종합설계 프로젝트 수행보고서

프로젝트명	시각장애인 학생들을 위한 스마트번역 애플리케이션
팀번호	S5-1
문서제목	수행계획서( ) 2차발표 중간보고서( ) 3차발표 중간보고서( ) 4차발표 중간보고서( ) 최종결과보고서(O)

2020.11.18

팀장 : 김민승

팀원 : 이두원

지도교수 : 정의훈

문서 수정 내역

작성일	대표작성자	버전(Revision)	수정내용	기타
2020.01.22	김민승	1.0	수행계획서	최초작성
2020.01.22	이두원	1.1	수행계획서	
2020.04.02	이두원	1.2	2차발표 중간보고서	
2020.04.20	이두원	1.3	3차발표 중간보고서	
2020.05.03	김민승	1.4	3차발표 중간보고서	
2020.06.25	이두원	1.5	4차발표 중간보고서	
2020.11.18	김민승	1.6	최종결과보고서	최종작성

문서 구성

진행단계	프로젝트 계획서 발표	중간발표1 (2월)	중간발표2 (4월)	학기말발표 (6월)	최종발표 (10월)
기본양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식
포함되는 내용	Ⅰ.서론 (1~6) Ⅱ.본론 (1~3) Ⅳ.참고자료	Ⅰ.서론 (1~6) Ⅱ.본론 (1~4) Ⅳ.참고자료	Ⅰ.서론 (1~6) Ⅱ.본론 (1~5) Ⅳ.참고자료	Ⅰ.서론 (1~6) Ⅱ.본론 (1~7) Ⅳ.참고자료	Ⅰ Ⅱ Ⅲ Ⅳ

이 문서는 한국산업기술대학교 컴퓨터공학부의
“종합설계” 교과목에서 프로젝트
“시각장애인 학생들을 위한 스마트번역 애플리케이션”을
수행하는 < S5-1, 김민승, 이두원 > 들이 작성한 것으로
사용하기 위해서는 팀원들의 허락이 필요합니다.

목 차

Ⅰ. 서론

Ⅱ. 본론

Ⅲ. 결론

Ⅳ. 참고자료

Ⅰ. 서론

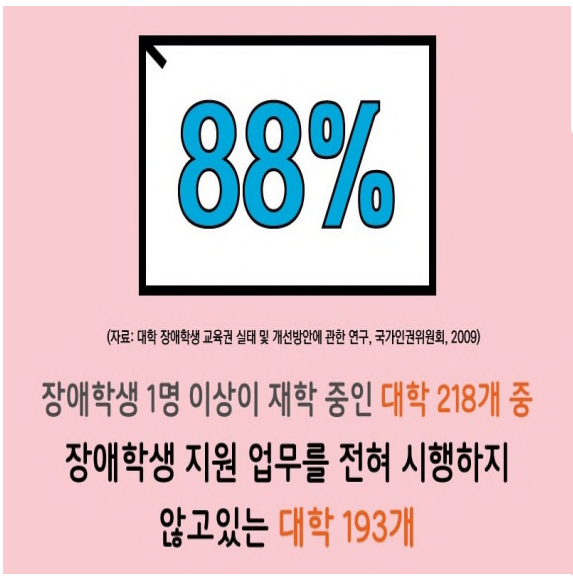
1. 작품선정 배경 및 필요성

<그림 1>
대학 교육목표 중 하나인 글로벌 인재 양성

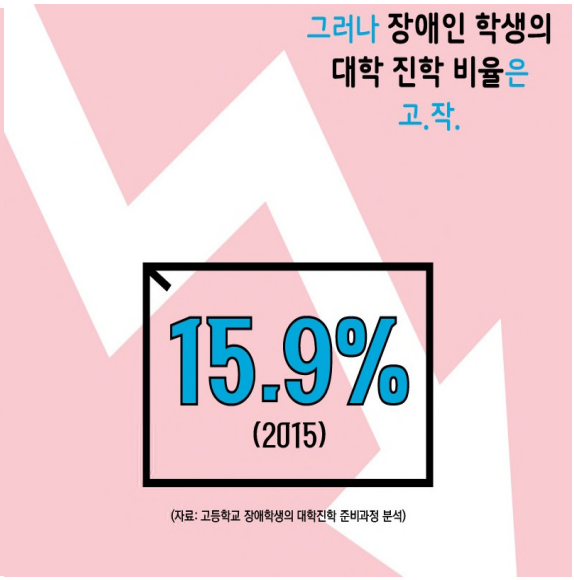
<그림 2>
주요 대학의 영어 강의 비율

<그림 1>에서 확인할 수 있듯이 최근 세계화의 영향으로 각 대학에서는 글로벌 인재 양성을 위한 영어 강의를 핵심 교육으로 선정하여 일부 교과목의 강의를 영어로만 진행하며, <그림 2>의 통계를 보면 카이스트는 교과목의 91%를 영어로 강의를 진행하고 있다.

학생들은 모국어가 아닌 영어로 강의를 진행하기에 해석 및 이해에 어려움을 겪고 있으며, 특히 다른 학생들보다 몸이 불편한 시각장애인 학생들은 더욱 어려움을 겪을 수밖에 없다.



<그림 3>
대학에 재학 중인 많은 시각장애인 학생들이
학습지원을 받지 못한 채 수업을 듣고 있음



<그림 4>
시각장애인 학생들의 대학진학 비율은
15.9%로 점차 하락세를 보이고 있음

상대적으로 어려움을 겪을 수밖에 없는 환경 속에서 <그림 3>의 자료를 보면 전국 대학교의 88%는 시각장애인 학생들에게 어떠한 학습지원을 하지 않고 있으며, 이에 따라 시간이 지날수록 시각장애인 학생들의 대학진학 비율이 <그림 4>에서 확인하듯이 15.9%까지 하락하였다.

대한민국 헌법 31조에 따르면 “능력에 따라 균등하게 교육받을 권리와 의무, 의무교육과 무상교육의 원칙”이 명시되어 있기는 하지만 명시된 교육권조차 시각장애인 학생들은 보장받지 못하는 현실이다.

이에 따라 시각장애인 학생들이 다른 사람의 도움 없이 혼자서 간편하게 사용할 수 있고 영어 해석 및 이해에 도움을 줄 수 있음과 더불어 점자해독 능력습득을 유도하는 스마트번역 애플리케이션을 개발하고자 생각하였다.

2. 기존 연구 | 기술동향 분석

<그림 5>
구글 번역기

<그림 6>
Taptilo 점자 교육 보조기

<그림 7>
봄 점자 교육 애플리케이션

사람들은 영어에 대한 해석 및 이해에 어려움을 겪을 시에 가장 먼저 번역기를 찾는다. 기존에 출시된 번역기 중 <그림 5>의 구글 번역기가 가장 많이 사용되고 있으며, 또한 출시 초기에는 소수 언어만을 제공하였으나 현재는 103개의 언어를 번역할 수 있다.

몇 년 전까지만 하더라도 구글 번역기의 정확도는 30~40% 정도에 그쳤으나 현재 구글 번역기의 정확도가 90% 이상일 정도로 발전하였다.

이 때문에 기존에 출시되어있는 번역기능을 가지고 있는 애플리케이션들은 구글 번역기의 정확한 Google Translation API가 사용되고 있으며, 시각장애인 학생들을 위한 스마트번역 애플리케이션에도 정확하고 가장 많이 사용되고 있는 구글 번역기의 Google Translation API를 사용할 예정이다.

<그림 6>은 전국 시각장애인을 위한 특수학교에서 70% 이상 사용되고 있는 Taptilo라는 점자 교육 보조기이다.

시각장애인들은 시각장애인들의 또 다른 언어인 점자의 필요성에 대해 누구보다 잘 알고 있지만 배우기 어렵다는 생각 때문에 결과적으로 세계적 점자 보급률은 10%가 채 되지 않는다. 이러한 문제점을 해결하기 위해 Taptilo라는 점자 교육 보조기가 출시되었고 스마트폰과 점자 디바이스간에 블루투스를 이용해 연결하여 사용한다.

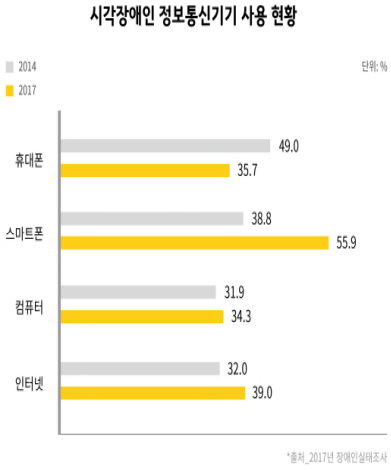
스마트폰을 이용한 음성안내를 통해 시각장애인들도 스스로 사용할 수 있고 스마트폰에 특정 단어를 입력하면 연결된 점자 디바이스를 통해 점자를 익힐 수 있는 시스템이다.

하지만 사용하기 위해서는 꼭 점자 디바이스가 필요하다는 점에서 불편함이 있으며, 이러한 불편함을 참고하여 점자 디바이스가 필요하지 않은 점자 교육 시스템을 개발하고자 한다.

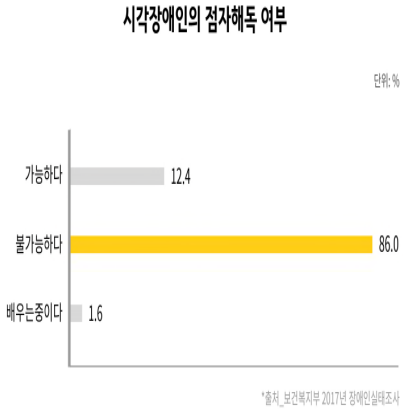
<그림 7>도 역시 시각장애인을 위한 점자 학습 애플리케이션이며, Taptilo와 다르게 점자 디바이스가 필요하지 않다는 점에서 우리가 개발하고자 하는 시각장애인을 위한 스마트번역 애플리케이션과 가장 유사한 애플리케이션이다.

스마트폰을 이용한 음성안내를 통해 시각장애인들이 혼자 스스로 사용할 수 있으며, 특정 단어를 입력하면 스마트폰 화면에 6개의 점자가 표시되어 손가락을 이용해 인식할 수 있다.
손가락이 6개의 점자 중 돌출된 부분에 터치가 되면 진동을 통해 시각장애인들이 인지할 수 있도록 도와주며, 인지가 어려울 시 돌출된 부분의 위치를 음성으로 안내해 큰 어려움 없이 점자를 습득할 수 있도록 도와준다.

