AI기반 카카오톡 연동 TTS 음성 안내 IOT 앱 개발

목차

프로젝트 소개

Our Voice
_Hands Free
System

스마트 폰 조작이 어려운 여러 상황을 고려해 메시지 내용을 음성으로 출력하여 손 조작 없이 메시지를 읽어주고 조작할 수 있도록 하기 위해 앱을 제작하였다.

특히나 운전 중 사고 방지를 주 목표로 하여 제작하였다.

-이벤트 정보 분석
-텍스트 정보 추출
-TTS엔진 음성 출력
-블루투스 통신

프로젝트 소개

배경/필요성

스마트 폰 사용이 보편화된 시대에서 사용자 편의를 위한 스마트 폰 관련 개발을 계속되고 있습니다.
이는 몇 년 전부터 소형 블루투스 이어폰, 블루투스 홈 기기 등 굳이 직접 손으로 조작하지 않아도 기능을 할 수 있는 형태로 다양하게 나타나고 있습니다. 이와 마찬가지로 사용자들은 스마트 폰을 통한 의사소통이 익숙해졌고 좀 더 편리한 조작을 추구해오고 있습니다. 그러나 아직 해결하지 못한 불편함도 존재합니다.

운전 중 스마트 폰 조작, 바쁜 업무로 인한 메신저 조작의 불편함, 운동, 요리 등 여러 상황에서 스마트 폰을 주시하며 메신저 활동을 지속하는데 문제가 있습니다. 위와 같은 상황에서 사용자에게 필요한 정보가 들어있는 카카오톡 메시지를 바로 확인 할 수 있도록 본 앱을 제작하였습니다. 카카오톡이 전달받은 메시지 내용을 TTS 기술을 이용하여 음성으로 전달한다면 간과되었던 운전 중 스마트 폰 조작 문제를 해결할 수 있습니다.

특징/장점

기존의 음성 합성 기술들을 사용한 어플리케이션들의 경우, 텍스트 처리 기능보다는 얼마나 잘 읽어내는지에 대해서만 포커스를 두며 해당 API들을 적용하는 데에 초점을 둔다고 한다면, 본 앱은 사용자에게 필요로 하는 메시지들만 전달하는 데에 목적을 두고 있기에 자연스러운 음성과 편리한 조작을 중점으로 한다. 어플리케이션 자체에 소리와 블루투스 기능을 두어, 핸드폰 내의 설정이 아닌 앱 내부에서도 기기의 설정에 필요시에 빠르게 접근할 수 있도록 하여 편리성을 높이고자 하였다.
더 나아가 정보 필터링, 장애인 음성인식 도움 등 여러 요소를 발전시켜 상업적 구현도 가능한 발전가능 앱입니다.

