0. 기본적인 설문조사

-연령대?

-약국에 얼마나 자주 가는지

-집에 상비약을 두고 있는지

-병원진료 예약하는지 >전화로 한다 >앱으로 한다 >예약 안한다

-약을 잘못 먹은 적이 있는지

>처방 실수

>알레르기 발생

>감기약+타이레놀 등의 중복 복용

>기타

**개인 의료 정보 관리 앱**

개인 맞춤형 복약 정보& 진료기록 관리/ 빠른 병원예약&상담 서비스 제공

1. 취지

* 약물사고 예방 기능 : 약물사고 상담건수가 연간 1000건을 넘어감. 생명과 직결될 수도 있는 부분이기 때문에 처방.조제 시 주의가 필요하고 환자 또한 약 복용법을 반드시 지켜야함.

특히, 약국 접근성이 낮은 곳 거주자, 거동이 불편한 노년층 활용성이 높을 것으로 기대

* 개인 맞춤형 관리 : 과거 의료기록, 복약 데이터(길고 어려운 약 이름. 기억이 안날 수도 있음) 관리, 약 복용 알림
* DUR(Drug Utilization Review): 의사와 약사가 처방 · 조제 시 함께 먹으면 안되는 약, 어린이 · 임신부가 먹으면 안되는 약 등 의약품 정보를 실시간으로 제공해 부적절한 약물 사용을 사전에 점검 · 예방하는 서비스 (건강보험심사평가원 급여 기준 DB나 처방조제 누적DB로 실시간 확인)

1. 시장성 분석

* 사회적 : 기대수명 증가와 함께 건강에 대한 관심 또한 증가
* 기업(제약회사) : 환자들 개인 체질과 병력, 자주 검색하는 증상에 대한 데이터 필요(기업에 데이터 팔 수 있음)

1. 경쟁업체
2. 파프리카케어

* 처방전 기반 질환, 약제에 대한 설명, 의료 정보 제공
* 가족건강관리
* 의약품안전점검(DUR) : 등록된 약제 비교분석하여 임부금기, 병용금기, 동일성분 중복, 동일 효능군 중복 약제가 있을 경우 해당 약제에 대한 알림과 설명 제공
* 사용자 피드백 : 달력추가(병원 방문 기록), 복약알림

1. 아야포

* 복약관리
* 처방전관리 : 촬영하거나 갤러리에서 선택 등록하면 시스템에서 자동으로 분석하여 처방받은 의약품 정보를 알려주고 복약 시간에 알림을 전송
* 병원예약 : 빠른예약, 지정예약
* 병원찾기
* 실시간문의 : 병원과 문의 주고받기
* 가족추가 : 예약정보 쉽게 업로드, 가족별 처방전관리

1. 주요 타겟층
2. 2-30대 : 스마트폰 활용 빈도가 높은 젊은 연령층/ 약물 오남용 비율이 가장 높은 연령층
3. 노년층(50대부터 90대까지) : 병원 혹은 약국 접근성이 낮은 노인분들, 거동이 불편하신 분들이 집에서 쉽게 약 정보와 복용법을 검색하실 수 있도록 함. 약 복용에 관해 궁금한 사항과 추가 증상이 발생했을 때 약을 처방받았던 약국으로 바로 전화 연결 될 수 있는 서비스 제공.(불필요한 방문 방지)
4. 수익구조
5. 광고: 건강보조식품, 바이오시밀러 약, 운동기구 등 의약, 건강에 관련된 상품 광고/ 제약회사, 대형병원 제휴 광고
6. 데이터 판매 : 환자들이 주로 검색한 내용들, 체질, 복약데이터를 제약회사, 건강 관련 상품 판매 회사에 판매 가능

한국기사

잘못복용http://health.chosun.com/site/data/html\_dir/2016/04/05/2016040502011.html

https://ysk6909.tistory.com/148

https://korean.mercola.com/sites/articles/archive/2018/04/05/%EA%B3%BC%EC%9E%89-%EC%B2%98%EB%B0%A9%EC%9D%84-%EB%B0%9B%EB%8A%94-%EB%85%B8%EC%9D%B8%EC%B8%B5.aspx

미국기사

medication error 정의 : https://www.nccmerp.org/about-medication-errors

기사1 : https://www.newlinlaw.com/pharmaceutical-malpractice-faqs/pharmacy-prescription-errors/

잘못복용한 비율: https://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/deaths/bulletins/deathsrelatedtodrugpoisoninginenglandandwales/2018registrations

위와같이, 세계적으로, 잘못된 처방전으로 인해 약을 잘못복용해서 사고로 이어지는 경우가 증가하고 있는 추세입니다.

따라서 우리는, 약을 복용하기 전에, 처방전을 먼저 앱에 업로드해서, 환자의 증상과 실제로 처방받은 약이 환자의 증상에 알맞는 약인지 먼저 확인 후 약을 복용하는것이 환자의 건강에 유의미한 결과를 보여줄 수 있다고 생각합니다.

1. 약 이미지 검색 (네이버 AI 기술 이용) - 민주

네이버 스마트렌즈 기술을 통하여 약 이미지 검색을 할 수 있습니다. 약 한 알만의 이미지로도 약을 매칭시켜 줄 수 있습니다.

카메라로 처방전, 약봉투, 약통, 약포장박스를 글자 인식 -> 약 매칭

포장지 없어도 약 매칭! (약학정보원 데이터베이스 이용)

->약국에서 일해보니, 약 포장지 없이 약의 이름을 찾는 분들이 꽤 계시다. 특히 조제약 경우.

약 데이터 베이스: 약학정보원



초반 처방전, 약봉투, 약통, 약포장지, 약 이미지 데이터를 어디서 모으는가.