3. 개발 목표



< 그림 8 >
시각장애인의 정보통신기기 사용 현황
50% 이상 시각장애인이 스마트폰 사용



< 그림 9 >
대부분의 시각장애인들은
점자해독 능력이 없음

개발하고자 하는 시각장애인 학생들을 위한 스마트번역 애플리케이션은 주 사용대상이 시각장애인 학생이므로 <그림 8>에서 확인할 수 있듯이 시각장애인들이 가장 많이 사용하는 정보통신기기인 스마트폰을 활용하여 개발하고자 한다.

시각장애인 학생들이 다른 사람의 도움 없이 스스로 영어 강의를 학습할 수 있도록 영어 강의의 내용을 실시간으로 한국어로 번역하여 음성안내를 지원함으로 해석 및 이해에 도움을 준다. 이와 함께 <그림 9>에서 보듯이 시각장애인의 86%는 점자해독 능력이 없으므로 한국어로 실시간 번역된 내용을 다시 점자로 변환하여 점자해독 능력이 없는 시각장애인들에게 점자해독 능력을 습득할 수 있도록 유도한다.

4. 팀 역할 분담

S5-1	김민승	이두원
자료수집	개발에 필요한 API 정보수집 시각장애인 점자해독 정보수집 아이디어 추출	시각장애인 관련 통계자료수집 시각장애인 애플리케이션 분석 점자 자료수집
설계	안드로이드 애플리케이션 설계	안드로이드 애플리케이션 설계
구현	STT & TTS, 번역, 텍스트 변환 등 핵심 기능구현	STT & TTS, 번역, 텍스트 변환 등 핵심 기능구현

5. 개발 일정

S5-1	11월	12월	1월	2~5월	5~8월	8~12월
기초 자료조사						
요구 사항 분석						
개발 환경 구축						
Application 구현						
Server Database 구현						
Demo 및 유지보수						
시연 및 논문작성						

6. 개발 환경

Java
애플리케이션 개발을 위한 언어

Android Studio
안드로이드 애플리케이션 제작을 위한
공식 통합 개발 환경

Google Developers
구글 API를 사용하기 위한 구글 사이트

NAVER Developers
네이버 API를 사용하기 위한 네이버 사이트

Firebase
애플리케이션 서버
점자 데이터베이스 관리를 위한 DBMS

Ⅱ. 본론

1. 개발 내용

영어 수업(강의 및 교재)의 내용들 중 영어 강의는 스마트폰의 마이크를 활용하고 영어 교재는 스마트폰의 카메라를 활용하여 음성 및 사진 데이터로 변환한다.

변환된 음성 데이터는 NAVER CLOVA API를 활용하고 변환된 사진 데이터는 Google Version API를 활용하여 길이 및 크기에 상관없이 음성 및 사진 데이터에 해당하는 영어를 텍스트로 추출이 가능하다.

<그림 10>
시각장애인들이 사용하는 점자 일람표

추출된 영어 텍스트 결과물을 NAVER PAPAGO API를 활용하여 한국어 텍스트로 번역하는 과정을 이어서 거치게 된다.

최종적으로 한국어로 번역된 텍스트 결과물을 Google Cloud TTS API를 활용하여 한국어 음성으로 변환하여 음성안내를 통해 시각장애인 학생들이 이해 및 해석에 어려움이 있었던 부분을 애플리케이션을 통해 배울 수 있도록 도와준다.

또한 최종적으로 한국어로 번역된 텍스트 결과물을 자체 개발한 점자 변환 알고리즘을 통해 한국어 텍스트에 맞는 점자로 다시 변환하여 시각장애인 학생들이 점자해독 능력습득에 도움이 될 수 있도록 음성안내와 함께 스마트폰 화면에 출력한다.

스마트폰 화면에 점자를 구현하기 위해 총 6개의 점을 스마트폰 화면에 돌출시켰으며, 시각장애인 학생들이 스마트폰 화면에 손가락으로 문지르면서 돌출된 6개의 점자를 터치하게 되는 순간 진동을 통해 인지시킨다. 또한 잘못 인지할 수 있음에 대비하여 돌출된 점자를 터치하게 되는 순간에 음성안내를 통해 정확하게 인지할 수 있도록 도와준다.

2. 문제 및 해결방안

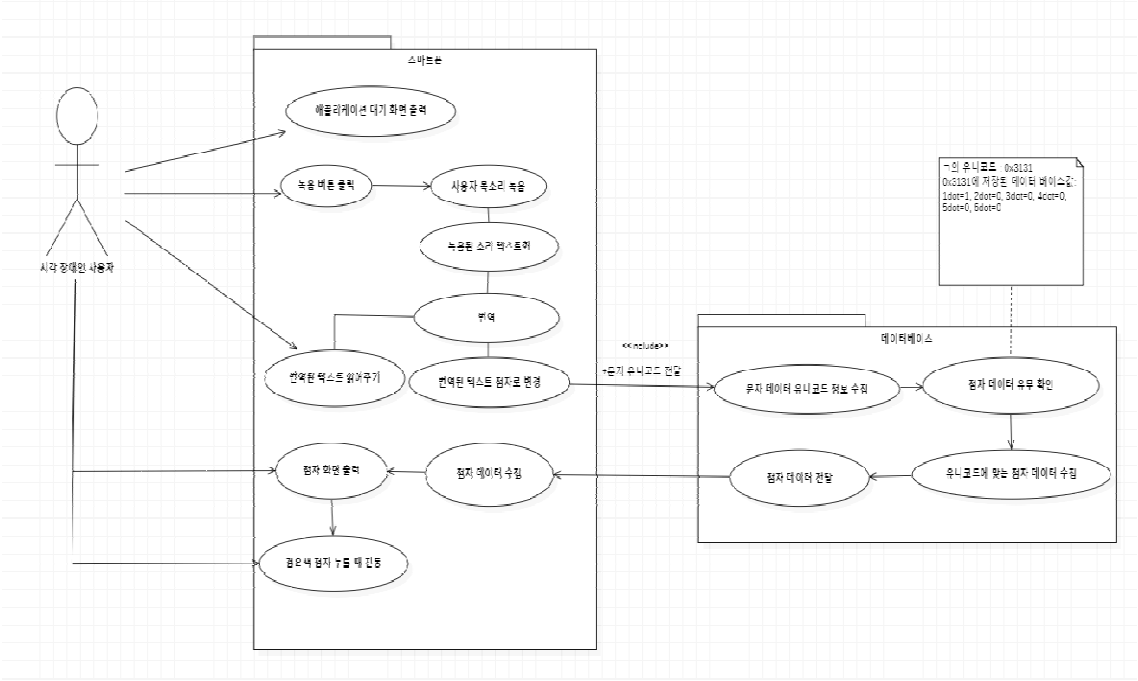
음성을 실시간으로 텍스트화하여 번역 후 점자로 변환하는 것이 지연없이 구현이 가능한가?

- 음성을 실시간으로 점자로 변환하는 것은 가능하나 그 길이가 길어질수록 지연시간이나 정확도가 점차 떨어지는 문제가 발생하였다.
- 길이가 어느 정도 있는 음성의 경우에는 시각장애인 학생들이 애플리케이션에서 시작버튼을 눌러 정지버튼을 누르기 전까지 음성을 모두 녹음하여 녹음한 음성을 번역한 후 점자로 변환하는 시나리오로 수정하여 지연시간과 정확성이 발생하는 문제를 개선하였다.

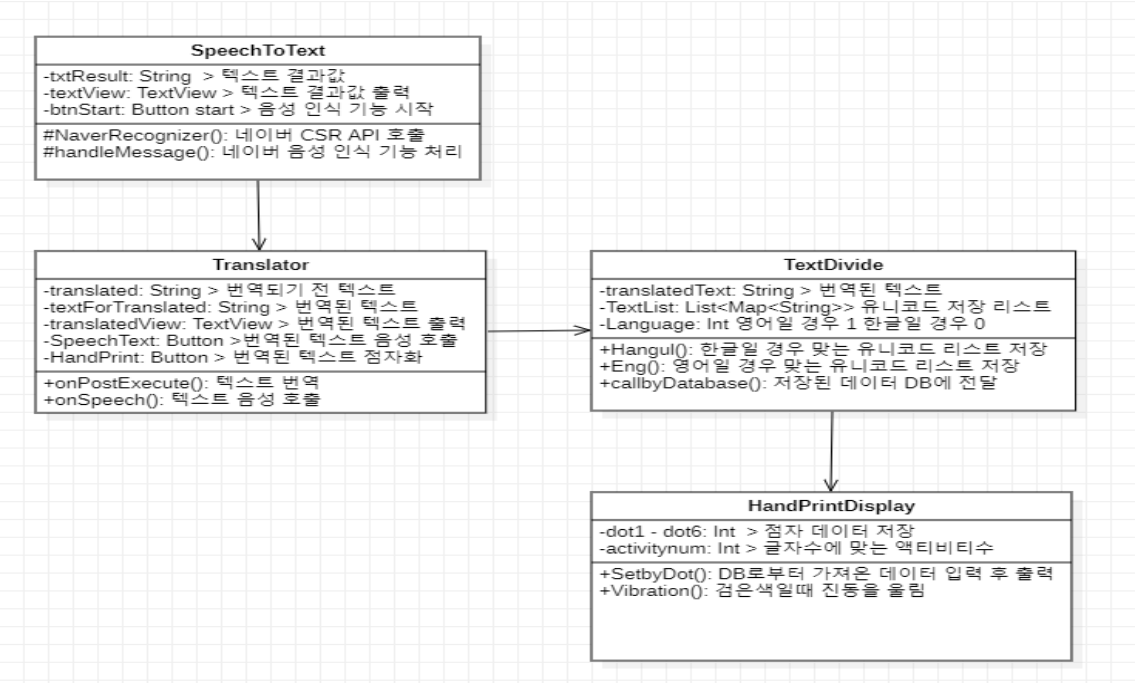
3. 시험 시나리오

(1) 영어 강의를 녹음한 음성이나 촬영된 사진을 영어 텍스트로 변환한다.
(2) 변환된 영어 텍스트를 한국어 텍스트로 번역한다.
(3-1) 번역된 결과물을 Google Cloud TTS API를 통해 음성안내를 시작한다.
(3-2) 번역된 결과물을 Server를 통해 Database로 전달한다.
(3-2-1) 전달된 결과물을 초성, 중성, 종성으로 분리한다.
Ex) 사과 -> ㅅ ㅏ ㄱ ㅗ ㅏ
Ex) 나무 -> ㄴ ㅏ ㅁ ㅜ
(3-2-2) 분리한 문자에 해당하는 점자 데이터를 Server를 통해 Database에서 가져온다.
(3-2-3) 가져온 점자 데이터를 한 개씩 스마트폰 화면에 6개의 돌출된 점으로 출력한다.
(3-2-4) 출력된 6개의 점 중 검은색 점자에 손가락이 터치되면 진동으로 사용자에게 알리고 흰 점자에는 손가락이 터치되어도 진동을 울리지 않는다.
(3-2-5) 해독을 모두 마치면 화면을 손가락을 통해 좌-우로 넘겨 다음 문자로 넘어가거나 이전 문자를 다시 확인할 수 있다,
(3-2-6) 위 과정을 점자로 번역된 결과물을 모두 읽을 때까지 반복한다.

