

DEMO: <https://youtu.be/8GiGhAlIYZ0>

# 사용자 인터페이스 및 실습

## —최종 보고서—



과 목 : 사용자 인터페이스 및 실습 (가반)  
담 당 교 수 : 최지웅 교수님  
팀 이 름 : 삼겹살  
– 김봉균 컴퓨터학부/20192891  
– 박성준 건축학부/20170872  
– 최성률 소프트웨어학부/20150283  
제 출 일 : 2020.12.13

# 목차

## 1.작업 분해표 (최종 계획안)




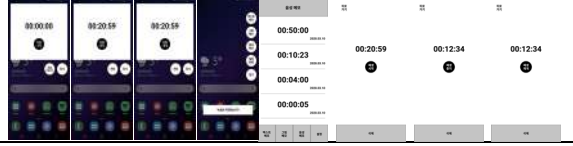


## 2.주차 별 진도보고서

- 1) 8주차 – 12주차 (10.17 ~ 11.21)
- 2) 13주차(11.22 – 11.28)
- 3) 14주차(11.29 – 12.4)
- 4) 15주차(12.5 – 12.11)
- 5) 주차별 진행사항 작업 목록

## 3.작업 화면 & 설명

## 4.최종 작업 분해표

## 1.작업 분해표 (최종 계획안)

화면	기능	담당자	난이도
	플로팅 액션 버튼	최성률 20150283 소프트웨어학부	★★★★★
	텍스트 메모	박성준 20170872 건축학부	★★★★☆
	그림 메모	최성률 20150283 소프트웨어학부	★★★★☆
	음성 메모	박성준 20170872 건축학부	★★☆☆☆ ▽ ★★★★☆
	일정 등록 (캘린더 연동, 파이어베이스)	김봉균 20192891 컴퓨터학부	★★★★★
	설정 창	박성준 20170872 건축학부	★★★★☆

- 진행 중 음성메모 기능을 구현하면서 API 차이로 인한 에러와 그 에러를 해결하는데 많은 시간이 걸려 난이도가 상승

## 2.주차 별 진도 보고서

### 1) 8주차 – 12주차 진도 보고서

2020년 11월 20일 ~ 2020년 11월 27일

#### ■ 8주차 진행사항

내용	진행자	진행율
<ul style="list-style-type: none"> <li>- 와이어프레임 작성</li> <li>- 각 버튼, 기능별 동작 정의</li> </ul>	김봉균	80%
<ul style="list-style-type: none"> <li>- 유사 어플 비교</li> <li>- 제안서 피피티 &amp; 로고 컨셉 디자인</li> </ul>	박성준	90%
<ul style="list-style-type: none"> <li>- 플로팅 버튼 기능 분석</li> <li>- 각 메모 기능(그림, 음성, 텍스트, 일정) 분석</li> </ul>	최성률	90%

#### ■ 9주차 진행사항

내용	진행자	진행율
<ul style="list-style-type: none"> <li>- 발표준비</li> <li>- 와이어프레임 작성</li> <li>- 각 버튼, 기능별 동작 정의</li> </ul>	김봉균	100%
<ul style="list-style-type: none"> <li>- 발표준비</li> <li>- 텍스트 메모 비슷한 어플 3개 시스템(레퍼런스 조사) ListView, RecyclerView, CardView</li> </ul>	박성준	70%
<ul style="list-style-type: none"> <li>- 프로토타입 디자인(Invision 이용)</li> <li>- 플로팅 버튼 설계</li> </ul>	최성률	80%

## 특이 사항

- 플로팅 버튼 설계

플로팅 버튼 기능 예제를 실습을 해보았지만 예제가 오래되어 Android SDK가 다른 점, API 업그레이드되고 권한 설정이 다른 점 등의 여러 문제가 생겨 시간이 오래 걸리고 있다.

## ■ 10주차 진행사항

내용	진행자	진행율
<ul style="list-style-type: none"><li>- 일정 기능을 구글 캘린더에 저장시킬지 로컬에 저장시킨 후 파이어베이스를 이용해 백업시킬지에 대한 조사</li></ul>	김봉균	90%
<ul style="list-style-type: none"><li>- Recyclerview를 통해 텍스트메모, 음성메모, 그림메모에 적용할 수 있는지 사례조사</li></ul>	박성준	80%
<ul style="list-style-type: none"><li>- Floating Action Button 재설계</li><li>- startService로 Service 컴포넌트 실행</li><li>- 권한 설정(depends on API version)</li></ul>	최성률	90%

## 특이 사항

- startService로 Service 컴포넌트 실행

startService로 Service 컴포넌트를 실행하여 Window에서 View를 띄웠다. 이를 bound Service 형태로 바꿔 Activity가 pause 상태에서만 view가 보이고 Activity가 종료되면 Service도 같이 따라 죽게하기 위한 예제를 찾으며 프로젝트에 적용해 구조를 재설계하고 있다.