프로젝트 소개

Hands Free

Driving

KaKaoTalk

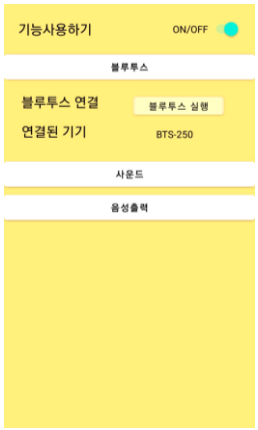
Feture

기능

화면 설계

권한 요청: 인트로에 권한허용 요청과 블루투스 사용허용 팝업

기능사용: 기능사용 on/off 를 추가하여 더 깔끔하고 편리한 디자인



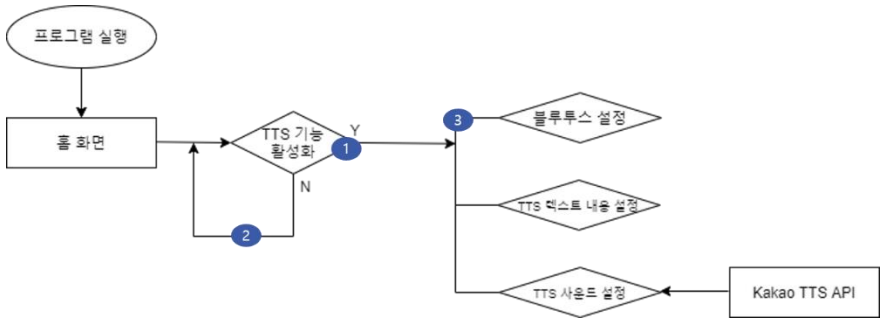
블루투스 연결: 카카오톡 메시지를 출력할 준비가 되면 어떤 기기를 통해서 출력할지 설정하도록 한다. 연결된 기기 탭 에는 현재 블루투스로 연결된 기기 이름이 출력된다. 블루투스 기기에 연결하지 않는 경우에는 사용자 스 마트폰 자체에서 출력된다.

사운드 설정: 카카오톡 TTS를 받을 때 사운드 크기를 조절할 수 있다. 앱 내에서 크기와 속도를 조절할 수 있어 사 용자 편의를 높인다.

TTS엔진 통한 설정: 카카오톡을 TTS로 출력할 시 수신받을 내용을 상세 설정할 수 있다. 발신시간/발신자/발신내용을 사용 자가 선택하여 출력하도록 한다.

기능

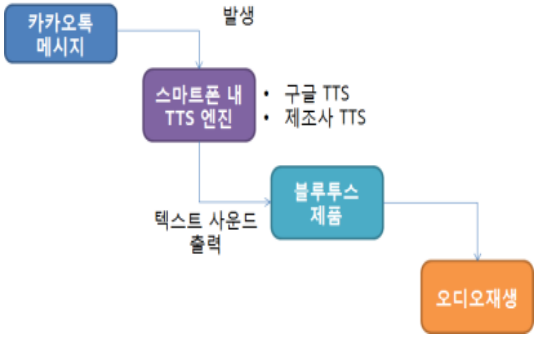
기능 처리

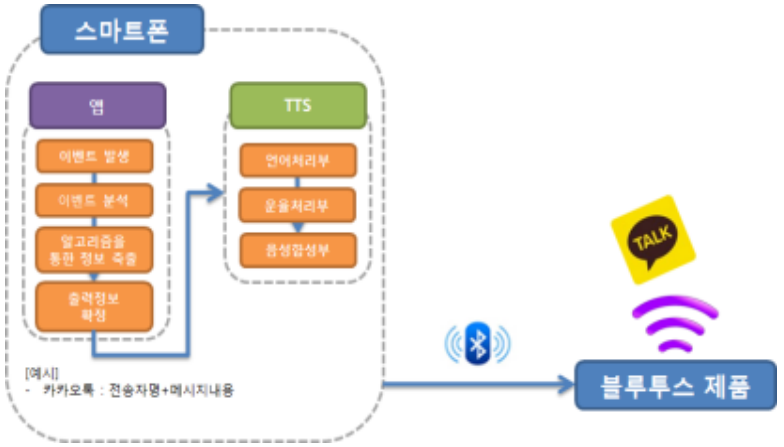


1. 어플리케이션을 실행한 후, 홈 화면에서 TTS 기능 활성화 여부를 클릭한다.
2. 활성화가 되었다면 자동으로 카카오톡 메신저의 메시지를 하나씩 받아서 TTS로 송출을 시작하게 된다. 하지만 활성화되지 않았을 경우, 활성화 버튼이 클릭될 때까지 어떠한 메시지도 TTS로 송출하지 않는다. 또한 블루투스 설정, TTS 텍스트 내용 설정, TTS 사운드 설정 기능들의 접근 자체가 불가능하다.
3. 활성화 되었을 경우, 블루투스 설정, TTS 텍스트 내용 설정, TTS 사운드 설정 등을 통해 TTS 서비스의 기능을 구체적으로 설정할 수 있으며, 이는 실시간으로 메시지가 전달될 때마다 반영되어 나타나게 된다.

