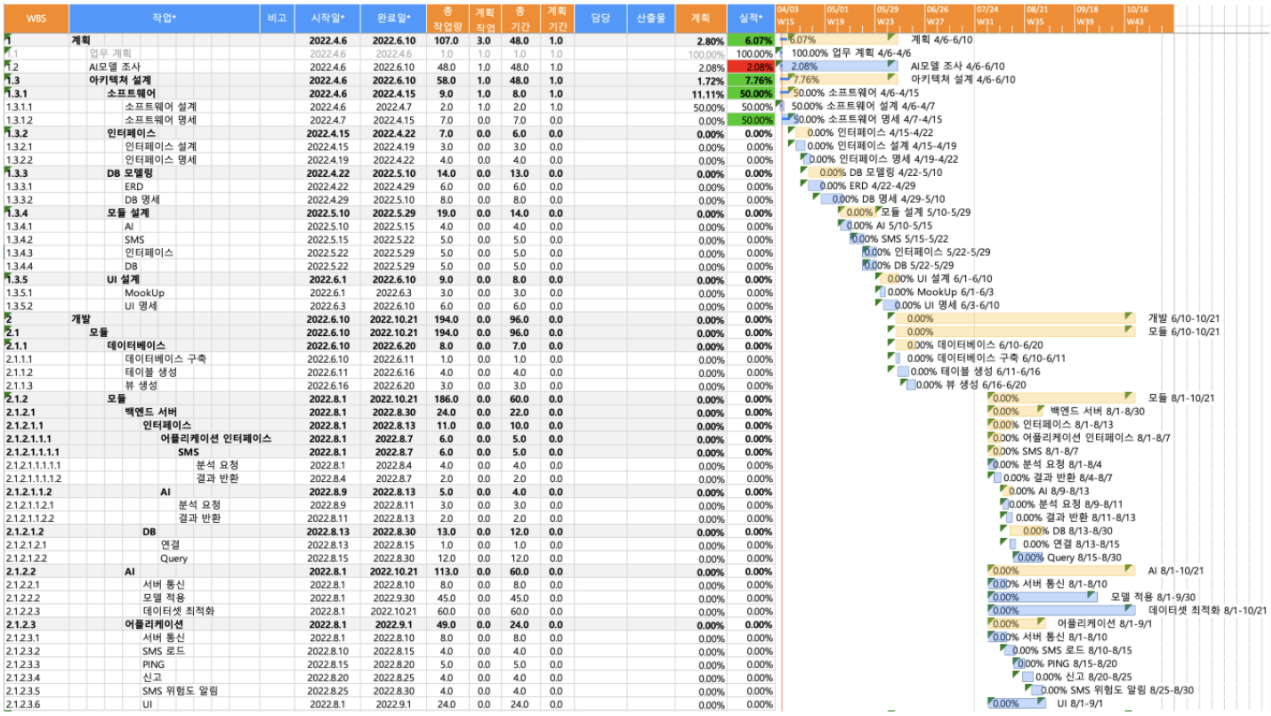
3. 프론트엔드 구성 :

프론트엔드 개발언어로는 Kotlin을 이용한다. SMS 메시지를 받아와서 AI서버와 직접 통신하여 결과를 받아오고, 회원관련한 데이터는 백엔드 RestApi와 통신하여 데이터 통신한다.

4. VirusTotal :

바이러스 토탈의 API키 사용제한이 있으므로 여러 개의 키를 이용하는 시스템을 개발한다.