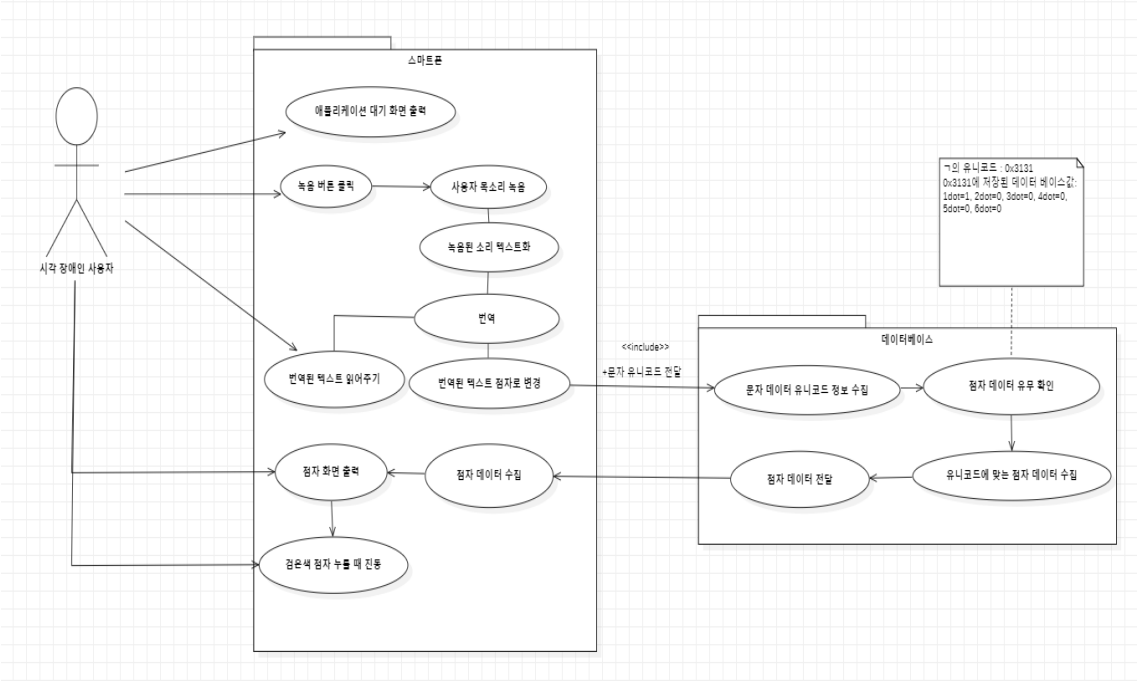
4. 상세 설계



<그림 11>
애플리케이션 구상도

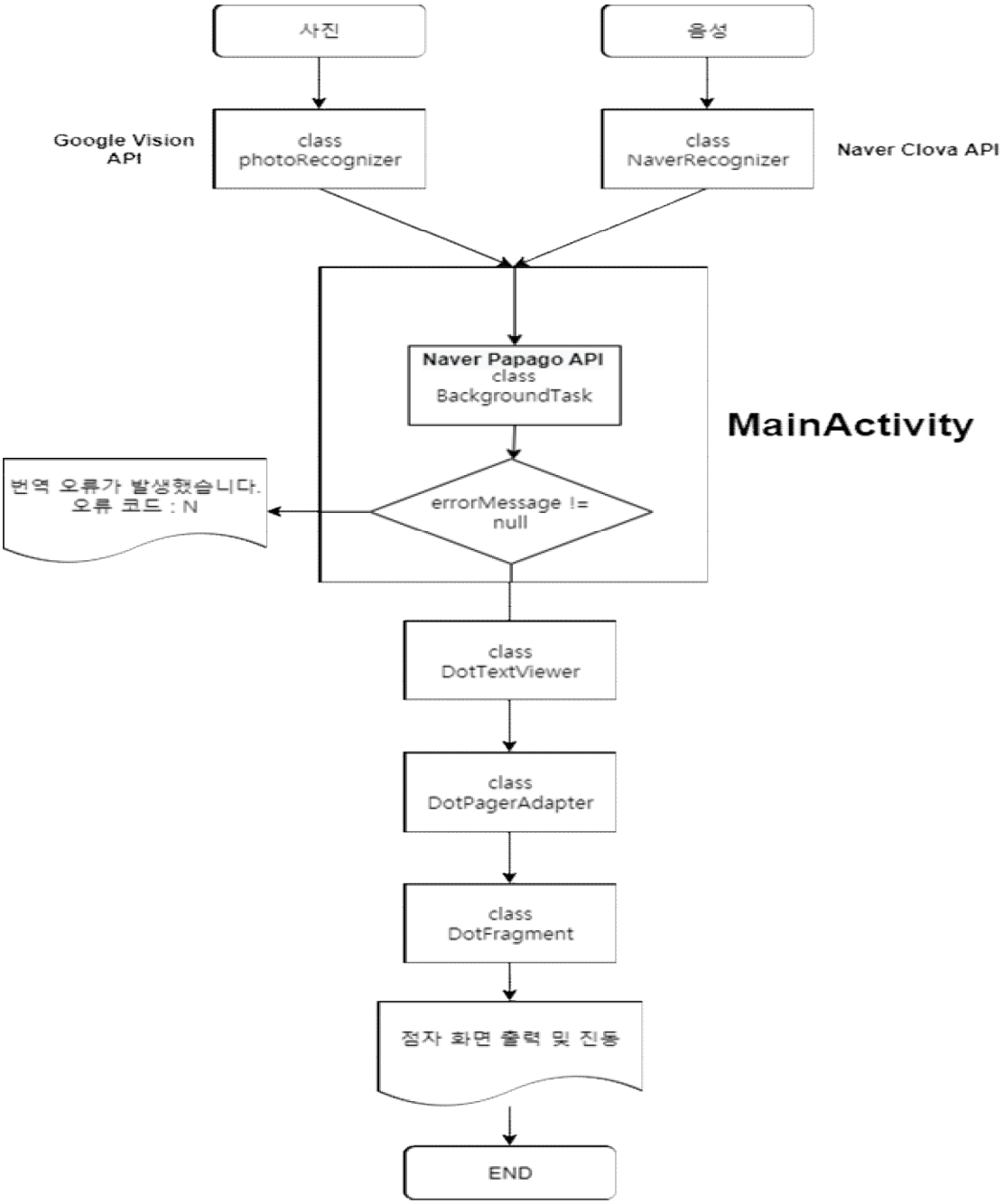


<그림 12>
애플리케이션 클래스 다이어그램



<그림 13>
애플리케이션 컴포넌트 다이어그램

<그림 14>
애플리케이션 UI 구상도



<그림 15>
애플리케이션 알고리즘

외국어 수업의 강의 음성 및 교재를 스마트폰의 마이크와 카메라를 통해 인지하며, 인지된 내용을 한국어로 번역 후 음성안내를 통하여 애플리케이션 사용자에게 이해 및 해석에 도움을 준다.

애플리케이션의 주 사용자인 시각장애인 학생들의 대부분은 점자해독 능력이 없으므로 한국어로 번역된 내용을 점자로 재변환하여 음성안내와 함께 점자해독 능력습득을 유도한다. 스마트폰 화면에 6개의 돌출된 점자를 표현하며, 손가락으로 화면을 문지르다 돌출된 점자에 터치가 되면 진동을 통해 사용자가 인지할 수 있도록 구성하였다.

상세 모듈 설계 - class NaverRecognizer

애트리뷰트	변수타입	역할
translatedView	TextView	번역된 결과값 출력
textForTranslated	String	번역하기 위한 텍스트 결과값
translated	String	번역된 결과값
address	String	API를 불러올 주소

메소드	역할

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);
}
	CSR 음성 인식 요청

public SpeechRecognizer getSpeechRecognizer() {
return mRecognizer;
}
	CSR 음성 인식 실행

public void recognize(boolean startKo) {
try {
mRecognizer.recognize(new SpeechConfig(startKo ? SpeechConfig.LanguageType.*KOREAN* : SpeechConfig.LanguageType.*ENGLISH*, SpeechConfig.EndPointDetectType.*AUTO*
));
} catch (SpeechRecognitionException e) {
e.printStackTrace();
}
}
	언어 선택을 한국어로 했으면 한국어만
	영어로 했으면 영어만 텍스트로 출력한다.

public void onInactive() {
Message msg = Message.*obtain*(mHandler, R.id.*clientInactive*);
msg.sendToTarget();
}
	CSR 음성 인식 실행

public void onReady() {
Message msg = Message.*obtain*(mHandler, R.id.*clientReady*);
msg.sendToTarget();
}
	CSR 음성 인식 준비

public void onRecord(short[] speech) {
Message msg = Message.*obtain*(mHandler, R.id.*audioRecording*, speech);
msg.sendToTarget();
}
	CSR 음성 인식 녹음

메소드	역할

public void onPartialResult(String result) {
Message msg = Message.*obtain*(mHandler, R.id.*partialResult*, result);
msg.sendToTarget();
}
	CSR 음성 인식 도중 텍스트 출력

public void onEndPointDetected() {
Log.*d*(*TAG*, "Event occurred : EndPointDetected");
}
	CSR 음성 인식 끝점 추출

public void onResult(SpeechRecognitionResult result) {
Message msg = Message.*obtain*(mHandler, R.id.*finalResult*, result);
msg.sendToTarget();
}
	CSR 음성 인식 결과를 모두 보여줌

public void onError(int errorCode) {
Message msg = Message.*obtain*(mHandler, R.id.*recognitionError*, errorCode);
msg.sendToTarget();
}
	CSR 음성 인식 도중 에러값 출력

public void onEndPointDetectTypeSelected(SpeechConfig.EndPointDetectType epdType) {
Message msg = Message.*obtain*(mHandler, R.id.*endPointDetectTypeSelected*, epdType);
msg.sendToTarget();
}
	CSR 음성 인식 끝점 추출을 통한 최종 결과 출력 요청