기능

기능 처리





주요 기능에는 이벤트 정보 분석,텍스트 정보 추출,TTS 엔진을 통한 음성 출력 ,블루투스 통신을 통해 제품에 음성 전달 총 네 가지 구성.

기능

핵심소스

메인
If문_전반적인 TTS

```
public class MainActivity extends Activity {

    private  final int CHECK_CODE = 0x1;
    private  final int LONG_DURATION =5000;
    private  final int SHORT_DURATION =1200;

    private Speaker speaker;

    private ToggleButton toggle;
    private CompoundButton.OnCheckedChangeListener toggleListener;
    private TextView smsText;
    private TextView smsSender;

    private BroadcastReceiver smsReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        toggle = (ToggleButton) findViewById(R.id.speechToggle);
        smsText = (TextView) findViewById(R.id.sms_text);
        smsSender = (TextView) findViewById(R.id.sms_sender);

        toggleListener = (OnCheckedChangeListener) (buttonView, isChecked) → {
                if(isChecked){
                    speaker.allow( allowed: true);
                    speaker.speak( text: "메세지 읽기를 시작합니다");
                } else {
                    speaker.speak( text: "중단합니다");
                    speaker.allow( allowed: false);
                }
        };
```

SMS
메시지 발생관련

```
private void initializeSMSReceiver() {
    smsReceiver = (BroadcastReceiver) (context, intent) → {
            Bundle bundle = intent.getExtras();
            if(bundle != null){
                Object [] pdus = (Object[]) bundle.get("pdus");
                for(int i = 0; i < pdus.length; i++){
                    byte[] pdu = (byte[]) pdus[i];
                    SmsMessage message = SmsMessage.createFromPdu(pdu);
                    String text = message.getDisplayMessageBody();
                    String sender = getContactName(message.getOriginatingAddress());
                    speaker.pause(LONG_DURATION);
                    speaker.speak( text: sender + "님에게서 " + sender + "메세 문습니다");
                    speaker.pause(SHORT_DURATION);
                    speaker.speak(text);
                    smsSender.setText("보낸사람은" + sender);
                    smsText.setText(text);
                }
            }
    };
}
private String getContactName(String phone){
    Uri uri = Uri.withAppendedPath(ContactsContract.PhoneLookup.CONTENT_FILTER_URI,
            Uri.encode(phone));
    String projection[] = new String[]{ContactsContract.Data.DISPLAY_NAME};
    Cursor cursor = getContentResolver().query(uri, projection,  selection: null, selectionArgs: null, sortOrder: null);
    if(cursor.moveToFirst()){
        return cursor.getString( columnIndex: 0);
    } else {
        return "알수없는 번호";
    }
}

private void registerSmsReceiver() {
    IntentFilter intentFilter = new IntentFilter( action: "android.provider.Telephony.SMS_RECEIVED");
    registerReceiver(smsReceiver,intentFilter);
}

}
```