3. 프로젝트 추진



WBS	작업*	비고	시작일*	완료일*	총 작업량	계획 작업	총 기간	계획 기간	담당	산출물	계획	실적*
1	계획		2022.4.6	2022.6.10	107.0	3.0	48.0	1.0			2.80%	6.07%
1	업무 계획		2022.4.6	2022.4.6	1.0	1.0	1.0	1.0			100.00%	100.00%
2	AI모델 조사		2022.4.6	2022.6.10	48.0	1.0	48.0	1.0			2.08%	2.08%
1.3	아키텍쳐 설계		2022.4.6	2022.6.10	58.0	1.0	48.0	1.0			1.72%	7.76%
1.3.1	소프트웨어		2022.4.6	2022.4.15	9.0	1.0	8.0	1.0			11.11%	50.00%
1.3.1.1	소프트웨어 설계		2022.4.6	2022.4.7	2.0	1.0	2.0	1.0			50.00%	50.00%
1.3.1.2	소프트웨어 명세		2022.4.7	2022.4.15	7.0	0.0	7.0	0.0			0.00%	50.00%
1.3.2	인터페이스		2022.4.15	2022.4.22	7.0	0.0	6.0	0.0			0.00%	0.00%
1.3.2.1	인터페이스 설계		2022.4.15	2022.4.19	3.0	0.0	3.0	0.0			0.00%	0.00%
1.3.2.2	인터페이스 명세		2022.4.19	2022.4.22	4.0	0.0	4.0	0.0			0.00%	0.00%
1.3.3	DB 모델링		2022.4.22	2022.5.10	14.0	0.0	13.0	0.0			0.00%	0.00%
1.3.3.1	ERD		2022.4.22	2022.4.29	6.0	0.0	6.0	0.0			0.00%	0.00%
1.3.3.2	DB 명세		2022.4.29	2022.5.10	8.0	0.0	8.0	0.0			0.00%	0.00%
1.3.4	모듈 설계		2022.5.10	2022.5.29	19.0	0.0	14.0	0.0			0.00%	0.00%
1.3.4.1	AI		2022.5.10	2022.5.15	4.0	0.0	4.0	0.0			0.00%	0.00%
1.3.4.2	SMS		2022.5.15	2022.5.22	5.0	0.0	5.0	0.0			0.00%	0.00%
1.3.4.3	인터페이스		2022.5.22	2022.5.29	5.0	0.0	5.0	0.0			0.00%	0.00%
1.3.4.4	DB		2022.5.22	2022.5.29	5.0	0.0	5.0	0.0			0.00%	0.00%
1.3.5	UI 설계		2022.6.1	2022.6.10	9.0	0.0	8.0	0.0			0.00%	0.00%
1.3.5.1	MookUp		2022.6.1	2022.6.3	3.0	0.0	3.0	0.0			0.00%	0.00%
1.3.5.2	UI 명세		2022.6.3	2022.6.10	6.0	0.0	6.0	0.0			0.00%	0.00%
2	개발		2022.6.10	2022.10.21	194.0	0.0	96.0	0.0			0.00%	0.00%
2.1	모듈		2022.6.10	2022.10.21	194.0	0.0	96.0	0.0			0.00%	0.00%
2.1.1	데이터베이스		2022.6.10	2022.6.20	8.0	0.0	7.0	0.0			0.00%	0.00%
2.1.1.1	데이터베이스 구축		2022.6.10	2022.6.11	1.0	0.0	1.0	0.0			0.00%	0.00%
2.1.1.2	테이블 생성		2022.6.11	2022.6.16	4.0	0.0	4.0	0.0			0.00%	0.00%