## ■ 11주차 진행사항

내용	진행자	진행율
<ul style="list-style-type: none"> <li>- 인앱에서 5개 페이지의 탭 만들고 아이콘 할당하기</li> <li>- 어플 테마 색상 변경</li> </ul>	김봉균	100%
<ul style="list-style-type: none"> <li>- Text Memo 기능 구현 스테디</li> <li>- 텍스트 메모 리스트를 위한 recycler view와 RecyclerView Adapter 생성 후 수정 저장 기능 구현</li> </ul>	박성준	70%
<ul style="list-style-type: none"> <li>- Floating Action Button 설계</li> <li>- Floating Action Button view 이동</li> </ul>	최성률	80%

## 특이 사항

- RecyclerView

RecyclerView를 통해 리스트를 만드는 작업에서 리스트를 구현하는 도중 AVD 어플이 실행되다가 중간에 멈추는 현상이 생겨 해결에 예상보다 시간이 오래 걸리고 있음 (RecyclerView에서 xml 또는 다른 클래스와 내용이 달라 문제가 생긴걸로 예상)

- Floating Action Button view 이동

View의 xml을 RelativeLayout view 안에 ImageButton view로 설계하였지만 Window(최상위)에서 View를 이동하는 과정에서 onClickListener가 방해되어 onTouchListener로 onClick을 구현하였다.

→ currentTimeMillis() 로 시간을 체크하여 짧은 시간은 onClick처럼 되도록 구현

→ 추후 Alpha 조절

.....

## ■ 12주차 진행사항

내용	진행자	진행율
<ul style="list-style-type: none"> <li>- fragment를 이용한 탭 기능 구현</li> <li>- 이미지 메모, 음성 메모에서 recycler view를 이용한 목록 가져오기 구현</li> <li>- 텍스트 메모를 sqlite를 이용해 db와 연결</li> <li>- gray color scale을 colors.xml을 통해 정의하였음</li> </ul>	김봉균	90%
<ul style="list-style-type: none"> <li>- RecorderMemo(녹음기능) 구현 스터디 —&gt;예제, 사례 분석</li> <li>- Recorder recycler view 목록 구현</li> </ul>	박성준	80%
<ul style="list-style-type: none"> <li>- Floating Action Button 설계</li> <li>- Floating Action Button (애니메이션) 전환</li> </ul>	최성률	80%

## 특이사항

- fragment를 activity 내에 어떻게 적용시킬지에 대해 이해에 시간이 걸리는 중.
- Floating Action Button (애니메이션) 전환

전환에 대해서는 Layout inflate를 이용해 window manager로 addView와 removeView를 적절하게 사용하여 구현하였지만 애니메이션은 권한 문제로 애니메이션 실행(실습과 동일하게)이 되지 않는다.

→ 직접 컨테이너 코드를 작성하여 window manager로 updatelayout을 일정 주기(프레임)로 애니메이션처럼 구현할 예정

.....

## 2) 13주차

2020년 11월 20일 ~ 2020년 11월 27일

### ■ 13주차 진행사항

내용	진행자	진행율
<ul style="list-style-type: none"><li>- 메모 작성 시 작성 시간을 저장하는 포맷 확정, 적용</li><li>- 그림 메모에 DB 연동</li><li>- 일정 메모에 동일한 과정 진행</li></ul>	김봉균	90%
<ul style="list-style-type: none"><li>- Mediarecorder 를 통해 음성 녹음 기능 구현</li></ul>	박성준	70%
<ul style="list-style-type: none"><li>- 여러 Floaing Dialog View를 동적으로 생성 가능하도록 재설계(모듈과 클래스 분할) (Service code → Service, FloatingView, FloatingManager)</li></ul>	최성률	80%

### 특이사항

- Floating View

View List의 View group일 때, 동적으로 설계한 클래스 코드 때문에 각 View들이 중구난방으로 움직이는 문제가 있다. (FloatingView code를 작성하여 FloatingViewManager Code에서 여러 개의 FloatingView 타입으로 inflate 하여 객체로 사용중이다) ——> View가 반응한 순서별로 State를 나누어 View들의 중구난방 움직임을 제한할 예정