상세 모듈 설계 - class MainActivity BackgroundTask

애트리뷰트	변수타입	역할
txtResult	TextView	음성을 텍스트로 변환한 값 저장
btnStart	Button	음성 인식 시작 버튼
mResult	String	텍스트로 변환된 값 저장
btnTranslate	Button	번역 시작 버튼
translatedView	TextView	번역된 결과값 출력
textForTranslated	String	번역하기 위한 텍스트 저장
translated	String	번역된 결과값 저장
address	String	음성 인식을 불러올 API 주소
startKo	boolean	시작점이 한국어 여부에 따라 번역 시작과 끝 부분을 다르게 함

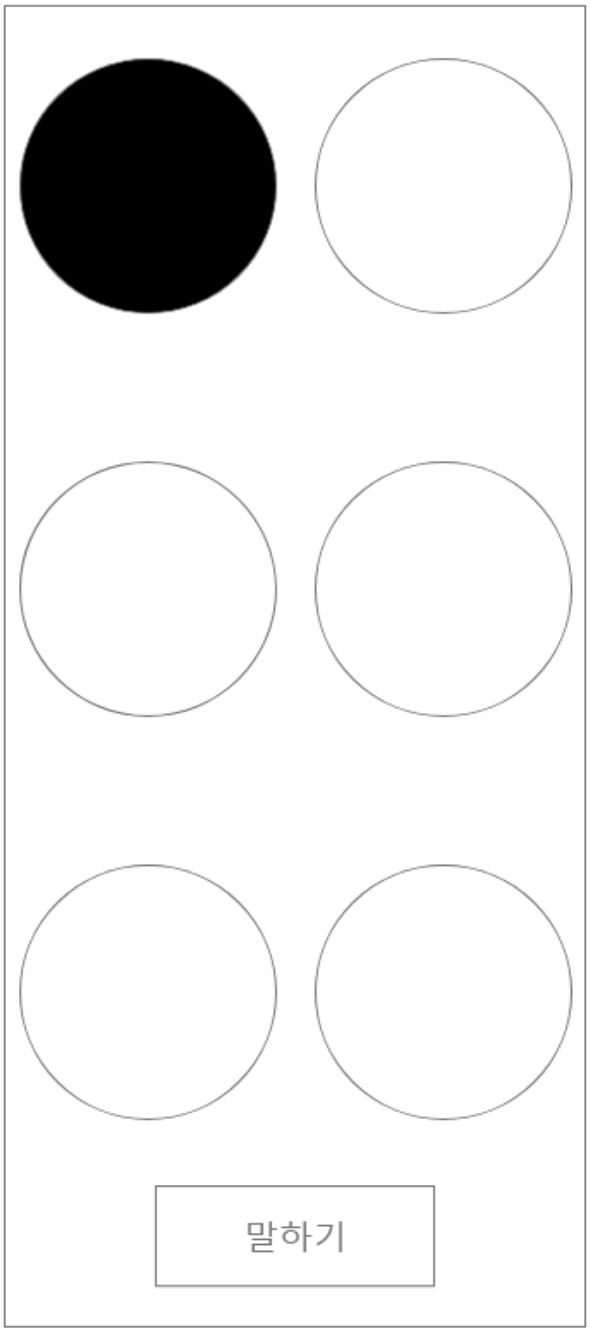
메소드	역할
private String forTranslated;	번역이 완료된 TextView에 띄우기 위한 메소드 번역 결과를 점자로 만드는 데 사용

protected Integer doInBackground(Integer... arg0) {
translated = request(address, forTranslated);
return null;
}
	API 주소에 번역 요청하는 함수

protected void onPostExecute(Integer a) {}	번역 결과를 출력하거나 오류가 발생했을 시 오류 코드와 함께 출력

메소드	역할
private void handleMessage(Message msg)	**case** R.id.*clientReady*: 음성인식을 시작할 준비가 완료된 경우 **case** R.id.*audioRecording* : 현재 음성 인식이 진행되고 있는 경우 **case** R.id.*partialResult* : 처리가 되고 있는 도중에 결과를 받은 경우 **case** R.id.*finalResult* : 최종 인식이 완료되면 유사 결과를 모두 보여줌
protected void onCreate(Bundle savedInstanceState)	버튼을 누르면 실행될 작업 선언 사용자의 OS 버전을 체크하고 권한이 허용되어 있는지 체크 권한이 있으면 음성 인식 기능 처리 권한이 없으면 취소
private void startRecognize()	CSR 음성 인식 시작 언어를 선택하는 화면 출력 언어 선택 출력 후 영어 -> 한국어로 할지 한국어 -> 영어로 할지 선택
protected void onStart()	음성 인식 서버를 초기화
protected void onResume()	음성 인식 서버에 접근 후 음성 인식 시작
protected void onStop()	음성 인식 서버를 종료
static class RecognitionHandler **extends** Handler	SpeechRecognizer 쓰레드의 메시지를 처리하는 핸들러 정의
private String request(String urlStr, String textForTranslated){}	애플리케이션 클라이언트와 시크릿 값을 통해 API 사용 권한 확인 포스트 방식으로 파라미터 전달 시작점이 한국어 여부에 따라 번역 시작과 끝 부분을 다르게 함

상세 모듈 설계 - class DotTextViewer



애트리뷰트	변수타입	역할
map	HashMap	점자 정보를 담는 Map
mapJong	HashMap	점자 종성 정보를 담는 Map
fortis	int[]	된소리를 표기하기 위한 점자
toSplit	HashSet	따로 나눠서 표시해야 하는 모음 점자
decomposed	List<String>	자모 분리
fragments	List<DotFragment>	분해된 문자의 DotFragment 클래스 점자 데이터 추가
viewPager	ViewPager	점자를 보일 페이지
pagerAdapter	DotPagerAdapter	ViewPager를 가져와서 adapter를 직접 만든 어댑터로 설정

메소드	역할
map.put('ㄱ', new int[] { 0, 1, 0, 0, 0, 0 });}	초성, 중성 점자 데이터
mapJong.put('ㄱ', new int[] { 1, 0, 0, 0, 0, 0 });	종성 점자 데이터가 담겨져 있는 맵 종성은 위 맵과는 달리 따로 다른 맵에 넣어서 초성과 겹치는 일을 방지
public Object instantiateItem(ViewGroup container, int position)	ViewGroup(ViewPager)에게 해당하는 Fragment를 등록해 준다.
public void destroyItem(ViewGroup container, int position, Object object)	뷰페이저에서 삭제
public int getCount()	뷰페이저의 전체 페이지 수는 decomposed된 문자열 길이로 설정
private DotFragment createDotFragment(String ch, int[] arr)	DotFragment fragment를 view에 추가하는 과정 뷰페이저에 추가 지정한 인덱시에 따라서 점자를 채워감 등록되지 않은 문자가 나오면 ViewPager에 로그를 띄우고 Null 변환

상세 모듈 설계-class DotTextViewer 한글 분리 과정

1) 초성인 경우 if(k == 0)

(가) 만약 두 글자가 합쳐진 경우 - 한번 더 분리한다

(1) 된소리인 경우 ex) ㄲ - 된소리 전용 점자 추가

(2) 된소리가 아닌 경우 ex) ㄳ - 나눠서 된소리 전용 점자에 각각 추가

(나) 만약 두 글자가 합쳐지지 않은 경우

(1) 대문자 알파벳인 경우

```
if('A' <= ch && ch <= 'Z')
        ch += 32;
```

소문자로 변경 아스키 코드상 a = 97, A = 65

따라서 32를 더하면 소문자로 바뀜

(2) 그 외 경우 - 그대로 추가

2) 중성인 경우 if(k == 1)

(가) 분해해야 하는 모음인 경우 - 분리하여 각각 추가

(나) 분해하지 않아도 되는 모음인 경우 - 그대로 추가

3) 종성인 경우 if(k == 2)