2.1.1.3	뷰 생성		2022.6.16	2022.6.20	3.0	0.0	3.0	0.0			0.00%	0.00%
2.1.2	모듈		2022.8.1	2022.10.21	186.0	0.0	60.0	0.0			0.00%	0.00%
2.1.2.1	백엔드 서버		2022.8.1	2022.8.30	24.0	0.0	22.0	0.0			0.00%	0.00%
2.1.2.1.1	인터페이스		2022.8.1	2022.8.13	11.0	0.0	10.0	0.0			0.00%	0.00%
2.1.2.1.1.1	어플리케이션 인터페이스		2022.8.1	2022.8.7	6.0	0.0	5.0	0.0			0.00%	0.00%
2.1.2.1.1.1.1	SMS		2022.8.1	2022.8.7	6.0	0.0	5.0	0.0			0.00%	0.00%
2.1.2.1.1.1.1.1	분석 요청		2022.8.1	2022.8.4	4.0	0.0	4.0	0.0			0.00%	0.00%
2.1.2.1.1.1.1.2	결과 반환		2022.8.4	2022.8.7	2.0	0.0	2.0	0.0			0.00%	0.00%
2.1.2.1.1.2	AI		2022.8.9	2022.8.13	5.0	0.0	4.0	0.0			0.00%	0.00%
2.1.2.1.1.2.1	분석 요청		2022.8.9	2022.8.11	3.0	0.0	3.0	0.0			0.00%	0.00%
2.1.2.1.1.2.2	결과 반환		2022.8.11	2022.8.13	2.0	0.0	2.0	0.0			0.00%	0.00%
2.1.2.1.2	DB		2022.8.13	2022.8.30	13.0	0.0	12.0	0.0			0.00%	0.00%
2.1.2.1.2.1	연결		2022.8.13	2022.8.15	1.0	0.0	1.0	0.0			0.00%	0.00%
2.1.2.1.2.2	Query		2022.8.15	2022.8.30	12.0	0.0	12.0	0.0			0.00%	0.00%
2.1.2.2	AI		2022.8.1	2022.10.21	113.0	0.0	60.0	0.0			0.00%	0.00%
2.1.2.2.1	서버 통신		2022.8.1	2022.8.10	8.0	0.0	8.0	0.0			0.00%	0.00%
2.1.2.2.2	모델 적용		2022.8.1	2022.9.30	45.0	0.0	45.0	0.0			0.00%	0.00%
2.1.2.2.3	데이터셋 최적화		2022.8.1	2022.10.21	60.0	0.0	60.0	0.0			0.00%	0.00%
2.1.2.3	어플리케이션		2022.8.1	2022.9.1	49.0	0.0	24.0	0.0			0.00%	0.00%
2.1.2.3.1	서버 통신		2022.8.1	2022.8.10	8.0	0.0	8.0	0.0			0.00%	0.00%
2.1.2.3.2	SMS 로드		2022.8.10	2022.8.15	4.0	0.0	4.0	0.0			0.00%	0.00%
2.1.2.3.3	PING		2022.8.15	2022.8.20	5.0	0.0	5.0	0.0			0.00%	0.00%
2.1.2.3.4	신고		2022.8.20	2022.8.25	4.0	0.0	4.0	0.0			0.00%	0.00%
2.1.2.3.5	SMS 위험도 알림		2022.8.25	2022.8.30	4.0	0.0	4.0	0.0			0.00%	0.00%
2.1.2.3.6	UI		2022.8.1	2022.9.1	24.0	0.0	24.0	0.0			0.00%	0.00%