병이(가) 표시된 사진

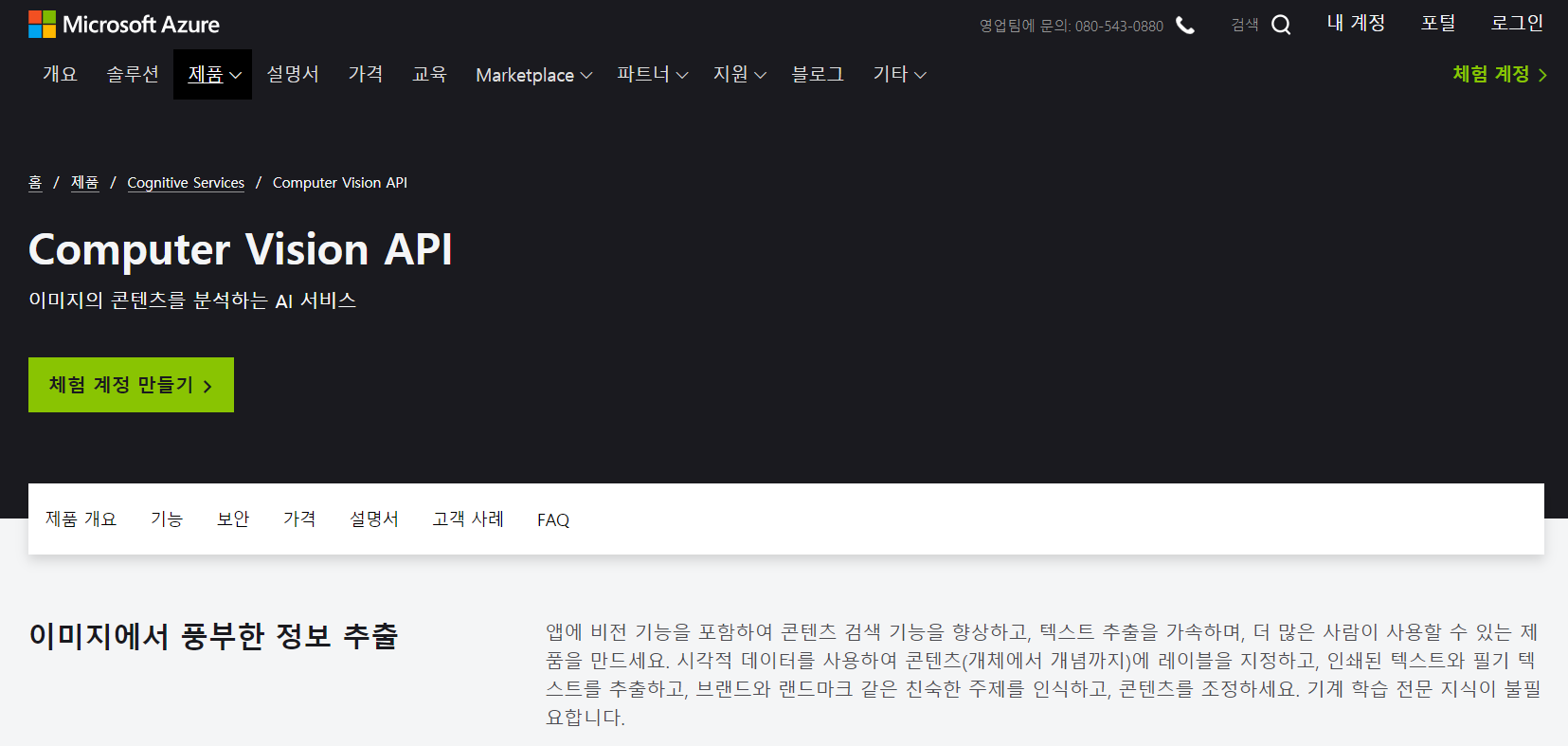
자동 생성된 설명

네이버에는 충분한 데이터가 있다.

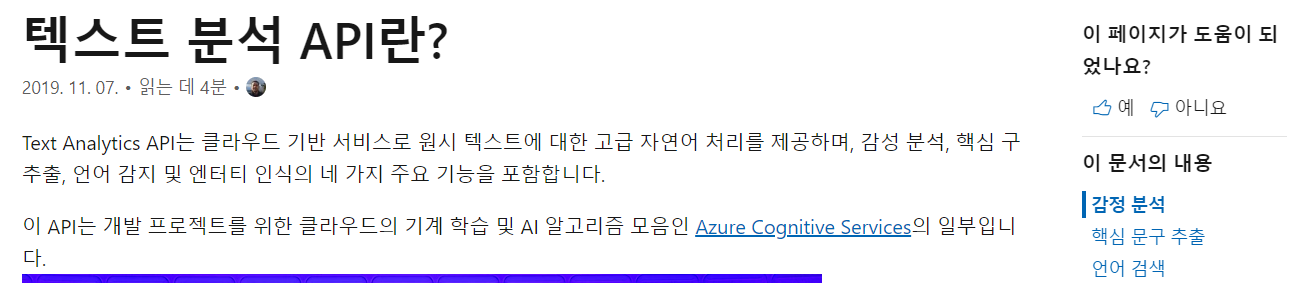
이미지 분석> 네이버 스마트렌즈 기술을 이용하면 충분히 구현 가능하다.

**낱개의 약 이미지 데이터 수집> 직접 사진 찍기..**

마이크로소프트 컴퓨터 비전 api



마이크로소프트 텍스트 분석 api



>vision 기술을 활용하여 텍스트 인식 (Microsoft computer vision api 와 text analytics api의 핵심단어추출 기술 이용) -> 약과 병세 매칭, 부작용, 복용방법 등 매칭

>처방전: 유사앱 존재. 충분히 기술 구현 가능. (파프리카케어)