TTS엔진
onInit

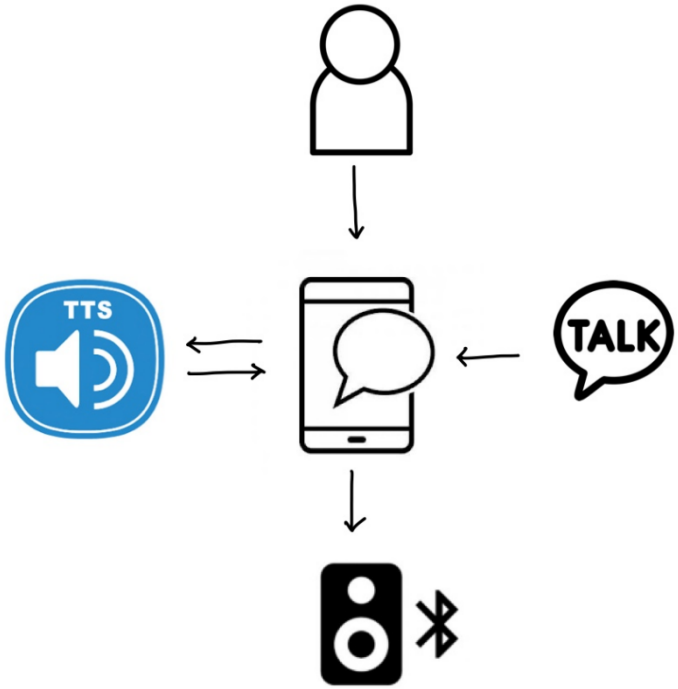
```
public class Speaker implements TextToSpeech.OnInitListener {

    private TextToSpeech tts;
    private  boolean ready = false;
    private boolean allowed  = false;

    public Speaker (Context context) {
        tts  = new TextToSpeech(context,  listener: this);
    }

    public boolean isAllowed () {
        return allowed;
    }

    public void allow (boolean allowed) {
        this.allowed = allowed;
    }

    @Override
    public void onInit(int status) {
        if (status == TextToSpeech.SUCCESS) {
            tts.setLanguage((Locale.KOREAN));
            ready = true;
        }else {
            ready = false;
        }

    }
```

Main.java코드로 브로드캐스트 리시버를 private선언, 각xml에 선언한 id를 findViewById로 기능과 버튼 연결 /OnCheckedChangeListener를 사용하여 TTS엔진이 생성한 버튼을 눌렀을 시 읽어 들일 수 있도록 선언하였다. if문을 쓴 이유는 실행과 중단할 시에 각각 나와야 할 음성이 다르기 때문에 사용하였다.

Broadcast Reciever를 사용하여 메시지가 발생할 시 화면에 띄우기 위한 소스 코드이다. string과 byte를 사용하여 글자를 가져올 수 있을 내용의 크기와 글을 선언한다. text와 sender를 선언한 후 메시지가 발생하였을 경우 화면에 띄우며 발생시 새메시지가 왔다는 알림음을 띄우도록 speak함수를 사용하였다.