그림 13 일정 전체를 표현한 WBS

프로젝트 진행일정은 그림13의 WBS를 기준으로 진행하였다.

2022.04~2022.06 : 계획 및 설계단계
2022.06~2022.07 : 데이터베이스 구축
2022.08~2022.09 : 공통모듈 개발
2022.06~2022.10 : AI서버 개발(일정변경)
2022.08~2022.10 : 어플리케이션 개발

4. 구현 결과

그림13은 AI 서버로, Flask API를 통하여 구현하였다. 아래 그림 14의 이미지의 코드를 통해서 SMS의 검사결과를 반환하며 악성메세지일 경우 message_confirm의 값을 1을 반환하고 정상 메세지일 경우 0을 반환한다.

```
st = input('text>>')
st = re.compile(r'[가-힣]+').findall(st)
st = [" ".join(st)]
st_tfidf = tfidf.transform(st)
st_predict = best.predict(st_tfidf)
if st_predict==0:
    print(st, "->> 일반적인 문자")
else:
    print(st, "->> 스미싱 의심문자")
```

```
['국민 건강 검진 통지서 입니다 자세한 내용을 확인'] ->> 스미싱 의심문자
```

그림 15

아래의 코드를 통해 바이러스 토탈 검사를 진행하고 메시지 검사와 동일하게 결과를 반환한다.

```
url_confirm=0
url = 'https://www.virustotal.com/vtapi/v2/url/scan'
params = {'apikey': my_apikey, 'url': scan_url}
response_scan = requests.post(url, data=params)
result_scan = response_scan.json()
scan_id = result_scan['scan_id']  # 결과를 출력을 위해 scan_id 값 저장
url_report = 'https://www.virustotal.com/vtapi/v2/url/report'
url_report_params = {'apikey': my_apikey, 'resource': scan_id}
response_report = requests.get(url_report, params=url_report_params)
report = response_report.json()  # 결과 값을 report에 json형태로 저장
report_scans = report.get('scans')  # scans 값 저장
report_scans_vendors = list(report['scans'].keys())  # Vendor 저장
for vendor in report_scans_vendors:
    outputs_result = report_scans[vendor].get('result')
    if outputs_result != 'clean site':
        if outputs_result != 'unrated site':
            url_confirm = 1
requests.post('http://localhost:8080/ai/insert_url', data={'pid':pid, '_confirm':ur
 .json()
```

그림 16

데이터셋은 총 77,553개로 악성 SMS는 KT 데이터마켓을 통해 다운로드 받은 KISA의 데이터셋을 이용하였고, 정상 SMS는 공공데이터 포털의 일상대화 자연어 데이터셋을 이용하였다. label 값의 1은 악성 SMS를 의미하고 0은 정상적인 메시지를 의미한다.