### 3) 14주차

2020년 11월 27일 ~ 2020년 12월 04일

#### ■ 14주차 진행사항

내용	진행자	진행율
<ul style="list-style-type: none"><li>- 탭 레이아웃 디자인 수정</li><li>- DB 접근에러 수정</li><li>- DrawingMemo 탭의 새로고침 기능을 참고하여 TextMemo, Calendar에도 새로고침 기능 추가</li></ul>	김봉균	90%
<ul style="list-style-type: none"><li>- RecordMemo DB 구현, 재생 구현</li><li>- Parcelable 데이터로 구성,</li></ul>	박성준	90%
<ul style="list-style-type: none"><li>- FloatingActionButton 설계 완료</li><li>- DrawingMemo 구현(메모하기, Share, Save, Paint 색 바꾸기, Erase, 등)</li></ul>	최성률	90%

#### 특이사항

- DrawingMemo 구현에서 'share' 기능으로 공유할 경우에 파일로 저장하고 공유할 목적이었으나, Android 7.0(누가)부터 URI가 file:/// 로 시작하면 FileUriExposedException 이 발생하여 난관이 있었다. 해결 방안으로 공유 저장소 MediaStore.images 에 저장하여 content:// 로서 URI의 string이 나오고 Uri.parse()를 통해 Uri를 인텐트에 extra에 넣고 공유(Share) 액티비티를 띄웠다.

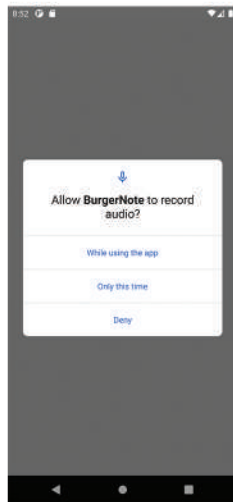
#### 4) 주차 별 진행사항 작업 목록

구분	상세구분	8주 차	9주 차	10주 차	11주 차	12주 차	13주 차	14주 차	15주 차
프로젝트 계획	제안서 작성								
	제안서 수정								
자료 조사 및 분석	유사 시스템 분석								
	레퍼런스 조사								
시스템 설계	플로팅 버튼 설계								
	레이아웃 작성(in app)								
기능 구현	1.텍스트 메모								
	2. 그림 메모								
	3. 음성 메모								
	4. 일정 등록								
	DB 연동								
보완 및 추가 작업	파이어베이스 백업 구현								
	버그 수정								
테스트 및 완성	디자인 검토								
	테스트								