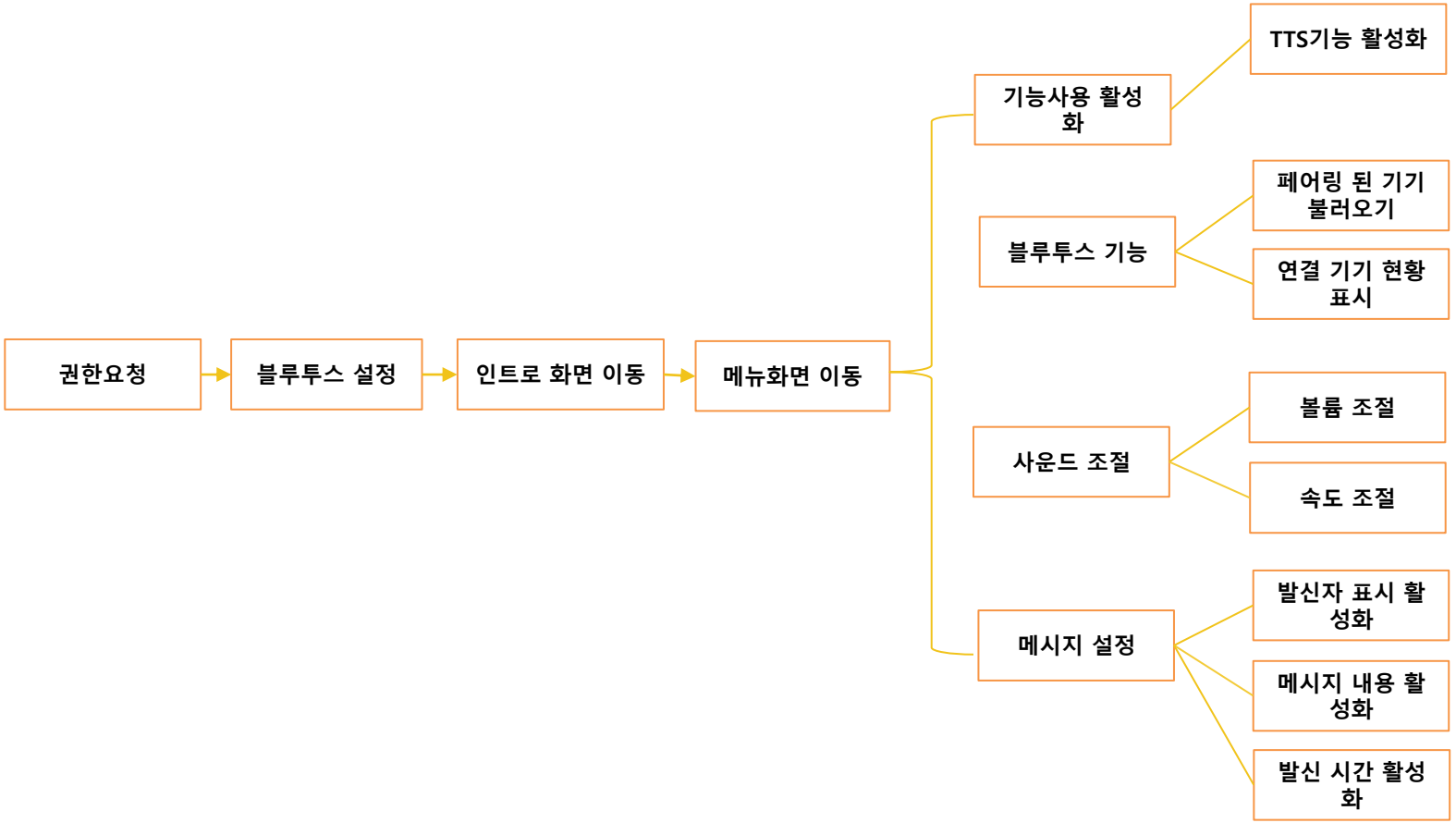
Speaker.java 파일이다. TTS엔진을 사용할 수 있도록 선언한 자바 코드이며 text to speech를 사용하기 위해 선언을 해준 후 onInit을 사용하여 기능이 사용가능하도록 클래스를 선언한다. 언어의 경우 한글을 사용하기 때문에 Locale.KOREAN으로 사용한다. 허용이 가능하도록 함수를 짜주며 반드시 text to speech함수를 써야 가능하다.