=> 종성이라 종성에 맞는 점자를 가져옴

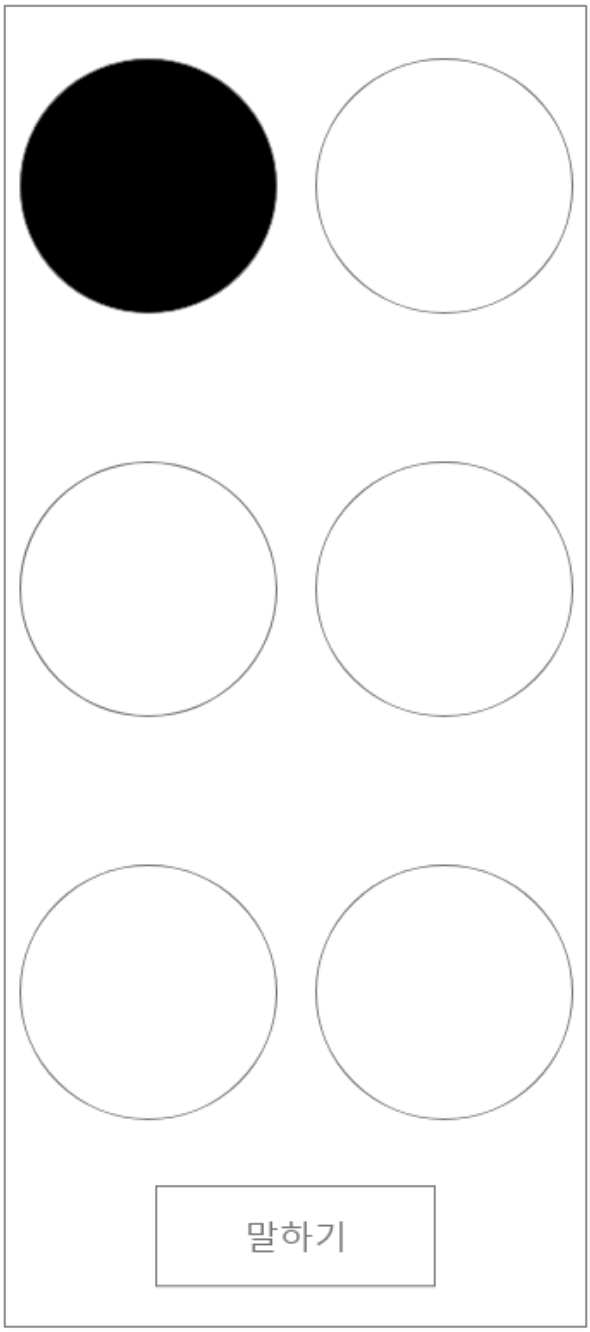
(가) 만약 두 글자가 합쳐진 경우 - 한번 더 분리한다

(1) 된소리인 경우 ex) ㄲ - 된소리 전용 점자 추가

(2) 된소리가 아닌 경우 ex) ㄳ - 나눠서 된소리 전용 점자에 각각 추가

(나) 만약 두 글자가 합쳐지지 않은 경우 - 그대로 추가

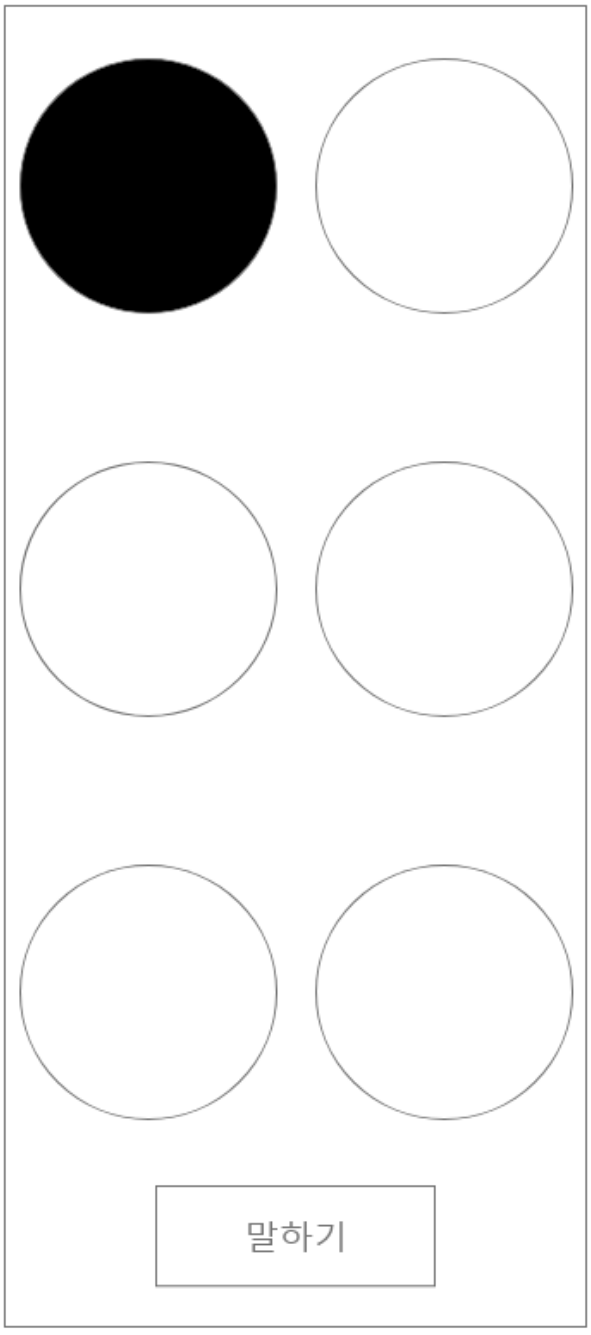
상세 모듈 설계 - class DotFragment



애트리뷰트	변수타입	역할
view	View	원형 점자
arr	int[]	n번째 값과 각 점자에 대한 인덱스를 매칭 R.id.*dot_circle_1*, R.id.*dot_circle_2*, R.id.*dot_circle_3*, R.id.*dot_circle_4*, R.id.*dot_circle_5*, R.id.*dot_circle_6*
dotText	TextView	점자에 해당하는 문자
filled	boolean[]	해당 점이 채워졌는가 true면 채워짐

메소드	역할
public void setFill(int index, boolean fill)	해당 인덱스에 대해 채워진 점자 빈 점자 여부를 설정함
public void setText(String text)	점자에 해당하는 문자
private void init(Context context)	Fragment를 위한 기본적인 레이아웃 inflate, 초기화, 터치 시 진동이 울리도록 설정

상세 모듈 설계 - class WordClass



애트리뷰트	변수타입	역할
초성	List<Character>	초성 리스트
중성	List<Character>	중성 리스트
종성	List<Character>	종성 리스트
jongDecompose	HashMap<Character, String>	자음 종성 분리하기
jongCompose	HashMap<String, Character>	자음 종성 합치기
vowelDecompose	HashMap<Character, String>	모음 분리
vowelCompose	HashMap<String, Character>	모음 합치기

된소리를 분리된 두 글자로 바꿔주는 경우의 수를 모두 등록
종성 때 된소리 경우를 따로 분리해야함
ㄲ과 같이 자음이 같을 때와 ㅥ과 같이 자음이 다를 때를 분리해서 작업해야함

지금까지 고려된 경우의 수

1. 자음(ㄱ)
2. 모음(ㅏ)
3. 자음 + 모음(가)
4. 자음 + 모음 + 자음(각각각각)
5. 자음 + 모음 + 자음 + 모음(가나)
6. 자음 + 모음 + 자음 + 자음 + 모음(각나)
7. 자음 + 모음 + 자음 + 자음 + 모음 + 자음(각난)

패턴

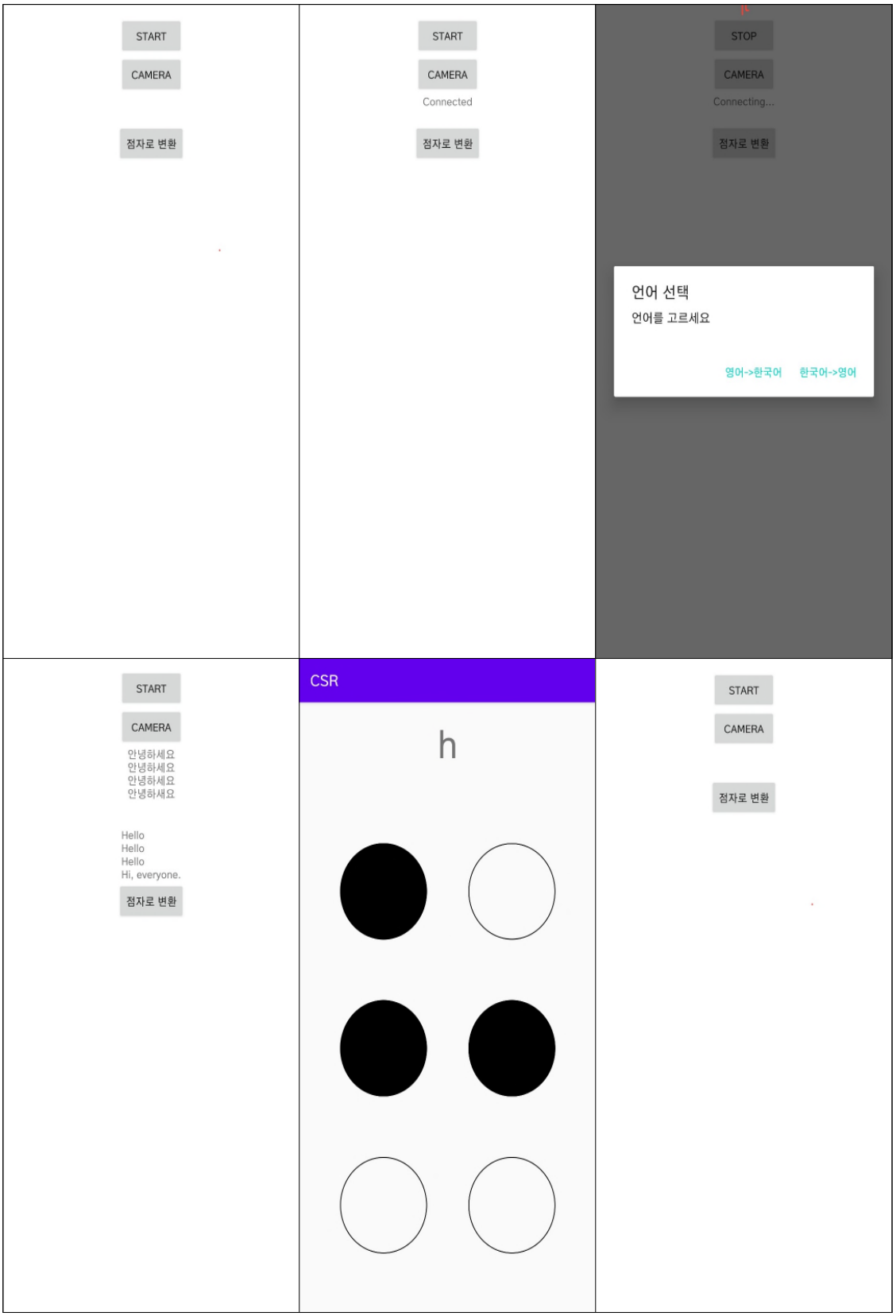
1. ㄱㅏ'ㄱ'ㄴㅏㄴ (앞 모음 뒤 자음)
2. ㄱㅏ'ㄱ'ㄱㄴㅏㄴ (앞 모음 뒤 자음.받침)
3. ㄱㅏ'ㄴ'ㅏ (앞 모음 뒤 모음)
4. ㄱㅏ'ㄱ'ㄱㄱㅏ (앞 모음 뒤 자음.받침)
5. ㄱㅏㄱ'ㄱ'ㄱㅏ (앞 자음 뒤 자음)
6. ㄱㅏㄱ'ㄱ'ㅏ (앞 자음 뒤 모음)

1. ㄱ'ㅏ'ㄱㄴㅏㄴ (앞 모음 뒤 자음)
2. ㄱ'ㅏ'ㄱㄱㄴㅏㄴ (앞 모음 뒤 자음.받침)
3. ㄱ'ㅏ'ㄴㅏ (앞 모음 뒤 모음)

메소드	역할
private static void addJong	종성 추가 메소드
private static void addVowel	모음 추가 메소드
public static String getJongDecompose(**char** ch)	자음 종성 분리 메소드
public static char getJongCompose(String str)	자음 종성 분리 후 합칠 때 메소드 ㄲ이나 ㄺ의 구분을 위해서 필요한 메소드
public static String getVowelDecompose(**char** ch)	모음 분리 메소드
public static char getVowelCompose(String str)	모음 분리 후 다시 합칠 때 필요한 메소드 모음 ㅘ, ㅞ,ㅒ 같은 모음 구분을 위한 메소드
public static String decompose(String data, **boolean** splitFortis)	분리된 문자열을 모두 붙여서 리턴 분리된 문자열 모두 합침
public static List<String> decomposeList(String data, **boolean** splitFortis)	분리된 각 문자를 리스트에 넣어서 리턴 lIST 리스트 변수를 나중에 점자 변환 클래스에 전달
public static String decompose(**char** nTmp, **boolean** splitFortis)	공식에 맞춰 char를 분리된 2~3개의 char로 변경(String 리턴)
int jong = nTmp % 28; = 종성 **int** jung = ((nTmp - jong) / 28) % 21; = 중성 **int** cho = (((nTmp - jong) / 28) - jung) / 21; = 초성	
public static char compose(**char** x1, **char** x2, **char** x3)	초성 + 중성 + 종성으로 이루어진 글자 합치기