그림 13 AI RestServer

```
df = pd.read_csv('sentiment.csv')
df['1'].value_counts()

1    45135
0    32418
Name: 1, dtype: int64
```

데이터베이스는 account, url, work, phone, black list, report, token으로 구성하였으며 phone table의 pk인 pid를 기준으로 정규화를 진행하였고 그 결과는 그림 14의 ERD이며 이를 구현하였다.

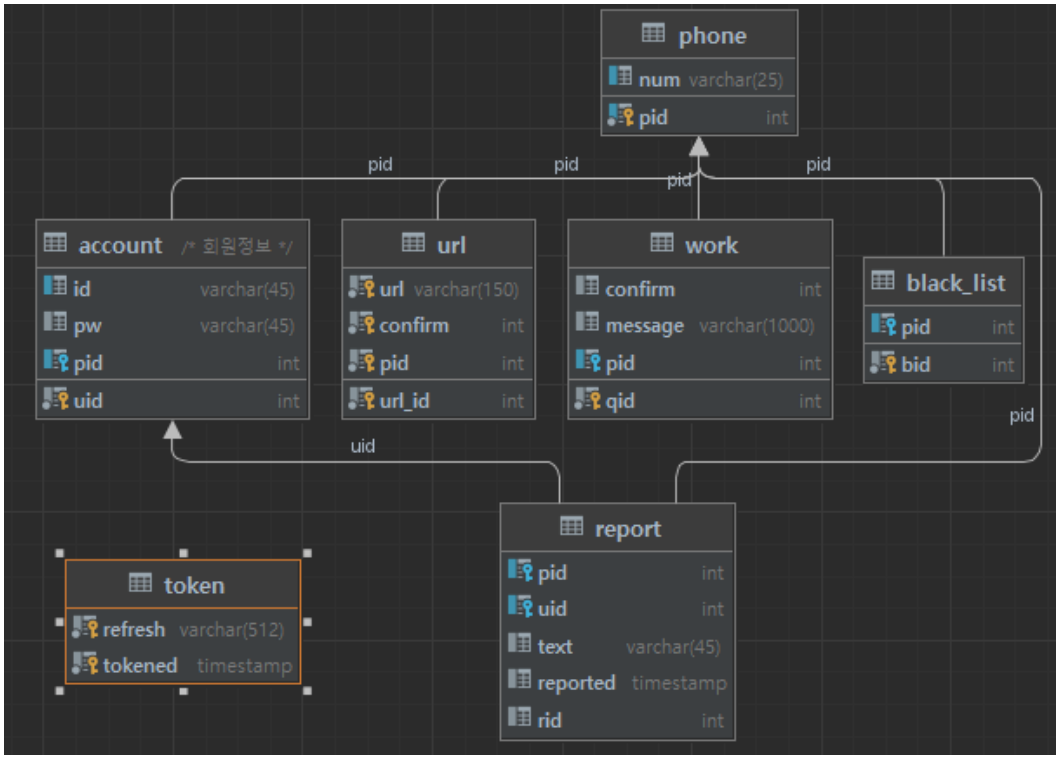
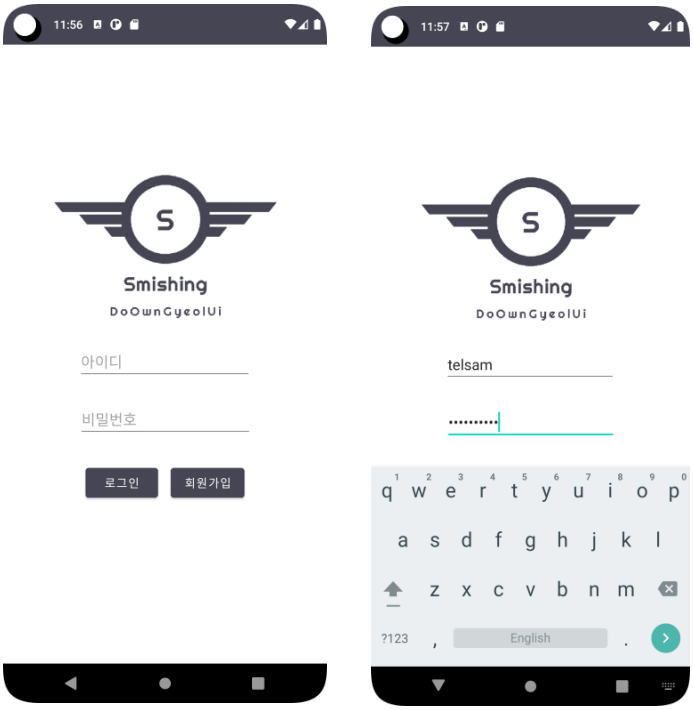
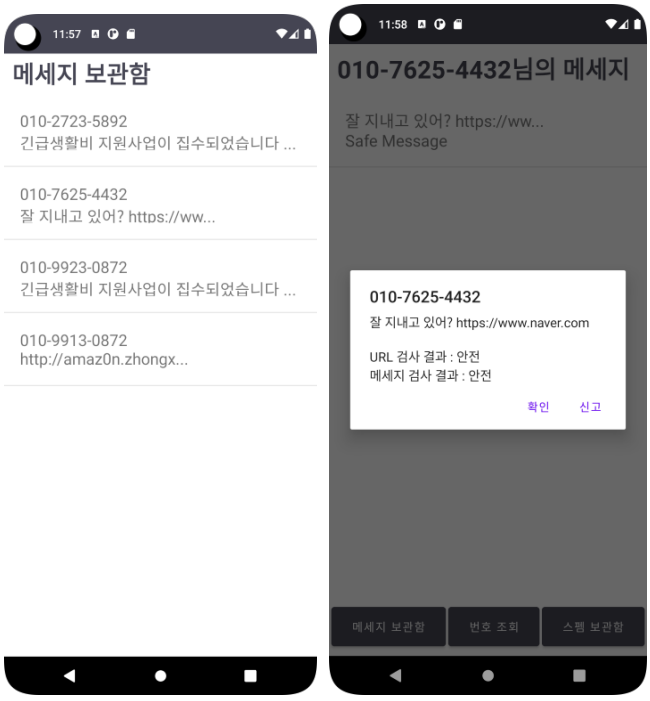