앱 첫화면에서 카메라 버튼 누르기> 사진찍기

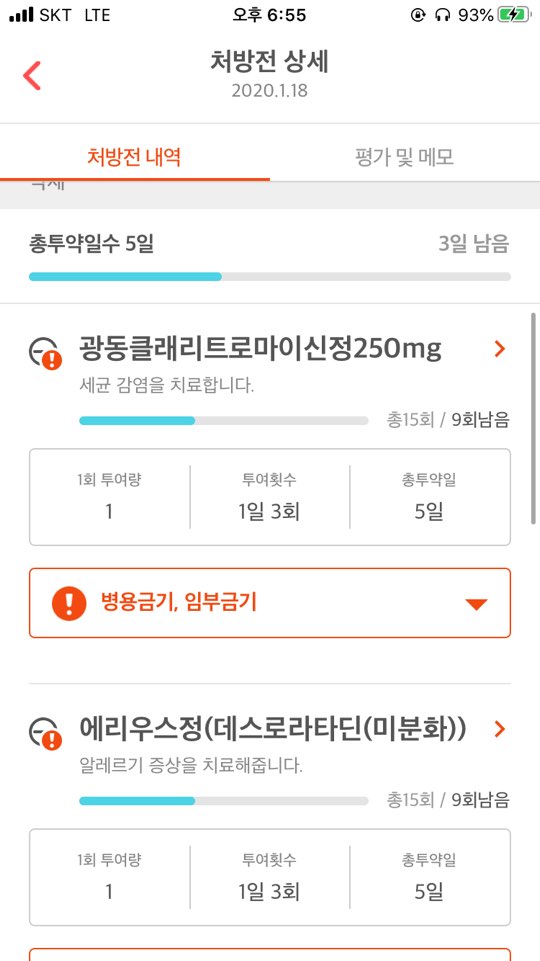
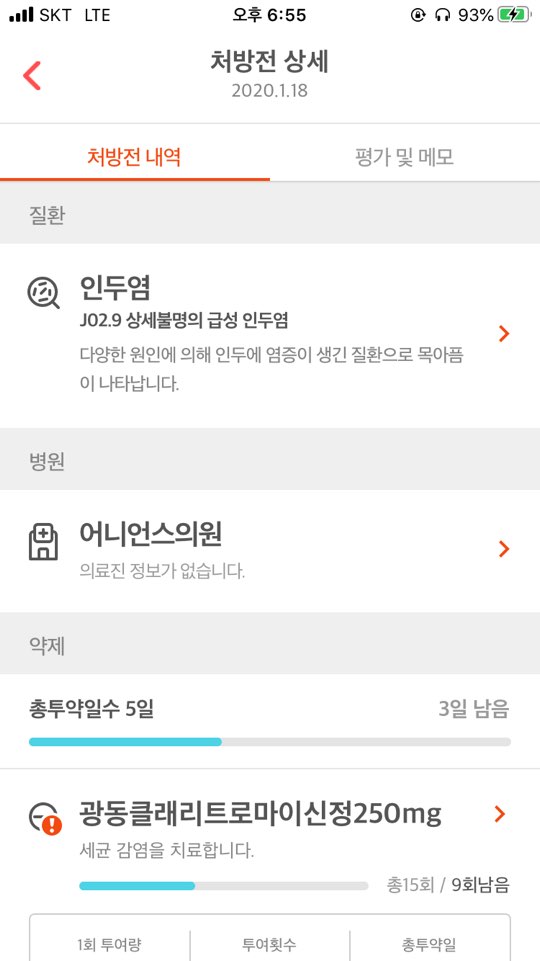
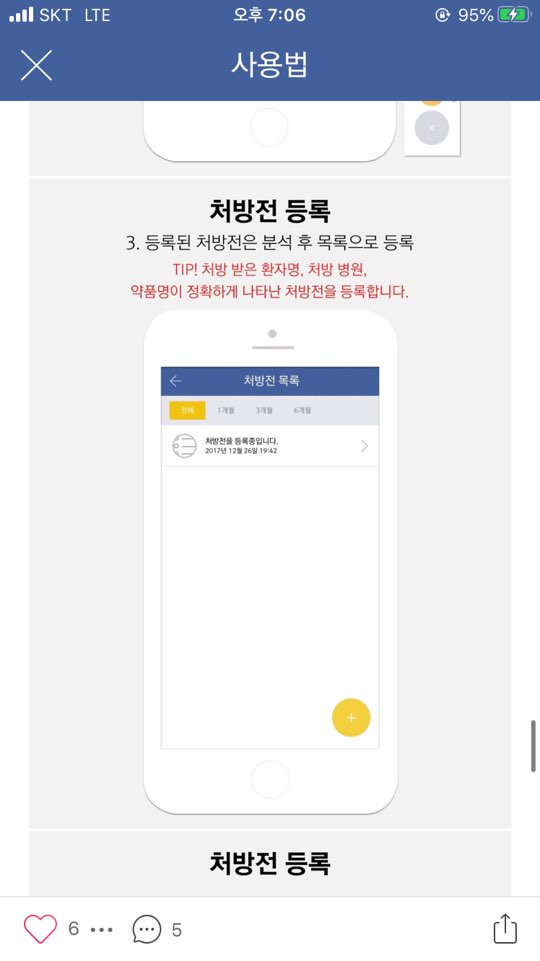
1. 약/약 포장 박스/통: 해당 약에 대한 간단한 설명과 밑에 약학정보원의 검색결과(자세한 내용)

2. 처방전/약봉투: 처방전 혹은 약봉투의 텍스트를 인식하여

-병원, 약국(약봉투만)의 정보(이름, 전화번호 – 병원과 약국 이름 클릭하면 맵으로 위치 확인 가능)

-처방약 이름과 용도, 약 별 생김새(이미지), 복용방법(횟수,일수 등), DUR, 주의사항(임부투여금지, 피임할것, 졸음주의 등), 알림 설정 및 해제 버튼

-알레르기는 흔하지 않은 증상이므로, 메뉴 탭에 따로 알레르기 탭 > 현재 복용약 리스트 > 알레르기 여부 설정.



설문조사>

집에 포장박스나 통 없이 낱개로 떠돌아다니는 약이 있다?