메소드	역할
public static char compose(char x1, char x2)	초성 + 중성으로 이루어진 글자 합치기

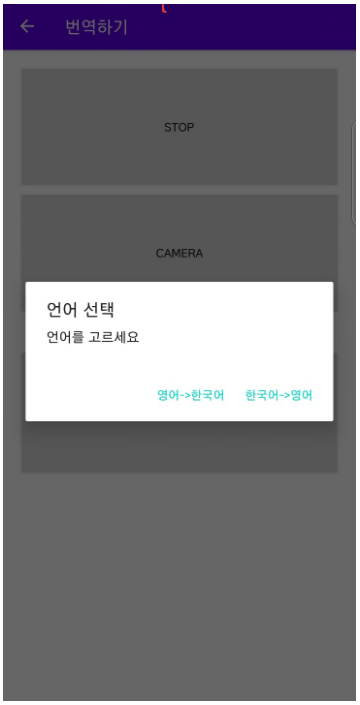
5. Prototype 구현



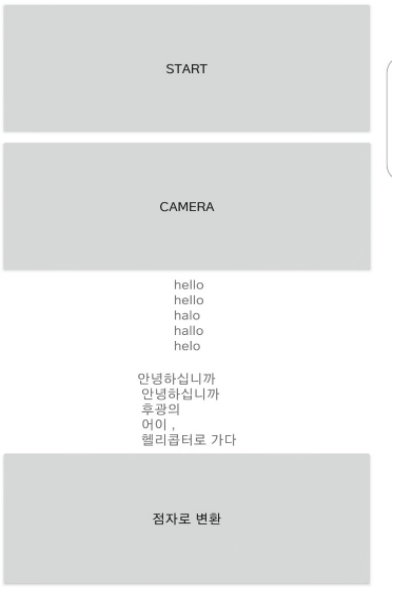
6. 시험 | 테스트 결과

6-1. 음성 번역

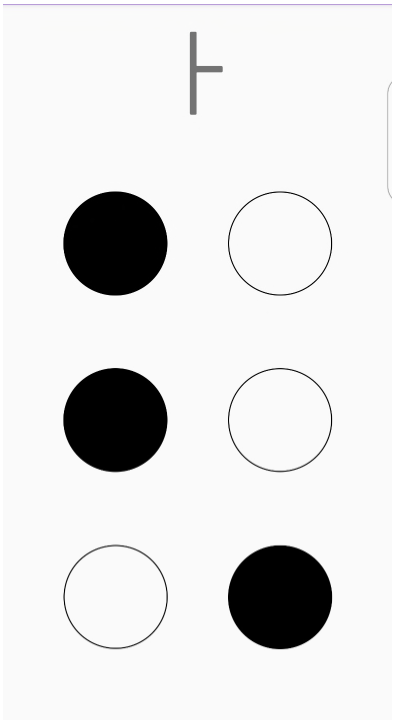
6-1-1. 음성 및 사진 선택창



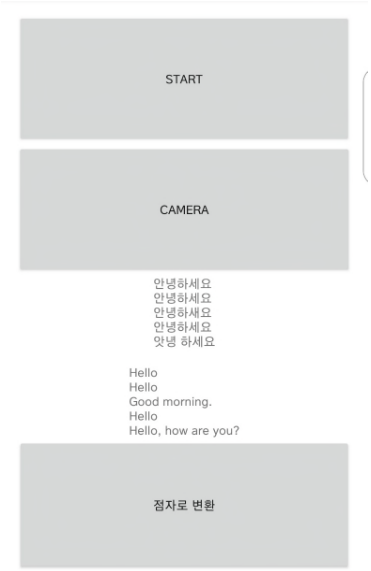
6-1-2. 언어 선택창



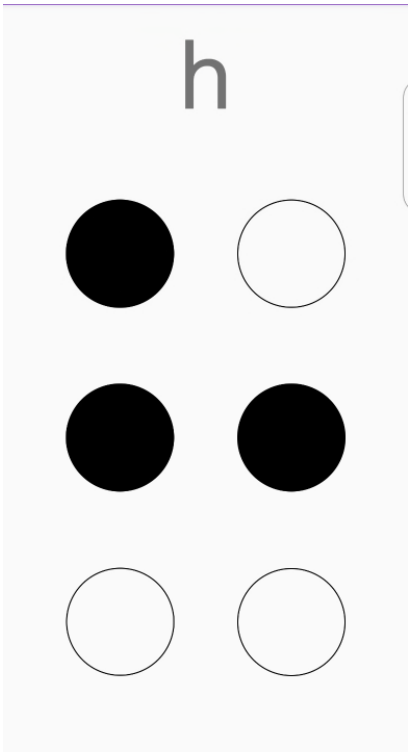
6-1-3. 영어 -> 한국어
번역된 결과창



6-1-4. 영어 -> 한국어
번역된 결과창 점자 변환



6-1-5. 한국어 -> 영어
번역된 결과창



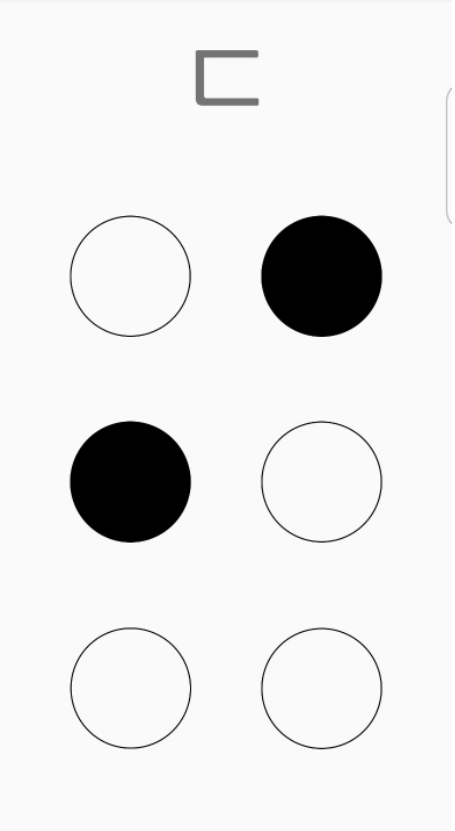
6-1-6. 한국어 -> 영어
번역된 결과창 점자 변환

6-2. 사진 번역

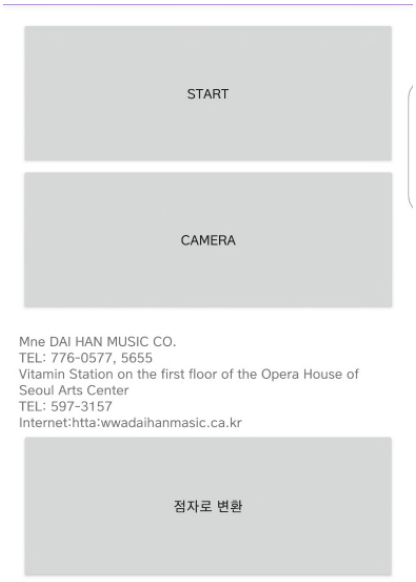
6-2-1. 음성 및 사진 선택창

6-2-2. 사용 예제

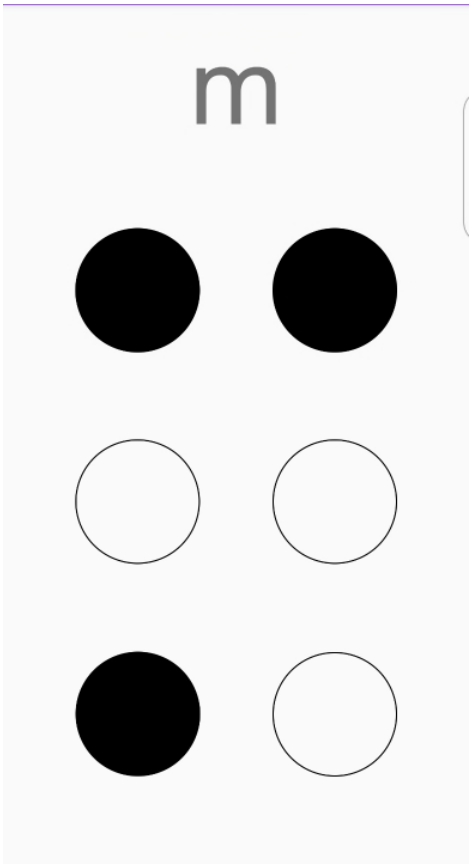
6-2-3. 영어 -> 한국어
번역된 결과창



6-2-4. 영어 -> 한국어
번역된 결과창 점자 변환



6-2-5. 한국어 -> 영어
번역된 결과창



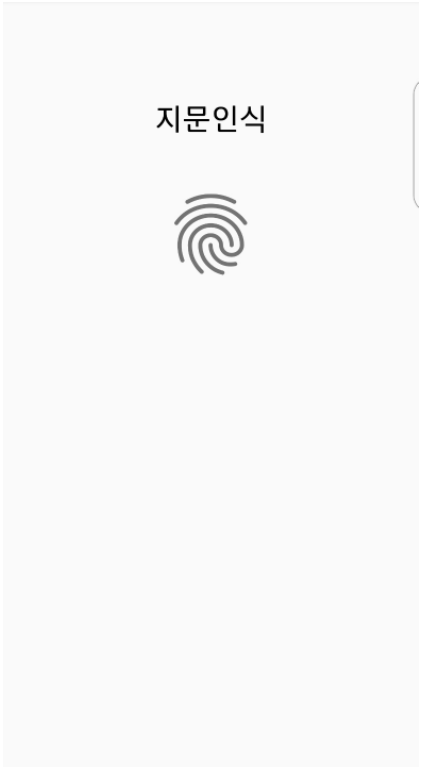
6-2-6. 한국어 -> 영어
번역된 결과창 점자 변환

6-3. 서버 연동을 통한 사용자 검색기록 번역

6-3-1. 로그인 및 회원가입 선택창

6-3-2. 회원가입창

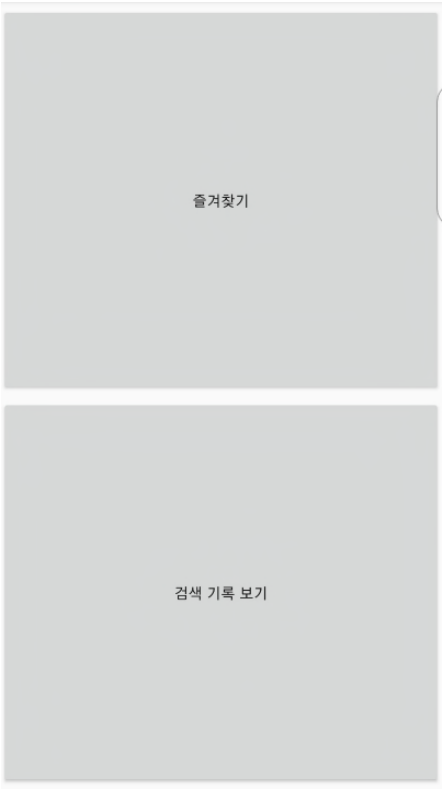
6-3-3. 로그인 선택창



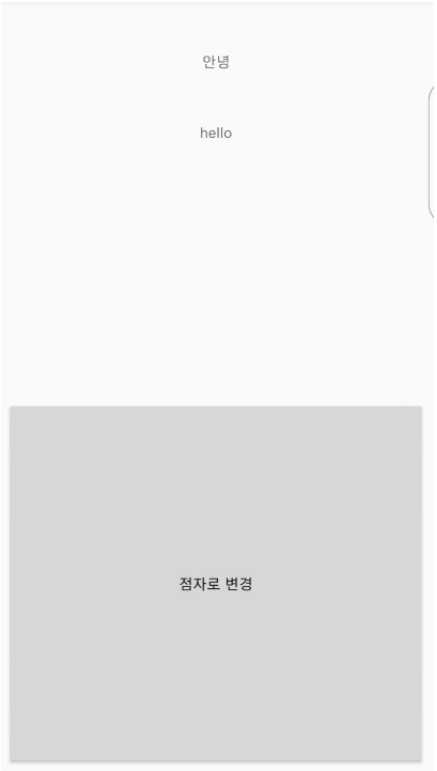
6-3-4. 지문 로그인

6-3-5. 일반 로그인

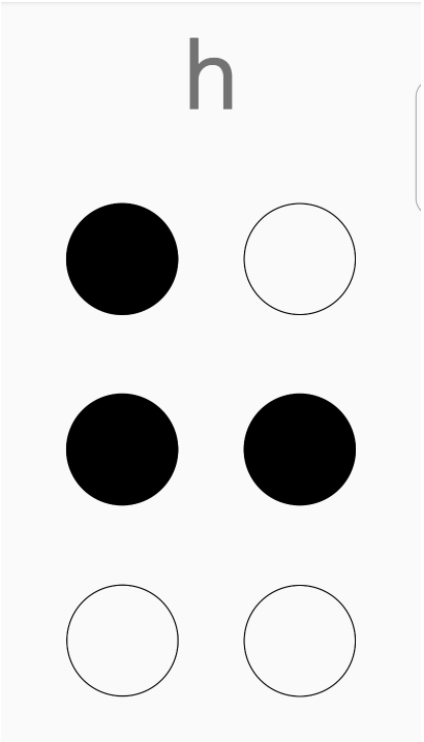
6-3-6. 로그인 후
검색기록 및 번역 선택창



6-3-7. 즐겨찾기 및 검색기록 보기



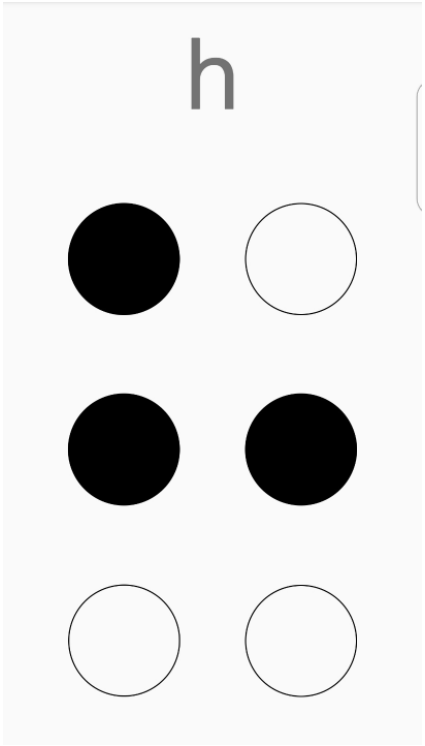
6-3-8. 즐겨찾기 및 검색기록
선택 시 해당 데이터 출력



6-3-9. 점자로 변경

7. Coding & DEMO

7-1. 점자 화면 출력 코드




```
private static int[] arr = {
        R.id.dot_circle_1,
        R.id.dot_circle_2,
        R.id.dot_circle_3,
        R.id.dot_circle_4,
        R.id.dot_circle_5,
        R.id.dot_circle_6
}; // n번째 값과 각 점자에 대한 인덱스를 매칭해줌
```

7-1-1. 점자 인덱스

```
private boolean[] filled = new boolean[] { // 해당 점이 채워졌는가 true면 채워짐
        false, false, false, false, false, false
};
```

7-1-2. 점자 디폴트값, 점자가 채워지면 true로 변경

```
// 해당 인덱스에 대해 채워진 점자, 빈 점자 여부를 설정함
public void setFill(int index, boolean fill) {
    View circle = view.findViewById(arr[index]);
    if(fill) {
        circle.setBackground(getDrawable(getResources(), R.drawable.circle_full,  theme: null));
    } else {
        circle.setBackground(getDrawable(getResources(), R.drawable.circle_empty,  theme: null));
    }
    filled[index] = fill;
}
```

7-1-3. 해당 글자의 점자 정보를 찾는 코드

```
    final Vibrator vibrator = (Vibrator)context.getSystemService(Context.VIBRATOR_SERVICE);

    for(int i = 0; i < 6; i++) {
        final int index = i;
        view.findViewById(arr[i]).setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                if(filled[index]) { // 만약 점자가 채워져있는 상태에서 터치시 500ms간 진동
                    vibrator.vibrate( milliseconds: 500);
                } else
                    vibrator.vibrate( milliseconds: 100);
            }
        });
    }
}
```

7-1-4. 검은색 점자를 누르면 진동을 길게 하얀색 점자는 짧게 울림

7-2. 문장 자모 분리 코드

```
// 자모 분리
decomposed = WordClass.decomposeList(intent.getStringExtra( name: "str"),  splitFortis: false);

Log.d( tag: "handprinting", decomposed.toString());

// decomposed된 단어의 제대로된 길이 구하기
for(int i = 0; i < decomposed.size(); i++) {
    for(int k = 0; k < decomposed.get(i).length(); k++) {
        char ch = decomposed.get(i).charAt(k);

        Log.d( tag: "handprinting",  msg: "ch: " + ch + ", index: " + i + ", " + k);

        if(k == 0) { // 초성인 경우
            if(WordClass.getJongDecompose(ch) != null) { // 만약 두 글자가 합쳐진 경우?
                String decomposedMore = WordClass.getJongDecompose(ch); // 쪼개본다 ㄳ -> ㄱㅅ, ㅃ -> ㅂㅂ