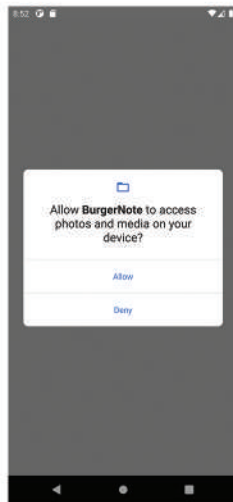
### 3. 작업 화면 & 설명

〈인앱〉

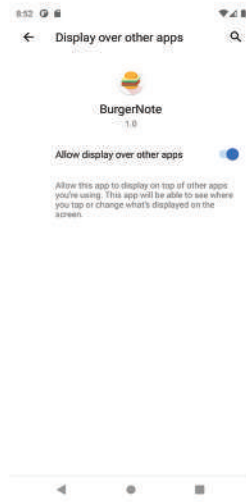
#### 1. 퍼미션



〈음성녹음 퍼미션〉



〈내부저장소 접근 퍼미션〉



〈오버레이 퍼미션〉

#### 2. 텍스트 메모



〈리스트〉



〈텍스트 메모 작성 화면〉

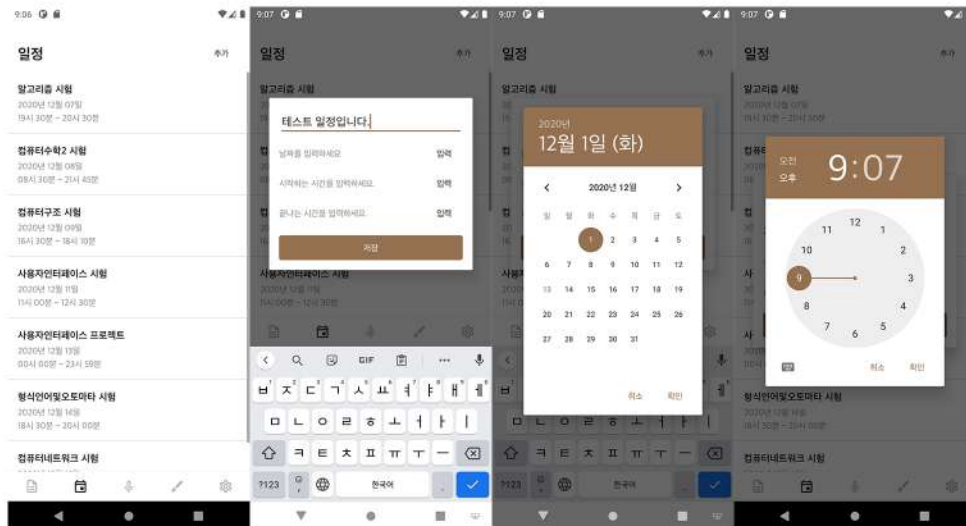


〈저장 후 리스트〉



〈텍스트메모 상세〉

### 3. 그림 메모

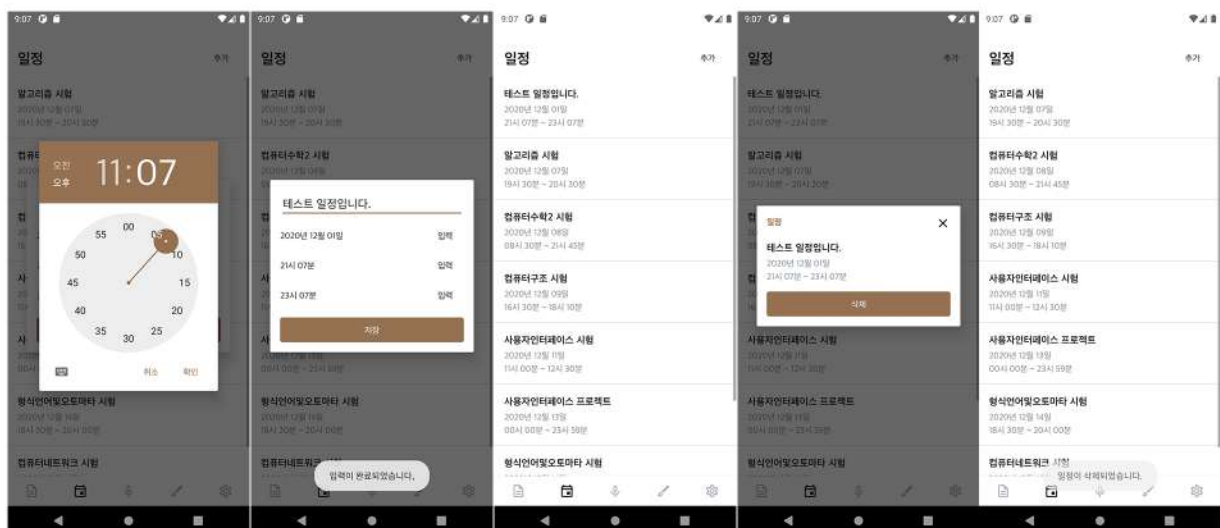


〈리스트〉

〈일정등록 화면〉

〈날짜입력화면〉

〈시작하는시간 입력화면〉



〈끝나는시간 입력화면〉

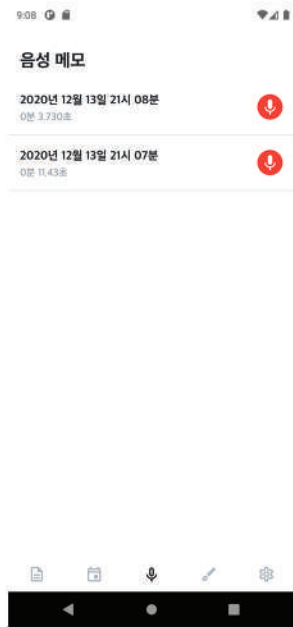
〈저장 시 토스트〉

〈저장 후 리스트화면〉

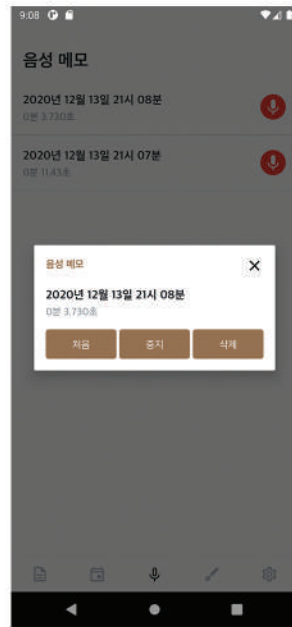
〈일정 상세〉

〈삭제 후 리스트화면〉

## 4. 음성 메모



〈리스트〉



〈음성메모 플레이백〉

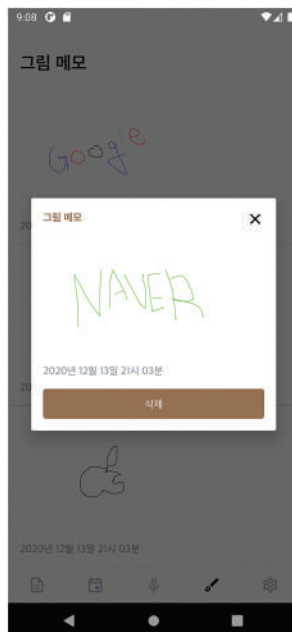


〈삭제 후 화면〉

## 5. 그림 메모



〈리스트〉



〈그림메모 상세〉



〈삭제 후 화면〉

## 5. 설정



〈설정 프래그먼트의 커스터마이징 체크박스〉

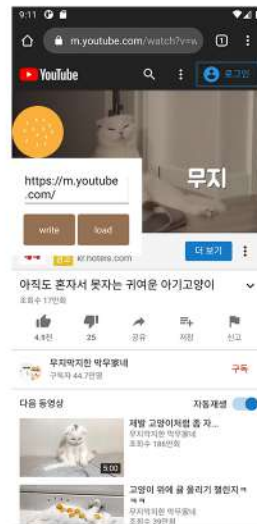
## 〈플로팅〉 – 유튜브와 동시 진행



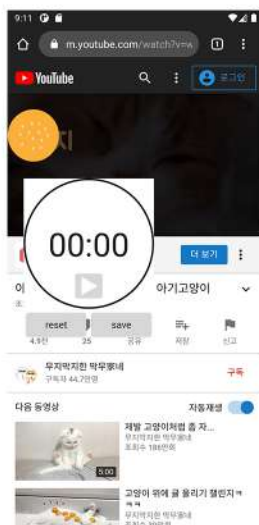
〈플로팅 기본 상태〉



〈펼쳤을 때 모습〉



〈메모의 클립보드〉



〈음성메모 화면〉



〈그림메모 화면〉



〈일정등록 화면〉

## [작업 과정 상세 설명]

[플로팅 버튼 – MainActivity]

플로팅 액션 버튼은 포그라운드 (바운드)서비스로 실행된다.

MainActivity가 실행되면 onCreate() 실행 후 onStart()가 실행될 때, 권한(Permission)을