1. 자주 있다

2. 거의 없다

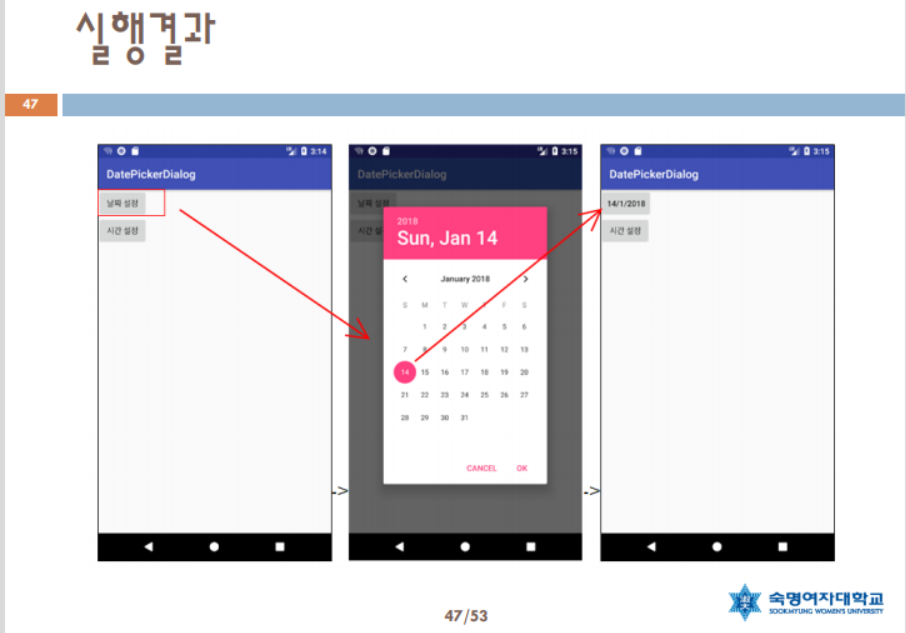
3. 곧장 버린다

1. 달력+약복용알림기능

처방전을 사진으로 찍으면 달력으로 데이터를 저장해주고, 해당 약에 맞추어 자동으로 복용 알림 설정이 됩니다. – 신형

-- 사진으로 찍힌 처방전에서 날짜를 문자로 입력하면 이를 해당 달력에 저장해주고 표시해주는 기능

=> 날짜를 문자로 인식하고 -> 이를 데이터 베이스에 저장한 다음 달력에 저장해주기



-- 오늘의 날짜가 오면 그 약에 대한 복용 알림을 설정하기

◆네이버 API는 네이버 캘린더랑 연동시켜서 해야한다면 사용해야 하지만, 굳이 그게 아니라 이 앱 자체적으로 있는 달력을 이용할 거면 이 API가 아닌 단순 안드로이드와 데이터베이스를 이용해서 사용해도 가능할 것 같다. ( 네이버 API 5000원 )

앱 내에서 자체적으로 만들 경우 예시

1.약복용

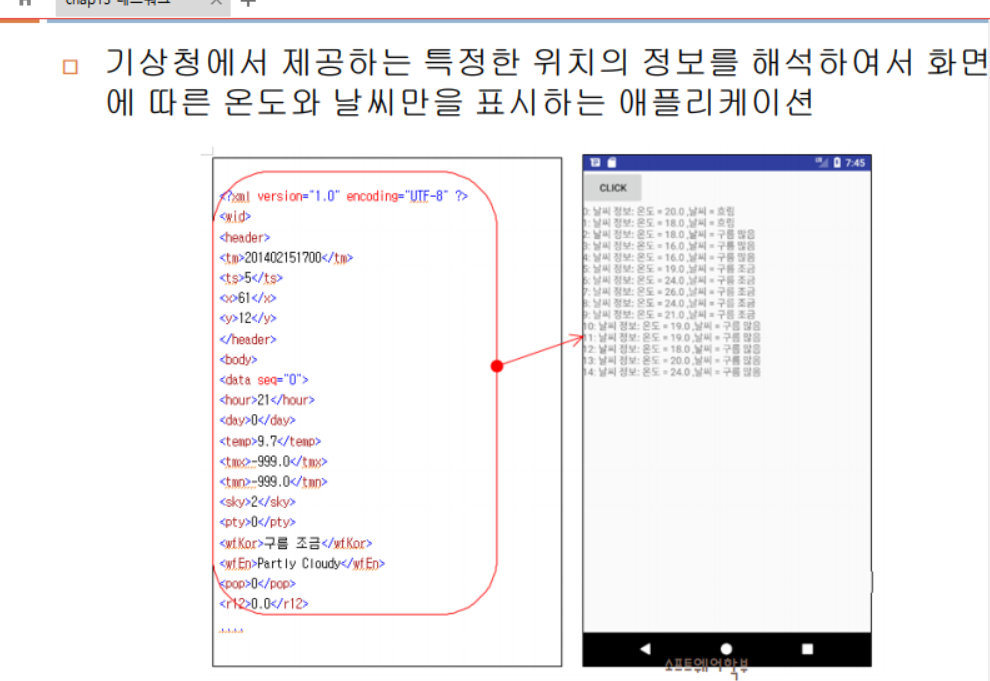
2진료예약.

**3. 둘다**

1. DUR(API 사용 가능)+일반약 데이터(약학정보원)

저장 되어있는 처방전 혹은 일반약 사진 데이터를 이용하여 DUR 정보를 알림 메세지로 알려줍니다. – 신형

DUR API : <https://www.data.go.kr/dataset/15020627/openapi.do>



그 API에서 바로바로 필요한 정보만 파밍해서 가져온다 -> 다음 알림메세지로 알려준다

1. 음성인식 서비스 (네이버 AI 기술 이용) - 지현

음성인식 버튼을 통하여 앱 기능을 이용할 수 있습니다. 다음 병원 예약이나,

    복용알림을 알려주고, 약을 검색해주고, 약 추천도 받습니다.

\* API 사용 전 준비 사항

1. 콘솔의 **AI·Application Service > AI·NAVER API > Application**에서 애플리케이션을 등록합니다.(자세한 방법은 [Application 사용 가이드](https://docs.ncloud.com/ko/naveropenapi_v3/application.html) 참고)
2. **AI·Application Service > AI·NAVER API > Application**에서 등록한 애플리케이션을 선택해 Client ID와 Client Secret값을 확인합니다.
3. **AI·Application Service > AI·NAVER API > Application**의 **변경** 화면에서 **Clova Speech Recognition**가 선택되어 있는지 확인합니다. 선택되어 있지 않으면 429 (Quota Exceed)가 발생하니 주의하시기 바랍니다.

\* Android 기준

**1. 음성 인식 버튼 구현**

**2. 음성 인식 관련 API – 약을 검색해준다.**

**2-1. CSR**

Clova Speech Recognition API(이하 CSR API)는 사용자의 음성 입력을 스트리밍 형태로 입력받은 후 음성 인식한 결과를 텍스트로 반환하는 API

**2-2. CSR Android API 사용하기**

- 1. 다음 구문을 app/build.gradle 파일에 추가합니다.

repositories {

jcenter()

}

dependencies {

compile 'com.naver.speech.clientapi:naverspeech-ncp-sdk-android:1.1.6'

}

- 2. 다음과 같이 Android Manifest 파일(AndroidManifest.xml)을 설정합니다.