                if(decomposedMore.charAt(0) == decomposedMore.charAt(1)) { // 된소리인 경우(앞뒤가 똑같음) ㄱ == ㅅ -> false, ㅂ == ㅂ -> true
                    fragments.add(createDotFragment( ch: "된소리 " + decomposedMore.charAt(0), fortis)); // 된소리 전용 점자 추가
                    Log.d( tag: "handprinting",  msg: "add: fortis");
                } else { // 된소리가 아닌 경우
                    fragments.add(createDotFragment(String.valueOf(decomposedMore.charAt(0)), map.get(decomposedMore.charAt(0)))); // 첫번째 추가
                    Log.d( tag: "handprinting",  msg: "add: " + decomposedMore.charAt(0));
                }
                fragments.add(createDotFragment(String.valueOf(decomposedMore.charAt(1)), map.get(decomposedMore.charAt(1)))); // 두번째 추가
                Log.d( tag: "handprinting",  msg: "add: " + decomposedMore.charAt(1));
            } else { // 합쳐진거 아니면
                if('A' <= ch && ch <= 'Z') { // 만약 대문자 알파벳인 경우
                    ch += 32; // 소문자로 변경 아스키코드상 a = 97, A = 65 따라서 32를 더하면 소문자로 바뀜
                }
                fragments.add(createDotFragment(String.valueOf(ch), map.get(ch))); // 그대로 추가
                Log.d( tag: "handprinting",  msg: "add: " + ch);
            }
```

```
        } else if(k == 1) { // 중성인 경우
            if(toSplit.contains(ch)) { // 만약 분해해야하는 모음인 경우
                String decomposedMore = WordClass.getVowelDecompose(ch); // 쪼갠다
                fragments.add(createDotFragment(String.valueOf(decomposedMore.charAt(0)), map.get(decomposedMore.charAt(0))));
                fragments.add(createDotFragment(String.valueOf(decomposedMore.charAt(1)), map.get(decomposedMore.charAt(1)))); // 각각 추가
                Log.d( tag: "handprinting",  msg: "add: " + decomposedMore.charAt(0));
                Log.d( tag: "handprinting",  msg: "add: " + decomposedMore.charAt(1));
            } else {
                fragments.add(createDotFragment(String.valueOf(ch), map.get(ch))); // 그대로 추가
                Log.d( tag: "handprinting",  msg: "add: " + ch);
            }
        } else if(k == 2) { // 종성인 경우
            if(WordClass.getJongDecompose(ch) != null) { // 만약 두 글자가 합쳐진 경우?
                String decomposedMore = WordClass.getJongDecompose(ch); // 쪼개본다
                if(decomposedMore.charAt(0) == decomposedMore.charAt(1)) { // 된소리인 경우(앞뒤가 똑같음)
                    fragments.add(createDotFragment( ch: "된소리 " + decomposedMore.charAt(0), fortis)); // 된소리 전용 점자 추가
                    Log.d( tag: "handprinting",  msg: "add: fortis");
                } else { // 된소리가 아닌 경우
                    fragments.add(createDotFragment(String.valueOf(decomposedMore.charAt(0)), mapJong.get(decomposedMore.charAt(0)))); // 첫번째 추가
                    Log.d( tag: "handprinting",  msg: "add: " + decomposedMore.charAt(0));
                }
                fragments.add(createDotFragment(String.valueOf(decomposedMore.charAt(1)), mapJong.get(decomposedMore.charAt(1)))); // 두번째 추가
                Log.d( tag: "handprinting",  msg: "add: " + decomposedMore.charAt(1));
                // 종성이니깐 종성에 맞는 점자를 가져옴
            } else { // 합쳐진거 아니면
                fragments.add(createDotFragment(String.valueOf(ch), map.get(ch))); // 그대로 추가
                Log.d( tag: "handprinting",  msg: "add: " + ch);
            }
        }
    }
}
```