검사한다. 오버레이, 파일 저장, 오디오 마이크 등

그리고 권한을 받으면 플로팅 버튼의 메인 서비스 컴포넌트가 MainActivity에 바운드 상태로 생명주기가 돌기 시작한다.

onCreate(): 권한 검사 및 바운드 서비스 실행, 각 메모 객체들 생성

onStart(): 플로팅 뷰 숨기기 함수(MainService의 hideView 함수)

onStop(): 플로팅 뷰 보여주기 함수(MainService의 ShowView 함수) —— 플로팅 버튼은 MainActivity가 stopped 상태가 될 때(화면이 가려질 때) 실행된다.

onDestroy(): MainService를 종료(unbind)

---

[MainService]: 메모 객체를 생성, 메모 객체 커스터마이징(객체를 직접 삭제하지 않음— show/hide )

onCreate(): FloatingViewManager 객체를 생성, LayoutParams 초기화(FloatingView의 LayoutParams)

onDestroy(): FloatingViewManager의 WindowManager의 올려진 View들을 remove

---

#### [FloatingView]

FloatingView는 FloatingViewManager를 통해 구현되며 WindowManager에 실질적으로 root View를 담당한다.(버거의 브레드 부분)

View를 상속받는 커스텀 뷰로 onTouchListener를 상속 받아 onTouch 할 때마다 raw x, y 좌표값을 기억하여 View를 움직일 수 있도록 정의한다.

그리고 FloatingView는 내부 interface를 선언하고 FloatingViewManager에서 구현하여 콜백 함수로서 실행한다.

Touch에 해당하는 것은 모두 onTouchListener가 하며,

FloatingView의 Click은 GestureDetector의 SimpleOnGestureListener가 이벤트를 감지한다.