그림 14 DB ERD

1. 앱을 실행한뒤 로그인 과정을 거쳐 서비스를 이용하도록 설계하였다. 이후 나올 기능 중 메세지 신고 기능을 통하여 일정 시간 내 자동으로 Blacklist 등록 하는 기능이 있는데 이를 악용하는 악성유저를 방지하기 위해 로그인 기능을 구현 하였다.

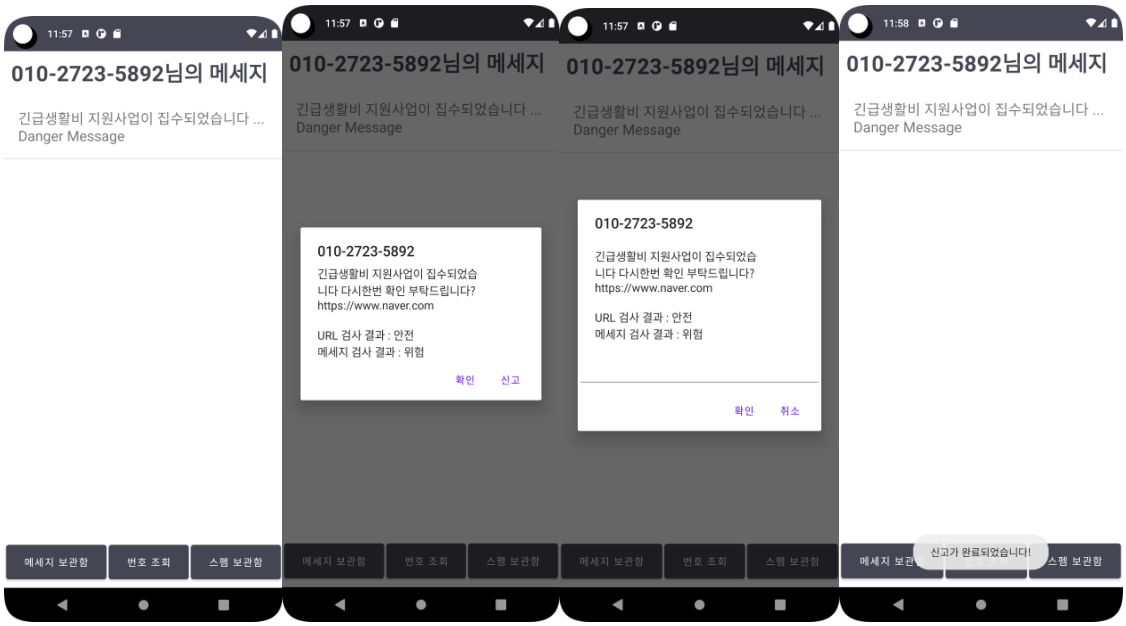


앱을 실행하면 메시지 보관함으로 이동하게되고메세지 목록들을 확인할 수 있고 정상적인 메세지인 경우 메시지 아래에 Safe Message라는 문구를 확인할 수 있다.



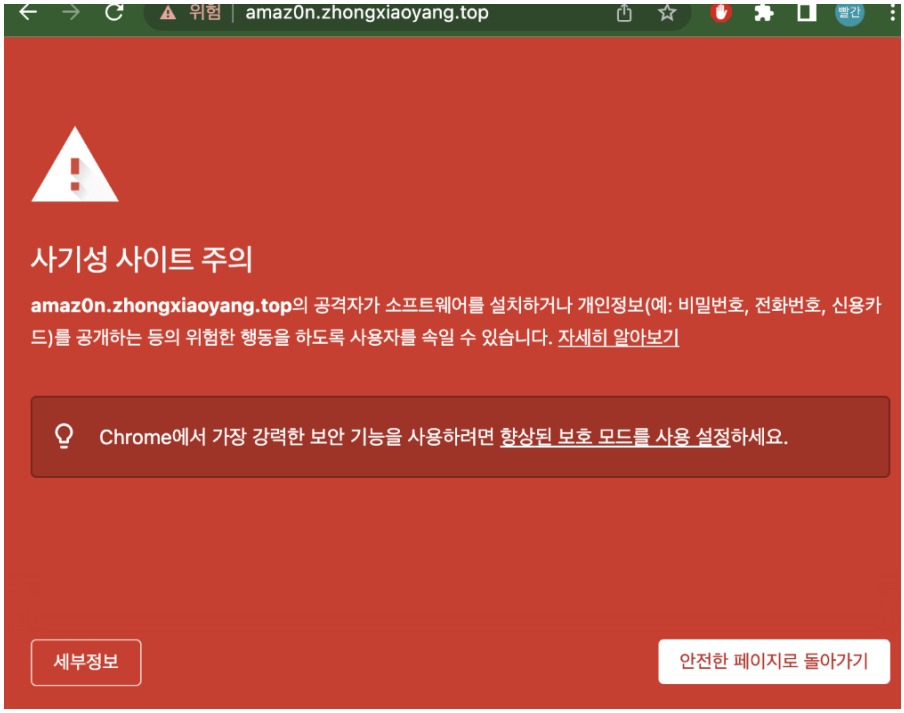
2020년 6월 기사로 알려진 스미싱 문자를 AI의 성능 테스트를 위해 입력하였고, 첨부링크는 스미싱 문자가 아닌 정상적인 주소를 입력하였다.

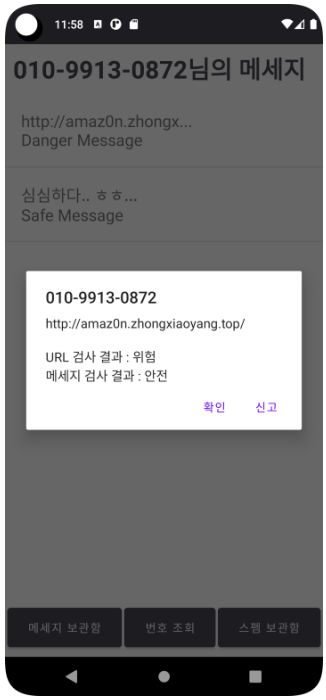
해당 문자를 수신한 사용자는 SMS신고기능을 통해 신고할 수 있고, 아래 이미지는 신고 과정이다.



하루에 10회 이상 신고가 누적될 경우 자동적으로 블랙리스트에 등록이 되며, 블랙리스트 된 번호를 사용자가 직접 확인 할 수 있다.

바이러스 토탈 기능을 이용하기 위해 실재로 존재하는 피싱사이트와 정상적인 문자를 입력하였고 메시지 보관함에서 Danger Message라는 문구를 확인할 수 있다. 메시지의 상세 내용 보기를 하면 URL 검사 결과를 확인 할 수 있다.