구성도

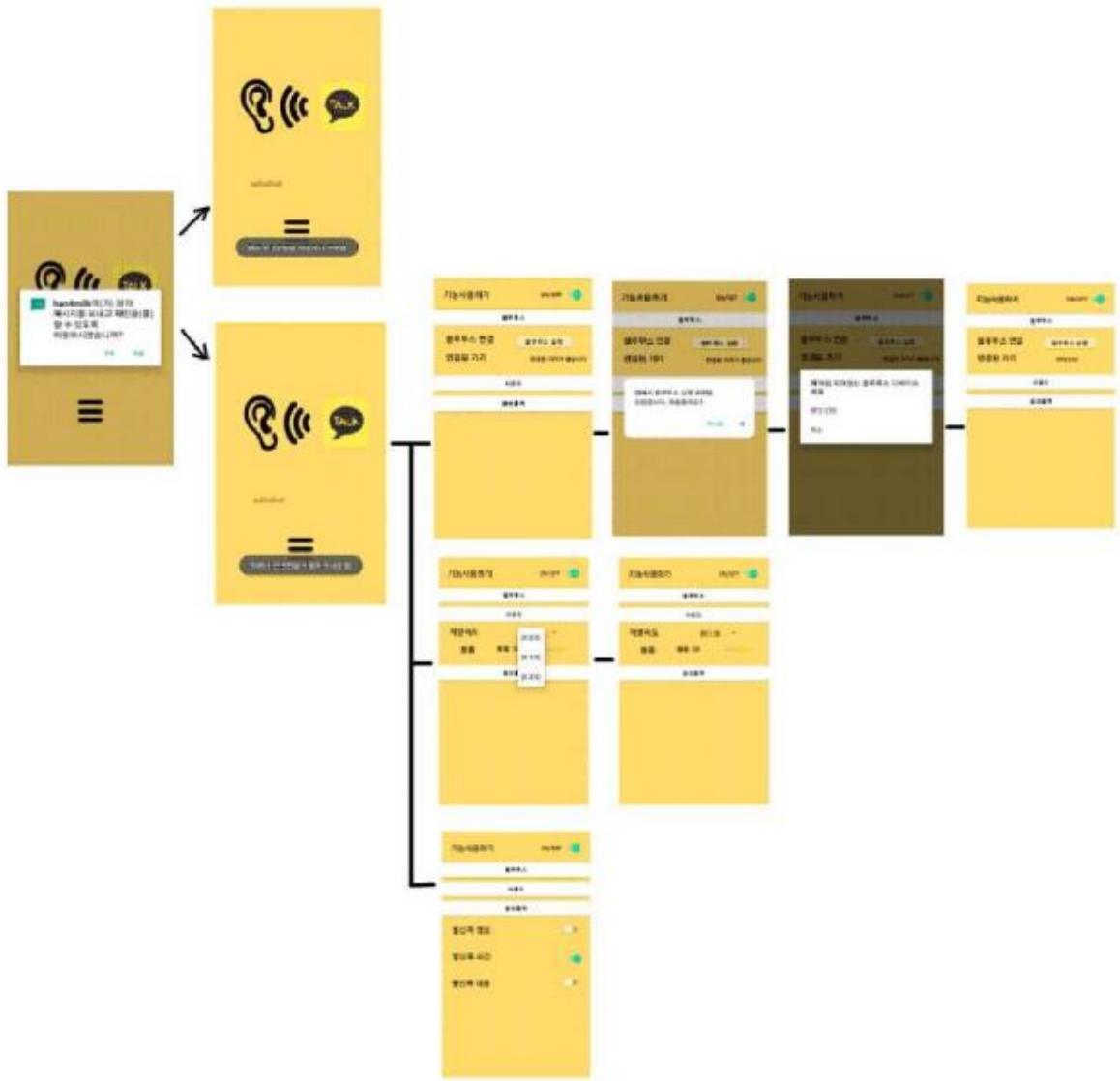


① 사용자가 앱 실행 후 TTS 기능 활성화 여부를 클릭.
②카카오톡 앱에서 받는 메시지들을 디바이스 내부의 Notification Listener를 통해 내용 받아 옴.
③TTS기능의 활성화 여부에 따라 받아온 메시지와 관련 정보를 모두 음성 합성 API의 인자로 넘겨 처리.
④ Text to Speech된 음성을 실시간으로 사용자에게 출력
⑤ 앱 내부에 구체적인 음성 상세 설정 가능.
⑥블루투스 디바이스에서 출력가능.