7-2-1. 문장을 하나의 문자로 분리되는 과정 영어와 한글 모두 포함

```
    private DotFragment createDotFragment(String ch, int[] arr) {
        // DotFragment fragment를 view에 추가하는 과정

        // 뷰페이저에 추가.

        if (arr != null) {
            DotFragment fragment = new DotFragment(getApplicationContext());
            // 지정한 인덱스에 따라서 점자를 채워낸다
            for (int i = 0; i < 6; i++) {
                fragment.setFill(i,  fill: arr[i] == 1);
            }
            fragment.setText(ch);
            return fragment;
        } else { // 등록되지 않은 문자가 나오면 ViewPager에 로그를 띄우고 null 반환
            Log.d( tag: "handprinting",  msg: "null 존재");
            return null;
        }

    }
}
```

7-2-2. 문자를 찾아서 점자를 채워나가는 코드

```
public class WordClass {
    // 각종 문자열들 리스트로 등록해두는 과정
    private static List<Character> 초성 = new ArrayList<>(Arrays.asList('ㄱ', 'ㄲ', 'ㄴ', 'ㄷ', 'ㄸ', 'ㄹ', 'ㅁ', 'ㅂ', 'ㅃ', 'ㅅ', 'ㅆ',
            'ㅊ', 'ㅋ', 'ㅌ', 'ㅍ', 'ㅎ'));

    private static List<Character> 중성 = new ArrayList<>(Arrays.asList('ㅏ', 'ㅐ', 'ㅑ', 'ㅒ', 'ㅓ', 'ㅔ', 'ㅕ', 'ㅖ', 'ㅗ', 'ㅘ', 'ㅙ',
            'ㅝ', 'ㅞ', 'ㅟ', 'ㅠ', 'ㅡ', 'ㅢ', 'ㅣ'));

    private static  List<Character> 종성 = new ArrayList<>(Arrays.asList(' ', 'ㄱ', 'ㄲ', 'ㄳ', 'ㄴ', 'ㄵ', 'ㄶ', 'ㄷ', 'ㄹ', 'ㄺ', 'ㄻ',
            'ㄼ', 'ㄽ', 'ㄾ', 'ㅁ', 'ㅂ', 'ㅄ', 'ㅅ', 'ㅆ', 'ㅇ', 'ㅈ', 'ㅊ', 'ㅋ', 'ㅌ',
            'ㅍ', 'ㅎ'));
```

7-2-3. 각종 문자열 리스트 등록한 코드

```
// 된소리를 분리된 두 글자로 바꿔주는 경우의수를 모두 등록
static {
    addJong( ch: 'ㄲ', str: "ㄱㄱ");
    addJong( ch: 'ㄳ', str: "ㄱㅅ");
    addJong( ch: 'ㄵ', str: "ㄴㅈ");
    addJong( ch: 'ㄶ', str: "ㄴㅎ");
    addJong( ch: 'ㄺ', str: "ㄹㄱ");
    addJong( ch: 'ㄻ', str: "ㄹㅁ");
    addJong( ch: 'ㄼ', str: "ㄹㅂ");
    addJong( ch: 'ㄽ', str: "ㄹㅅ");
    addJong( ch: 'ㄾ', str: "ㄹㅌ");
    addJong( ch: 'ㄿ', str: "ㄹㅍ");
    addJong( ch: 'ㅀ', str: "ㄹㅎ");
    addJong( ch: 'ㅄ', str: "ㅂㅅ");
    addJong( ch: 'ㅆ', str: "ㅅㅅ");
```

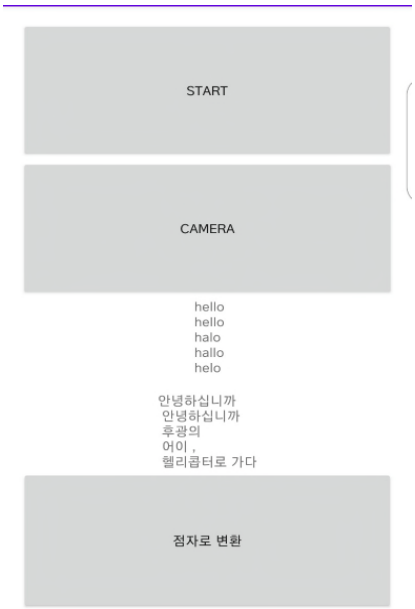
7-2-4. 된소리를 포함한 경우의 수 모두 등록

```
// 공식에 맞춰 char를 분리된 2~3개의 char로 변경(String 리턴)
public static String decompose(char nTmp, boolean splitFortis) {
    if(nTmp>='가' && nTmp<='힣') {
        nTmp -= 0xAC00;

        StringBuilder sb = new StringBuilder( capacity: 3);

        int jong = nTmp % 28;
        int jung = ((nTmp - jong) / 28) % 21;
        int cho = (((nTmp - jong) / 28) - jung) / 21;

        char choResult = 초성.get(cho);

        // splitFortis가 true면 ㄲ 대신 두 자모를 분리해 ㄱㄱ을 리턴
        if(splitFortis && jongDecompose.containsKey(choResult)) {
            sb.append(getJongDecompose(choResult));
        } else {
            sb.append(choResult);
        }

        sb.append(중성.get(jung));

        if (jong != 0) {
            char jongResult = 종성.get(jong);

            if(splitFortis && jongDecompose.containsKey(jongResult)) {
                sb.append(getJongDecompose(jongResult));
            } else {
                sb.append(jongResult);
            }
```

```
            return sb.toString();
        } else {
            String decomposed = getJongDecompose(nTmp);
            if(decomposed!=null) {
                return decomposed;
            } else {
                return String.valueOf(nTmp);
            }
        }
    }
```

7-2-5. 한글 및 영어의 아스키코드값 찾아서 분리

7-3. 음성 및 사진 텍스트 변환 코드



```java
btnStart2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // 언어를 고르라는 Dialog 보여주기
        AlertDialog.Builder builder = new AlertDialog.Builder( context: MainActivity.this);
        builder.setTitle("언어 선택").setMessage("언어를 고르세요").setPositiveButton( text: "한국어->영어", new DialogInterface.OnCl
            @Override
            public void onClick(DialogInterface dialog, int which) { // Dialog에서 무언가를 골랐으면 시작지점을 한국어로 설정했는지
                startKo = true; // 분석할때 시작을 한국어로 했으면 true
                Intent intent = new Intent(getApplicationContext(), photoRecognizer.class);
                startActivity(intent);
            }
        }).setNegativeButton( text: "영어->한국어", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) { // Dialog에서 무언가를 골랐으면 시작지점을 한국어로 설정했는지
                startKo = false;
                Intent intent = new Intent(getApplicationContext(), photoRecognizer.class);
                startActivity(intent);
            }
        });
        AlertDialog alertDialog = builder.create();
        alertDialog.show();

    }
});
```

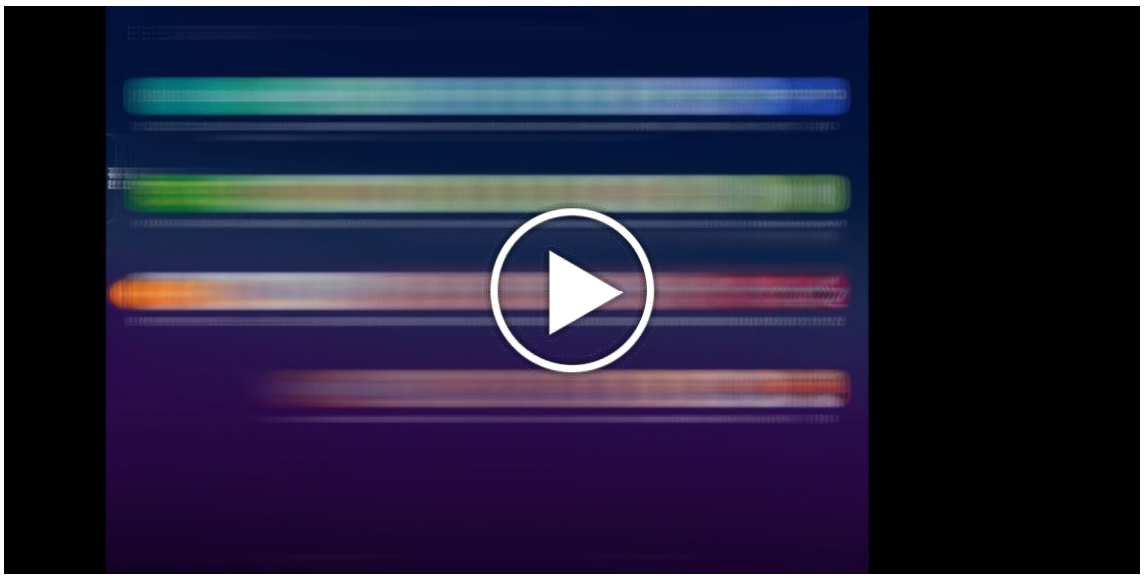
7-3-1. 인식할 사진 및 음성 언어 선택하는 다이얼로그

```java
/ 2. SpeechRecognitionListener를 상속한 클래스를 정의합니다.
lass NaverRecognizer implements SpeechRecognitionListener {
   /* 텍스트를 영어로 변환하기 위한 소스코드입니다. */
   public TextView translatedView;
   public String textForTranslated;
   public String translated;
   public String address = "https://openapi.naver.com/v1/papago/n2mt";

   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; }
```

7-3-2. 음성 인식 텍스트화 클래스 및 메소드

Ⅲ. 결론

1. 연구 결과



2. 작품제작 소요재료 목록

시각장애인 학생들을 위한 스마트번역 애플리케이션 제작으로 해당 사항 없음

Ⅳ. 참고자료

1. 참고자료

[1] http://www.kpu.ac.kr/contents/main/cor/vision.html

[2] http://www.kmoumedia.com/news/articleView.html?idxno=1205

[3] http://www.mediatoday.co.kr/news/articleView.html?idxno=132086

[4] http://www.mediatoday.co.kr/news/articleView.html?idxno=132086

[5] https://www.huffingtonpost.kr/dongshin-yang/story_b_13033828.html

[6] https://www.welfarenews.net/news/articleView.html?idxno=66047

[7] https://1boon.kakao.com/bloter/307031

[8] https://100up.kakaoimpact.org/problems/1/view

[9] https://100up.kakaoimpact.org/problems/1/view