---

#### [FloatingViewManager]

실질적으로 FloatingView(root view)를 띄우고 모든 메모 View들을 관리한다.

MainService는 해당 객체의 함수를 실행하며 플로팅 액션 버튼이 구동된다.

---

#### [Memo]



각 Memo클래스들은 onClickListener를 구현하여 View들마다의 이벤트가 실행된다.

---

#### [플로팅 버튼 – TextMemo]

"Memo를 상속받음"

문제점: EditText View를 사용하면 키보드가 작동할 것이라 예상했지만 Window에서 그려진 모습에서는 키보드는 강제로도 올릴 수 없다는 점을 알았고 커스텀 뷰로서 키보드를 구성하여 구현할 수 있다.(시간 관계상 구현 실패)

load 버튼은 클립보드매니저를 통해 클립보드 아이템을 가져와 데이터를 정해진 EditText에 세팅한다.

Write는 intent로 인앱의 WriteActivity를 실행시키고 putExtra로 플로팅 버튼에서 실행됨을 알려준다.(실행된 액티비티에서 비교 처리=인앱에서 띄운건지 플로팅 버튼에서 띄운건지)

---

#### [플로팅 버튼 – DrawMemo]

"Memo를 상속받음"

그림 메모는 DrawingView를 커스터마이징하여 onDraw의 update를 통해서 그림이 그려진다.

View의 Canvas는 Bitmap과 연결시켜 공유(share), 저장(save)를 즉각적으로 할 수 있다. 공유는 MediaStore에 저장과 동시에 Path를 받아 URI로 Parsing하고 extra와 함께 공유 intent로 실행된다.

저장은 내부 스토리지에 저장하고 DB에 등록한다.

어려웠던 점: 내부 스토리지에 저장하여 공유하면 콘텐츠 프로바이더를 구현하여 제공해야 했고 구현이 어려웠다.

---

[플로팅 버튼 – CalendarMemo]

"Memo를 상속받음"

마찬가지로 키보드를 띄울 수 없어 Write는 인앱으로 인텐트를 제공하여 구현하였다.  
캘린더 메모는 일정 패턴으로 나뉜 텍스트를 Load한다. (/ 로 나뉜 텍스트)

---

[플로팅 버튼 – RecoreMemo]

"Memo를 상속받음"

View의 이벤트를 구성하여 Stop, Start, Save, Reset 등의 함수를 실행한다.

모든 녹음은 MediaRecorder의 객체를 생성, 초기화 등을 진행하고 MediaRecorder의 release() 메소드로 파일을 저장한다.

그리고 저장할 때, DB에 등록하여 인앱에서 리스트로 조회 가능하도록 한다.

어려웠던 점: 파일 포맷마다 런타임 에러가 많이 발생하였고 많은 테스트와 Android API 문서를 통해 구현하였다.

---

## [어플 내부 – SettingsFragment] 설정 탭

checkBox view를 onClickListener로 이벤트를 감지하여 플로팅 버튼의 개수를 조절한다.  
플로팅 버튼이 구성되는 시점은 해당 fragment가 MainActivity의 생명주기에 따라서  
onStop 되었을 때,  
MainService의 customizeMemos() 함수를 호출하여 Memo 버튼의 Show, Hide(visible과  
gone)로 처리하여 플로팅 메모 버튼의 구성을 결정한다.  
그리고 그 설정을 계속 유지할 수 있도록 fragment의 onCreateView()가 호출할 때, 멤버  
변수로 갱신해준다.

---

## [어플 내부 – MainActivity에서 Fragment를 이용한 Tab 구성]

각 메모의 리스트는 하단 네비를 동일하게 공유하므로 별개의 액티비티를 생성하지  
않고 하나의 액티비티에서 프래그먼트를 이용해 각각 다른 화면을 구성하였다.

---

## [어플 내부 – Recycler View를 이용한 DB 가져오기]

Fragment가 onCreateView 될 때 DB에서 해당 탭의 table의 모든 column을 읽어들인다.  
읽어들인 정보를 메모 객체에 할당하여 메모 객체 배열을 만들고, 그 배열을 adapter에  
넘겨준다.  
adapter에서는 넘겨받은 메모 객체 배열을 이용해 Recycler View에 목록을 생성시킨다.