5. 결론 및 기대효과

프로젝트는 어플리케이션을 활용해 스미싱 공격을 효과적으로 탐지, 차단할 수 있는 시스템을 구현하고자 하였다. 그 결과 프로그램 유스케이스와 시스템 플로우차트, 데이터베이스 ER다이어그램를 기반으로 진행된 프로젝트를 통해 테스트 케이스 기준 99.94%의 탐지율을 가진 스미싱 공격을 탐지 할 수 있는 어플리케이션을 구현하였다. node.js와 kotlin, flask, mysql 등을 이용하여 개발을 진행했으며, 이용자가 문자메세지 수신 시 위 언어를 통해 개발한 API와 감성분석기법을 이용한 AI모델, VirusTotal open API를 통해 스미싱을 여부를 탐지하는 서비스를 이용자에게 제공한다.

감정분석 기반 AI모델을 사용하여 정확한 탐지로 스미싱 문자를 사전에 차단하고, 바이러스 토탈과 같이 URL을 검사하여 해당 URL 링크가 악성 링크인지 확인한다. 검사를 통해 해당 도메인에 국내 국내외 보안 기업에서 탐지한 내역을 조회할 수 있고, 다운로드된 파일 중 악성파일이 포함되어 있거나, 실제 허가된 서비스에서 제공하는 URL주소인지를 확인하여 볼 수 있다.

최근 다시 증진되는 코로나 바이러스로 방역지침이나 확진자 추이 등을 미끼로하는 스미싱 문자뿐 아니라 악성 앱 설치 유도, 악성 앱 주소 등을 포함한 스미싱문자까지 전반적인 스미싱 문자메시지 및 문자 메시지에 포함되는 악성 URL등을 차단하여 일반 사용자에서 더 나아가 정보 취약 계층이 보다 안전하고, 편리한 Network Services를 이용에 대해 초점을 맞춘다. 또한 스미싱 범죄에 피해를 줄이고 사전에 차단할 수 있는 효과까지 생각하며 대응되는 보안 어플리케이션을 제작함으로써 피해를 사전에 차단함에 중점을 둔다. 해당 어플리케이션은 AI를 기반으로 탐지 시스템을 구축하여 보다 신속하고, 정확한 탐지율로 악성 문자 및 URL을 탐지하고 차단하여 사용자가 스미싱 범죄에 피해를 보지 않게끔 관리한다.

사용자의 실수 또는 인지하지 못한 상태에서 발생할 수 있는 스미싱 공격의 피해 또한 낮출 수 있을 것이다. 이는 자연스럽게 스미싱 공격으로 이익을 보는 가해자가 점차 감소하는 효과를 가져오며 약 2,200만원으로 추정되는 보이스피싱 악성 앱으로 인한 피해액 평균치를 낮춰 경제적 손실을 방지할 수 있다.

6. 마치며

AI 기반 스미싱 보안 어플리케이션을 제작하면서 서버와 클라이언트, 데이터 베이스들간의 관계에 대한 지식을 넓힐 수 있었고, 실제로 개발을 진행하고 관계를 구축하는 과정을 통해 새로운 경험을 해볼 수 있었다. 자료 조사를 하면서 정보 취약 계층을 대상으로하는 스미싱 범죄가 생각보다 많았고, 이에 따른 피해 금액 또한 상당히 많다는 것을 알게되었다. 완벽한 어플리케이션은 아니지만 시스템의 동작 과정을 이해하고 공부해보면서 보다 깊은 보안 어플리케이션 개발을 하며 이런 악성 범죄나 악성 파일 등에 대해 취약 계층의 피해를 줄일수 있기를 바란다.

참고문헌 및 자료 :

그림1 : 충청뉴스(http://www.ccnnews.co.kr)
스미싱 : https://www.boho.or.kr/cyber/smishing.do
NPL : https://www.oracle.com/kr/artificial-intelligence/what-is-natural-language-processing/
Tfidf : https://ko.wikipedia.org/wiki/Tf-idf
바이러스토탈 엔진 목록 : https://namu.wiki/w/VirusTotal
데이터 과학 기반의 파이썬 빅데이터 분석
감성분석 이미지 : https://medium.com/naver-cloud-platform/
긴급 생활비 접수 : http://www.safetimes.co.kr/news/articleView.html?idxno=82272