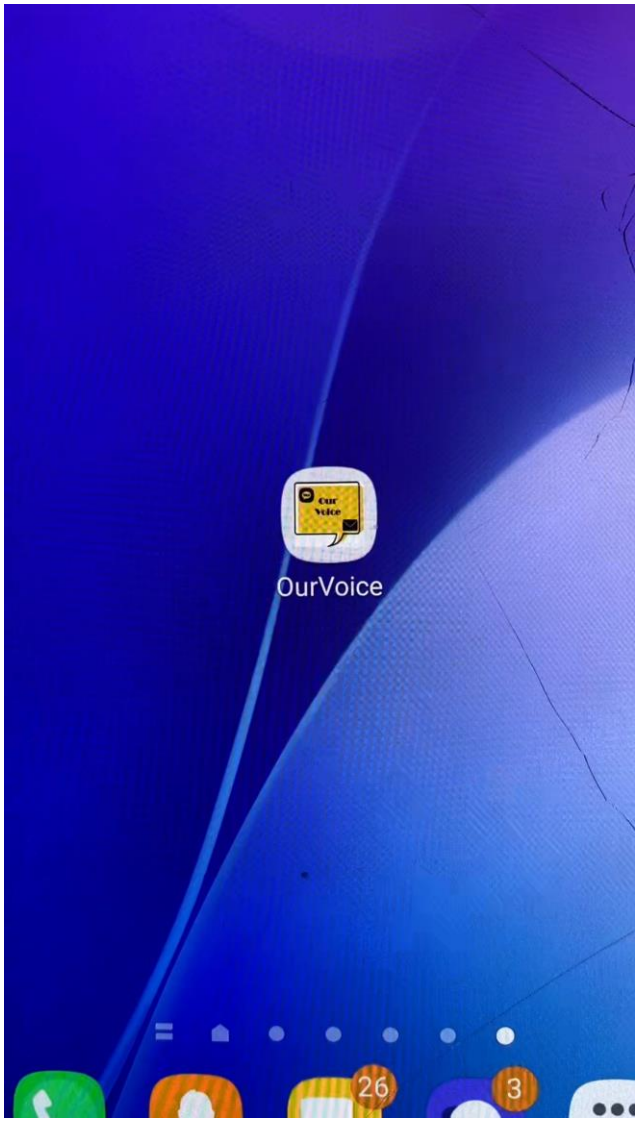
구성도


권한요청
블루투스 설정
인트로 화면 이동
메뉴화면 이동
기능사용 활성화
TTS기능 활성화
블루투스 기능
페어링 된 기기 불러오기
연결 기기 현황 표시
사운드 조절
볼륨 조절
속도 조절
메시지 설정
발신자 표시 활성화
메시지 내용 활성화
발신 시간 활성화

구성도



리뷰



앱 실행 영상.

리뷰

- 코로나로 인해 오프라인 만남이 어려워지면서 팀원들과 의사소통의 제약이 있어 개발의 불편함이 많았다. 화상 회의를 통해 진행현황을 공유하며 해결해 나갔다.

- 형상관리 경험에 부족함이 있어 팀원들과 원활한 관리 환경을 가꾸지 못한 것에 아쉬움이 있다.

- 화면 내부 설계에서 음성에 관한 세부 조작들이 앱을 실행하는 동안 초기화되는 현상을 해결하는데 어려움이 있었다. 이 부분을 해결하기 위해 많은 해결 코드를 접해 볼 수 있었고 해결하게 되었다.

- TTS엔진을 통해 음성이 출력되는 과정에서 말소리가 끊긴다 던지 출력 문제를 직면하게 되었다. 생각보다 간단한 문제로 출력 시 공백처리를 추가하여 음성 속도와 출력 속도를 조정하여 끊어져 들리는 문제를 해결했다.

-Bluetooth 통신: 어플리케이션 메뉴화면에서 사용자가 한눈에 볼 수 있도록 블루투스 페어링 된 목록을 바로 확인 가능하게 구현하고 연결기기를 출력하는 과정에서 기기 출력 결과가 나타나지 않아 블루투스 부분에 전반적인 코드 수정이 이루어졌다.

리뷰

진행 일정

5월
계획서 기반 방향 설정

6월
화면설계/내부화면 설계

7월
이벤트 받아오는 기능 구현
TTS엔진 개발
블루투스 연동

8월
이벤트 받아오는 기능 구현
TTS엔진 개발
블루투스 연동

9월
이벤트 받아오는 기능 구현
TTS엔진 개발
블루투스 연동

10월
애플리케이션 테스트

11월
프로젝트 마무리

Thank you!