---

## [그림 메모와 음성 메모의 DB 이용 방법]

그림 메모와 음성 메모는 각 파일을 DB에 직접 저장하지 않고 내부저장소에 저장한 후  
파일 명을 문자열로 DB에 저장시킨다.  
DB를 읽어올 때는 각 파일 명을 이용해 내부 저장소에서 해당 그림 메모와 음성 메모를  
불러온다.

---

## [SQLite를 이용한 DB 연결에서 어려웠던 점]

아직 DB 수업을 듣지 않아 쿼리 문을 작성하기 위해 많은 것을 찾아보아야 했다. rawQuery와 execSQL은 무엇이 다른지, 행을 지울 때는 왜 delete 함수를 사용하는 것이 권장되는지를 알기 위해 많은 웹 사이트를 찾아보았다. 한국어 사이트는 접근하기 좋지만 정보의 깊이가 얇았고, 공식 문서와 스택 오버플로우의 도움을 많이 받았다.

---

#### [어플 내부 – TextMemo]

텍스트 메모는 모든 메모들의 가장 기본이 되는 포맷이기 때문에 가장 먼저 작업했다. 네 종류의 메모는 모두 텍스트 메모의 큰 틀을 이용하지만 각 메모의 특성에 맞게 필요한 기능을 추가하였다. 텍스트 메모의 목록 조회에서는 maxLine을 3으로 한정하여 하나의 메모가 지나치게 공간을 잡아먹는 일을 방지했다. 메모 하나를 터치하면 dialog 형식으로 새로운 액티비티가 나타난다. 이때 intent를 통해 해당 메모의 id, 내용, 작성 시각이 전달된다. 그 액티비티에서 해당 메모의 전체 내용을 볼 수 있다. 메모가 지나치게 길어질 경우를 대비해 메모 내용, 텍스트뷰에 스크롤을 가능하게 만들었다. 삭제 시 db에 해당 메모의 id를 식별자로 하여 삭제 요청을 보낸다.

---

#### [어플 내부 – Calendar]

일정 메모는 텍스트 메모의 확장 버전이다. 일정을 입력하기 위해 안드로이드의 기본 dialog를 이용해 달력과 시계가 나타나도록 하였다. dialog상에서 입력을 마치면 그 결과가 String으로 전달되고, String으로 DB에 저장된다. 그 외의 구현 방법은 텍스트 메모와 동일하다.

---

#### [어플 내부 – RecordMemo]

음성 메모는 파일을 DB에 직접 저장하지 않고 파일 명으로 저장하는 것이 특징이다. Recycler View에서 리스트로 보여줄 때는 개별 음성 메모를 가져오지 않는다. id, 파일 명, 저장된 시각, 파일의 길이만을 가져오며 리스트에서 음성 메모 하나를 터치하면 새 액티비티가 dialog 형태로 나타나며 그때 파일 명을 이용해 내부저장소에서 파일을 로드한다. 음성메모 재생을 위해 안드로이드의 기본 MediaPlayer를 사용했다. 특이한 부분으로는 pausePosition 이라는 변수를 사용해 해당 메모의 재생 위치를 기억해둔다. 제일 좌측의 ‘처음’ 버튼을 누르면 pausePosition을 0으로 설정한 후 처음부터 재생한다. 재생 도중 중지를 누르면 pausePosition이 해당 시점을 기록해두었다가, 재생 버튼을 다시 눌렀을 때 그 지점부터 재생하게 된다. 마찬가지로 삭제 버튼을 누르면 id를 이용해 해당 메모를 삭제한다.

---

### [어플 내부 – DrawingMemo]

그림 메모는 RecyclerView에서 리스트로 보여줄 때 부터 해당 이미지를 내부저장소에서 로드해온다. 해당 이미지 중 하나를 터치하면 intent로 해당 그림 메모의 id, 이미지 파일 이름, 기록된 시각을 넘겨주고 새 액티비티에서 또 한 번 이미지를 내부저장소에서 로드해온다.

---

### [UI 디자인]

안드로이드의 기본 디자인을 사용하지 않고 대부분의 요소를 커스텀하였다.

우선 colors.xml에서 어플 내부에서 사용할 컬러를 정의하였다. 이때 흰색, 검은색 이외에도 회색을 10단계로 나누어 지정해두었고, 어떤 상황에서 어떤 컬러가 사용될 것인지를 미리 정리하여 모든 화면에서 일관성 있게 사용될 수 있도록 노력하였다. 폰트 역시 Roboto 대신 애플SD고딕을 사용했다. 팀원 3명 모두 iOS 이용자이기 때문에 작업 시 친숙함을 느낄 수 있었다. 한편 Noto Sans를 사용하는 것도 애플과 안드로이드 OS의 일체감을 위해 고려해볼만한 사안인것 같다. 텍스트 사이즈는 24dp, 16dp, 14dp, 13dp 로 한정지었다.