● 패키지 이름 : manifest 요소의 package 속성 값이 [사전 준비사항](https://docs.ncloud.com/ko/naveropenapi_v3/speech/recognition-sdk.html#Preparation)에서 등록한 **안드로이드 앱 패키지 이름**과 같아야 합니다.

● 권한 설정 : 사용자의 음성 입력을 마이크를 통해 녹음해야 하고 녹음된 데이터를 서버로 전송해야 합니다. 따라서, android.permission.INTERNET와 android.permission.RECORD\_AUDIO에 대한 권한이 반드시 필요합니다.

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.naver.naverspeech.client"

android:versionCode="1" android:versionName="1.0" >

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.RECORD\_AUDIO" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" /><uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

- 3. (선택) proguard-rules.pro 파일에 다음을 추가합니다. 아래 코드는 앱을 보다 가볍

고 안전하게 만들어줍니다.

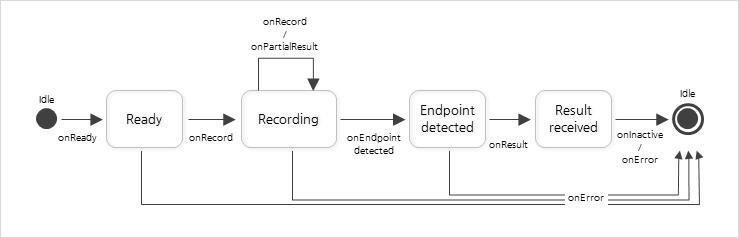
-keep class com.naver.speech.clientapi.SpeechRecognizer {

protected private \*;

}

- **Note!**

네이버 Open API는 Android SDK 버전 10 이상을 지원합니다. 따라서, *build.gradle* 파일의 minSdkVersion 값을 이에 맞게 설정해야 합니다.

클라이언트는 "준비", "녹음", "중간결과 출력", "끝점 추출", "최종결과 출력"과 같은 일련의 이벤트 흐름을 수행합니다. 애플리케이션 개발자는 SpeechRecognitioinListener 인터페이스를 상속받아 해당 이벤트가 발생할 때 처리할 동작을 구현하면 됩니다. 

- **Note!**

API에 대한 자세한 설명은 [https://github.com/NaverCloudPlatform/naverspeech-sdk-android](https://github.com/NaverCloudPlatform/naverspeech-sdk-android/)를 참고합니다.

**2-3. UX 고려사항**

- 일반적으로 사용자는 음성 인식 버튼을 누르자마자 발화를 시작하려고 할 것입니다.

하지만 음성 인식을 시작하는 recognize() 메서드를 호출하면 음성 인식을 위

한 메모리 할당, 마이크 자원 할당, 음성 인식 서버 접속 및 인증 등의 준비 과정을

수행해야 하기 때문에 사용자의 발화 일부가 누락될 수 있습니다. 따라서, 앱은 모든

준비가 완료된 후 사용자에게 발화해도 좋다는 정보를 전달해야 합니다. 이 방법은

다음과 같이 처리할 수 있습니다.

* 모든 준비가 완료되면 onReady callback 메서드가 호출됩니다.
* onReady callback 메서드가 호출되기 전까지 "준비 중입니다."와 같은 메시지를 표시하거나 준비 중임을 나타내는 UI 표시를 해야 합니다.
* onReady callback 메서드가 호출되면 "이야기해주세요."와 같은 메시지를 표시하거나 사용 가능함을 나타내는 UI를 표시해야 합니다.

- **Note!**

● SpeechRecognitionListener의 onReady, onRecord 등의 callback 메서드는 [Worker Thread](https://developer.android.com/guide/components/processes-and-threads.html)에서 호출되는 메서드이며, Handler에 등록하여 사용해야 합니다.

**2-4. 오류 처리**

- SR API를 사용할 때 다양한 원인으로 오류가 발생할 수 있으며 이때 오류 callback 함

수를 통해 오류 코드가 전달됩니다. 오류 코드를 분석하면 원인을 분석하거나 오류 처

리를 할 수 있습니다. CSR API의 오류 callback 함수가 전달하는 오류는 다음과 같습니다.

| **오류 이름** | **오류 코드** | **설명** |
| --- | --- | --- |
| ERROR\_NETWORK\_INITIALIZE | 10 | 네트워크 자원 초기화 오류 |
| ERROR\_NETWORK\_FINALIZE | 11 | 네트워크 자원 해제 오류 |
| ERROR\_NETWORK\_READ | 12 | 네트워크 데이터 수신 오류. 클라이언트 기기의 네트워크 환경이 느려 Timeout 이 발생하는 경우 주로 발생합니다. |
| ERROR\_NETWORK\_WRITE | 13 | 네트워크 데이터 전송 오류. 클라이언트 기기의 네트워크 환경이 느려 Timeout 이 발생하는 경우 주로 발생합니다. |
| ERROR\_NETWORK\_NACK | 14 | 음성 인식 서버 오류. 느린 네트워크 환경으로 인해 클라이언트가 서버로 음성 패킷을 제시간에 보내지 못하면 서버는 Timeout를 발생시킵니다. 이때 발생하는 오류입니다. |
| ERROR\_INVALID\_PACKET | 15 | 유효하지 않은 패킷 전송으로 인한 오류 |
| ERROR\_AUDIO\_INITIALIZE | 20 | 오디오 자원 초기화 오류. 오디오 사용 권한이 있는지 확인합니다. |
| ERROR\_AUDIO\_FINALIZE | 21 | 오디오 자원 해제 오류 |
| ERROR\_AUDIO\_RECORD | 22 | 음성 입력(녹음) 오류. 오디오 사용 권한이 있는지 확인합니다. |
| ERROR\_SECURITY | 30 | 인증 권한 오류 |
| ERROR\_INVALID\_RESULT | 40 | 인식 결과 오류 |
| ERROR\_TIMEOUT | 41 | 일정 시간 이상 서버로 음성을 전송하지 못하거나, 인식 결과를 받지 못함. |
| ERROR\_NO\_CLIENT\_RUNNING | 42 | 클라이언트가 음성 인식을 수행하지 않는 상황에서 특정 음성 인식 관련 이벤트가 감지됨. |
| ERROR\_UNKNOWN\_EVENT | 50 | 클라이언트 내부에 규정되어 있지 않은 이벤트가 감지됨. |
| ERROR\_VERSION | 60 | 프로토콜 버전 오류 |
| ERROR\_CLIENTINFO | 61 | 클라이언트 정보 오류 |
| ERROR\_SERVER\_POOL | 62 | 음성 인식 가용 서버 부족 |
| ERROR\_SESSION\_EXPIRED | 63 | 음성 인식 서버 세션 만료 |
| ERROR\_SPEECH\_SIZE\_EXCEEDED | 64 | 음성 패킷 사이즈 초과 |
| ERROR\_EXCEED\_TIME\_LIMIT | 65 | 인증용 타임 스탬프(time stamp) 불량 |
| ERROR\_WRONG\_SERVICE\_TYPE | 66 | 올바른 서비스 타입(service type)이 아님. |
| ERROR\_WRONG\_LANGUAGE\_TYPE | 67 | 올바른 언어 타입(language type)이 아님. |
| ERROR\_OPENAPI\_AUTH | 70 | Open API 인증 오류. Client ID와 등록된 package 이름(Android) 또는 Bundle ID 정보(iOS)가 잘못되었을 때 발생합니다. |
| ERROR\_QUOTA\_OVERFLOW | 71 | 정해진 API 호출 제한량(quota)을 모두 소진함. |

위 오류 코드 외에도 다음과 같은 오류가 발생하거나 문의가 있을 수 있습니다.

| **현상 또는 문의** | **원인 또는 해결 방법** |
| --- | --- |
| UnsatifiedLinkError 오류 발생 | CSR API는 armeabi와 armeabi-v7a로 빌드된 라이브러리를 제공합니다. 만약 개발하는 앱에서 사용하는 라이브러리 중 armeabi와 armeabi-v7a를 지원하지 않는 것이 있다면 이 오류가 발생할 수 있습니다. |
| android fatal signal 11 (sigsegv) 오류 발생 | CSR API를 사용하여 음성을 입력받기 전에 우선 자원을 준비해야 합니다. recognize() 메서드를 호출하기 전에 initialize() 메서드가 잘 호출되는지 확인해야 합니다. 또한 release() 메서드도 호출할 수 있어야 합니다. |
| 인식 결과로 ""(null)이 반환됩니다. | 사용자가 매우 작은 목소리로 발성하였거나, 주변 소리로 인해 목소리가 인식되지 않았을 경우 발생할 수 있습니다. 극히 드물게 발생하지만 인식 결과가 null일 때도 예외 처리해주는 것을 권장합니다. |
| 오디오 파일 인식 | CSR API는 오디오 파일 인식을 지원하지 않습니다. |
| 저사양 스마트 폰에서 제대로 동작하지 않습니다. | CSR API는 Android SDK 버전 10 이상과 iOS 버전 8 이상의 기기를 지원하고 있습니다. |

2-5. 구현 예제

* 예제 코드 저장소 : <https://github.com/NaverCloudPlatform/naverspeech-sdk-android>
* 설명
  + [Main Activity](https://github.com/NaverCloudPlatform/naverspeech-sdk-android/blob/ncp/samples/EPDTypeAutoSample/app/src/main/java/com/naver/naverspeech/client/MainActivity.java) 클래스 : SpeechRecognitionListener를 초기화하고, 이후 이벤트를 handleMessage에서 받아 처리합니다.
  + [SpeechRecognitionListener를 상속한 클래스](https://github.com/NaverCloudPlatform/naverspeech-sdk-android/blob/ncp/samples/EPDTypeAutoSample/app/src/main/java/com/naver/naverspeech/client/NaverRecognizer.java) : 음성인식 서버 연결, 음성전달, 인식결과 발생등의 이벤트에 따른 결과 처리 방법 정의합니다.

// 1. Main Activity 클래스

public class MainActivity extends Activity {

private static final String TAG = MainActivity.class.getSimpleName();

private static final String CLIENT\_ID = "YOUR CLIENT ID"; // "내 애플리케이션"에서 Client ID를 확인해서 이곳에 적어주세요.

private RecognitionHandler handler;

private NaverRecognizer naverRecognizer;

private TextView txtResult;

private Button btnStart;

private String mResult;

private AudioWriterPCM writer;

// Handle speech recognition Messages.

private void handleMessage(Message msg) {

switch (msg.what) {

case R.id.clientReady: // 음성인식 준비 가능

txtResult.setText("Connected");

writer = new AudioWriterPCM(Environment.getExternalStorageDirectory().getAbsolutePath() + "/NaverSpeechTest");

writer.open("Test");

break;

case R.id.audioRecording:

writer.write((short[]) msg.obj);

break;

case R.id.partialResult:

mResult = (String) (msg.obj);

txtResult.setText(mResult);

break;

case R.id.finalResult: // 최종 인식 결과

SpeechRecognitionResult speechRecognitionResult = (SpeechRecognitionResult) msg.obj;

List<String> results = speechRecognitionResult.getResults();

StringBuilder strBuf = new StringBuilder();

for(String result : results) {

strBuf.append(result);

strBuf.append("\n");

}

mResult = strBuf.toString();

txtResult.setText(mResult);

break;

case R.id.recognitionError:

if (writer != null) {

writer.close();

}

mResult = "Error code : " + msg.obj.toString();

txtResult.setText(mResult);

btnStart.setText(R.string.str\_start);

btnStart.setEnabled(true);

break;

case R.id.clientInactive:

if (writer != null) {

writer.close();

}

btnStart.setText(R.string.str\_start);

btnStart.setEnabled(true);

break;

}

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

txtResult = (TextView) findViewById(R.id.txt\_result);

btnStart = (Button) findViewById(R.id.btn\_start);

handler = new RecognitionHandler(this);

naverRecognizer = new NaverRecognizer(this, handler, CLIENT\_ID);

btnStart.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

if(!naverRecognizer.getSpeechRecognizer().isRunning()) {

mResult = "";

txtResult.setText("Connecting...");

btnStart.setText(R.string.str\_stop);

naverRecognizer.recognize();

} else {

Log.d(TAG, "stop and wait Final Result");

btnStart.setEnabled(false);

naverRecognizer.getSpeechRecognizer().stop();

}

}

});

}

@Override

protected void onStart() {

super.onStart(); // 음성인식 서버 초기화는 여기서

naverRecognizer.getSpeechRecognizer().initialize();