모든 탭의 타이틀은 애플의 기본 타이틀 대신 TextView를 이용한 타이틀을 사용했다. 최근 몇 년간 카카오톡, 페이스북, 인스타그램 등에서도 채용되고 있는 좌측 상단의 큰 크기 bold체 타이틀이다.

margin, padding을 비롯한 모든 간격 요소는 안드로이드 디자인 가이드에 따라 가급적 8의 배수를 채택했다.

---

### [소스 디자인]

포토샵과 일러스트 그리고 무료 저작권의 소스들을 이용하여 애플에 있는 아이콘을 제작하고 서비스 상태에서의 버튼이 눈에 한번에 잘 들어올 수 있도록 색을 지정하여 디자인하였다.

버거라는 디자인에 맞게 원형 플로팅 버튼을 위에서 바라볼 수 있도록 하였고 이를 통해

---

### [예제와 양식의 통일 과정]

TextMemo를 리사이클러 뷰와 SQLiteDB를 사용하는 예제와 soundrecorder masters 라는 녹음 어플 예제를 가져와 다른 양식의 DB와 DB Helper, Adapter 등 통일을 시키면서 조금씩 API가 달라 에러가 나는 현상을 겪어 어려웠다. 이 과정에서 텍스트 메모 작업과 레코드 메모 작업 도중에 다른 팀원에게 맡겨 양식을 맞출 수 있는 부분을

타협하며 작업을 했다.

[제출물]

비교적 코딩 능력이 떨어져 코딩을 작성하다 다른 팀원에게 도움을 요청하여 기능을 마무리하고 제출물의 모든 디자인을 담당하였다. 모든 소스코드와 기능들을 전체적으로 확인하여 데모 시나리오와 보고서를 작성하였다.

## 4. 최종 작업 분해표

(● - 완성 / ★ - 교체 / ▲ - 부분완성 / X - 미완성)

기능	세부 작업	기여도	결과	기타사항
SYSTEM	1. 프래그먼트 & SQLite를 이용해 DB와 리사이클러뷰 연결 (Overall Layout)	1-1. Fragment	●	
		1-2. 리사이클러뷰		
		1-3. SQLite를 이용한 DB 연결		
	2. 플로팅 액션 버튼 (FAB)	2-1. 설계	●	플로팅 액션 버튼은 포그라운드 (바운드)서비스로 실행된다. MainActivity가 실행되면 onCreate() 실행 후 onStart() 가 실행될 때, 권한(Permission)을 검사한다. 그리고 권한을 받으면 플로팅 버튼의 메인 서비스 컴포넌트가 MainActivity에 바운드 상태로 생명주기가 돌기 시작한다.
		2-2. 기능 구현		
FUNCTION	3. 텍스트 메모 (Text Memo)	3-1. Recycler View	★	
		3-2. DB		
		3-3. 기능 구현		
	4. 그림 메모 (Record Memo)	4-1. Recycler View	●	
		4-2. DB		
		4-3. 기능 구현		
	5. 일정 (Calendar)	5-1. Recycler View	★	구글 캘린더와 연동하여 일정을 관리하고자 하였으나, 이 어플에서 등록한 일정은 이 어플 내에서 보이는 것이 보다 직관적일 것이라 판단하여 어플 내에 저장하였음.
		5-2. DB		
		5-3. 기능 구현		
		5-4. 구글 캘린더 연동		
	6. 음성 메모 (Recording Memo)	6-1. Recycler View	●	
		6-2. DB		
		6-3. 기능구현		
		6-4. PlayBack 구현		
	7. 파이어베이스	7-1. 백업 구현	X	어플 내부 저장소의 자료를 파이어베이스를 이용해 백업하고자 하였으나 잔여 시간에 비해 구현 난이도가 높았음. 네트워크 통신이나 사용자 인증과 같은 부분에 대한 이해가 부족하여 구현하지 못했음.
	8. 설정탭 (Option Tab)	8-1. 메모 탭 커스터마이징	●	
보완	9. 소스 & 디자인 (Source & Design)		●	햄버거를 연상시키는 버거 아이콘을 손수 제작하였음. 어플 내부 UI는 안드로이드의 기본 UI를 사용하지 않고 최근 많은 어플들에서 사용하는 UI를