}

@Override

protected void onResume() {

super.onResume();

mResult = "";

txtResult.setText("");

btnStart.setText(R.string.str\_start);

btnStart.setEnabled(true);

}

@Override

protected void onStop() {

super.onStop(); // 음성인식 서버 종료

naverRecognizer.getSpeechRecognizer().release();

}

// Declare handler for handling SpeechRecognizer thread's Messages.

static class RecognitionHandler extends Handler {

private final WeakReference<MainActivity> mActivity;

RecognitionHandler(MainActivity activity) {

mActivity = new WeakReference<MainActivity>(activity);

}

@Override

public void handleMessage(Message msg) {

MainActivity activity = mActivity.get();

if (activity != null) {

activity.handleMessage(msg);

}

}

}

}

// 2. SpeechRecognitionListener를 상속한 클래스

class NaverRecognizer implements SpeechRecognitionListener {

private final static String TAG = NaverRecognizer.class.getSimpleName();

private Handler mHandler;

private SpeechRecognizer mRecognizer;

public NaverRecognizer(Context context, Handler handler, String clientId) {

this.mHandler = handler;

try {

mRecognizer = new SpeechRecognizer(context, clientId);

} catch (SpeechRecognitionException e) {

e.printStackTrace();

}

mRecognizer.setSpeechRecognitionListener(this);

}

public SpeechRecognizer getSpeechRecognizer() {

return mRecognizer;

}

public void recognize() {

try {

mRecognizer.recognize(new SpeechConfig(LanguageType.KOREAN, EndPointDetectType.AUTO));

} catch (SpeechRecognitionException e) {

e.printStackTrace();

}

}

@Override

@WorkerThread

public void onInactive() {

Message msg = Message.obtain(mHandler, R.id.clientInactive);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onReady() {

Message msg = Message.obtain(mHandler, R.id.clientReady);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onRecord(short[] speech) {

Message msg = Message.obtain(mHandler, R.id.audioRecording, speech);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onPartialResult(String result) {

Message msg = Message.obtain(mHandler, R.id.partialResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetected() {

Log.d(TAG, "Event occurred : EndPointDetected");

}

@Override

@WorkerThread

public void onResult(SpeechRecognitionResult result) {

Message msg = Message.obtain(mHandler, R.id.finalResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onError(int errorCode) {

Message msg = Message.obtain(mHandler, R.id.recognitionError, errorCode);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetectTypeSelected(EndPointDetectType epdType) {

Message msg = Message.obtain(mHandler, R.id.endPointDetectTypeSelected, epdType);

msg.sendToTarget();

}

}

case R.id.recognitionError:

if (writer != null) {

writer.close();

}

mResult = "Error code : " + msg.obj.toString();

txtResult.setText(mResult);

btnStart.setText(R.string.str\_start);

btnStart.setEnabled(true);

break;

case R.id.clientInactive:

if (writer != null) {

writer.close();

}

btnStart.setText(R.string.str\_start);

btnStart.setEnabled(true);

break;

}

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

txtResult = (TextView) findViewById(R.id.txt\_result);

btnStart = (Button) findViewById(R.id.btn\_start);

handler = new RecognitionHandler(this);

naverRecognizer = new NaverRecognizer(this, handler, CLIENT\_ID);

btnStart.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

if(!naverRecognizer.getSpeechRecognizer().isRunning()) {

mResult = "";

txtResult.setText("Connecting...");

btnStart.setText(R.string.str\_stop);

naverRecognizer.recognize();

} else {

Log.d(TAG, "stop and wait Final Result");

btnStart.setEnabled(false);

naverRecognizer.getSpeechRecognizer().stop();

}

}

});

}

@Override

protected void onStart() {

super.onStart(); // 음성인식 서버 초기화는 여기서

naverRecognizer.getSpeechRecognizer().initialize();

}

@Override

protected void onResume() {

super.onResume();

mResult = "";

txtResult.setText("");

btnStart.setText(R.string.str\_start);

btnStart.setEnabled(true);

}

@Override

protected void onStop() {

super.onStop(); // 음성인식 서버 종료

naverRecognizer.getSpeechRecognizer().release();

}

// Declare handler for handling SpeechRecognizer thread's Messages.

static class RecognitionHandler extends Handler {

private final WeakReference<MainActivity> mActivity;

RecognitionHandler(MainActivity activity) {

mActivity = new WeakReference<MainActivity>(activity);

}

@Override

public void handleMessage(Message msg) {

MainActivity activity = mActivity.get();

if (activity != null) {

activity.handleMessage(msg);

}

}

}

}

// 2. SpeechRecognitionListener를 상속한 클래스

class NaverRecognizer implements SpeechRecognitionListener {

private final static String TAG = NaverRecognizer.class.getSimpleName();

private Handler mHandler;

private SpeechRecognizer mRecognizer;

public NaverRecognizer(Context context, Handler handler, String clientId) {

this.mHandler = handler;

try {

mRecognizer = new SpeechRecognizer(context, clientId);

} catch (SpeechRecognitionException e) {

e.printStackTrace();

}

mRecognizer.setSpeechRecognitionListener(this);

}

public SpeechRecognizer getSpeechRecognizer() {

return mRecognizer;

}

public void recognize() {

try {

mRecognizer.recognize(new SpeechConfig(LanguageType.KOREAN, EndPointDetectType.AUTO));

} catch (SpeechRecognitionException e) {

e.printStackTrace();

}

}

@Override

@WorkerThread

public void onInactive() {

Message msg = Message.obtain(mHandler, R.id.clientInactive);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onReady() {

Message msg = Message.obtain(mHandler, R.id.clientReady);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onRecord(short[] speech) {

Message msg = Message.obtain(mHandler, R.id.audioRecording, speech);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onPartialResult(String result) {

Message msg = Message.obtain(mHandler, R.id.partialResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetected() {

Log.d(TAG, "Event occurred : EndPointDetected");

}

@Override

@WorkerThread

public void onResult(SpeechRecognitionResult result) {

Message msg = Message.obtain(mHandler, R.id.finalResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onError(int errorCode) {

Message msg = Message.obtain(mHandler, R.id.recognitionError, errorCode);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetectTypeSelected(EndPointDetectType epdType) {

Message msg = Message.obtain(mHandler, R.id.endPointDetectTypeSelected, epdType);

msg.sendToTarget();

}

}

btnStart.setEnabled(true);

}

@Override

protected void onStop() {

super.onStop(); // 음성인식 서버 종료

naverRecognizer.getSpeechRecognizer().release();

}

// Declare handler for handling SpeechRecognizer thread's Messages.

static class RecognitionHandler extends Handler {

private final WeakReference<MainActivity> mActivity;

RecognitionHandler(MainActivity activity) {

mActivity = new WeakReference<MainActivity>(activity);

}

@Override

public void handleMessage(Message msg) {

MainActivity activity = mActivity.get();

if (activity != null) {

activity.handleMessage(msg);

}

}

}

}

// 2. SpeechRecognitionListener를 상속한 클래스

class NaverRecognizer implements SpeechRecognitionListener {

private final static String TAG = NaverRecognizer.class.getSimpleName();

private Handler mHandler;

private SpeechRecognizer mRecognizer;

public NaverRecognizer(Context context, Handler handler, String clientId) {

this.mHandler = handler;

try {

mRecognizer = new SpeechRecognizer(context, clientId);

} catch (SpeechRecognitionException e) {

e.printStackTrace();

}

mRecognizer.setSpeechRecognitionListener(this);

}

public SpeechRecognizer getSpeechRecognizer() {

return mRecognizer;

}

public void recognize() {

try {

mRecognizer.recognize(new SpeechConfig(LanguageType.KOREAN, EndPointDetectType.AUTO));

} catch (SpeechRecognitionException e) {

e.printStackTrace();

}

}

@Override

@WorkerThread

public void onInactive() {

Message msg = Message.obtain(mHandler, R.id.clientInactive);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onReady() {

Message msg = Message.obtain(mHandler, R.id.clientReady);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onRecord(short[] speech) {

Message msg = Message.obtain(mHandler, R.id.audioRecording, speech);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onPartialResult(String result) {

Message msg = Message.obtain(mHandler, R.id.partialResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetected() {

Log.d(TAG, "Event occurred : EndPointDetected");

}

@Override

@WorkerThread

public void onResult(SpeechRecognitionResult result) {

Message msg = Message.obtain(mHandler, R.id.finalResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onError(int errorCode) {

Message msg = Message.obtain(mHandler, R.id.recognitionError, errorCode);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetectTypeSelected(EndPointDetectType epdType) {

Message msg = Message.obtain(mHandler, R.id.endPointDetectTypeSelected, epdType);

msg.sendToTarget();

}

}

@Override

@WorkerThread

public void onReady() {

Message msg = Message.obtain(mHandler, R.id.clientReady);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onRecord(short[] speech) {

Message msg = Message.obtain(mHandler, R.id.audioRecording, speech);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onPartialResult(String result) {

Message msg = Message.obtain(mHandler, R.id.partialResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetected() {

Log.d(TAG, "Event occurred : EndPointDetected");

}

@Override

@WorkerThread

public void onResult(SpeechRecognitionResult result) {

Message msg = Message.obtain(mHandler, R.id.finalResult, result);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onError(int errorCode) {

Message msg = Message.obtain(mHandler, R.id.recognitionError, errorCode);

msg.sendToTarget();

}

@Override

@WorkerThread

public void onEndPointDetectTypeSelected(EndPointDetectType epdType) {

Message msg = Message.obtain(mHandler, R.id.endPointDetectTypeSelected, epdType);

msg.sendToTarget();

}

}

1. 잠금기능

* 어플 사용 도중 잠시 다른 취소 버튼을 눌러 다른어플이나 홈화면에 들렸다가 다시 어플을 키더라도 다시 비밀번호(혹은 지문인식)을 입력해야 어플을 사용할 수 있도록 함

<”사용자의 시간을 줄여준다”는 측면에서 추가하고 싶은 기능>//서울부터 확장.

1. 약국찾기 기능(일요일 운영 약국, 심야 운영 약국 검색 기능도 추가)

* 기존 Aya4(앱스토어) 어플에 있는 병원찾기 기능 옆에 ‘약국찾기’ 카테고리 추가
* 오픈뱅킹과 비슷한 개념으로 한 어플에서 전국 약국 검색 가능(편리함, 시간단축)
* 아래 링크에서 심야 운영 약국 목록 csv파일 다운 가능
* https://www.e-gen.or.kr/egen/search\_pharmacy.do?searchType=general



* 일요일, 공휴일 운영 약국 검색



* 검색결과 다운로드 누르면 아래 그림처럼 html파일 다운받아짐
* 

>>추후에 기술 확장

<https://www.sciencedirect.com/science/article/pii/S0958166918301903>

= *현재 빅데이터 분석으로 약을 추천해주는 기술*

#### 체질 -> 어떤약이 위험한지, 추천받아야 하는지

***3. 음성 재생 관련 API* –** 다음 병원 예약이나, 복용알림을 알려주고, 약 추천도 받습니다.

**3-1. CSS**

****

아름다운 성우의 목소리로 글을 읽어주는 애플리케이션을 만들 때 유용한 서비스입니다. 주어진 텍스트를 자연스럽게 말하듯 재생해주어 음성 안내 시스템, 뉴스/책 읽기 서비스 등에 활용할 수 있습니다.

1. 응급실 찾기 기능



* 약국찾기 정보 제공하는 사이트에서 동일하게 제공
* 검색결과 다운로드 기능 있음
* ‘의료 정보 빠르고 편리하게 제공’을 컨셉으로 잡는다면 추가하는거 고려해보아도 좋을 것 같습니다

1. 병원 예약 서비스 (병원에 채팅 형태로 문의 진행도 가능-카카오 플러스 친구추가 기능과도 비슷)

* 어플 후기 중 가장 호평이었던 서비스(막상 제가 직접 사용해보려고 하니 예약하기 기능이 거의 지원안되었음)
* 실제로 저도 자주 가는 피부과 예약할때 가장 만족했던 서비스 중에 하나였는데, 그 병원 홈페이지에서 제공하는 기능이었습니다.
* 어플에서 모든 병원을 이런 방법으로 예약할 수 있다면 고객들에게 편리함 제공과 더불어 상당한 시간단축 효과를 얻을 수 있을 것
* 병원과 협업해서 진행해야 하거나 카카오플러스친구 등록된 병원들은 같은 방법으로 연결해서 진행 가능

