Report

모바일 프로그래밍

WithDiary 개발 최종 보고서

사이버보안학과 201520932 윤지우

미디어학과 201721590 윤현민

<목차>

<그림 순서>

프로젝트 개요

주제 선정 배경

스마트폰의 사용이 보편화 되면서 다양하고 많은 어플리케이션이 시장에 출시되었다. SNS 및 일기장 또한 마찬가지로 다양한 어플리케이션이 존재했는데 실제로 사용해본 결과 혼자 또는 두 명이서 쓰는 커플 일기장이거나 불특정 다수에게 보이는(ex. 페이스북 등) 어플리케이션만 존재하였다.

SNS 서비스 중 하나인 인스타그램을 예시로 들면, 사용자가 글을 업로드한 경우, 불특정 다수에게 자신의 글이 공유되며, sns 계정을 비공개로 설정한 경우에도, 자신의 일부 친구들이 아닌, 모든 친구들에게 자신의 글이 공개가 된다. 카카오톡 같은 경우에는 자신과 친한 사람들과 그룹을 이룰 수 있지만(그룹 채팅), 일기 혹은 글을 기록하기 보다는 채팅 또는 잡담을 하는 목적으로 주로 사용된다.

따라서 커플 뿐만 아니라 여러 명의 친구들과 그룹을 이루어 함께 쓸 수 있는 그룹 일기장을 제작한다면 수요가 있을 것이라고 판단하여 With Diary 라는 어플리케이션을 개발하기로 결정하였다.

프로젝트 목표

함께 쓰는 일기장이라는 뜻을 갖은 With Diary 라는 어플리케이션 이름처럼, 그룹 기능을 갖은 일기장과, 일반적인 일기장 기능을 구현하는 것이 목표이다. 그룹 기능을 사용하기 위해서는 로그인 기능이 필요하기 때문에 일반적인 로그인 기능을 구현할 예정이다. 하나의 어플리케이션을 구현하는 것이기 때문에 세부적인 부분까지 신경 써 완성도를 높여 구현하는 것이 목표이다.

개발 과정

진행 스케줄

날짜	내용
4 월 말	프로젝트 주제 선정, 일기 작성, 파이어 베이스 데이터 입출력
5 월 초	파이어 베이스를 활용한 로그인 및 회원가입 구현
5 월 말	그룹 기능 추가
6 월 초	일기 삭제 기능, 버그 수정 및 UI 개선

팀원간 개발 업무 분담

팀원	개발 내용
윤지우	파이어 베이스를 이용한 텍스트 및 이미지 저장 일기 생성 및 삭제 기능 로그인 및 회원가입 기능 그룹 생성 및 탈퇴 기능
윤현민	파이어 베이스를 이용한 텍스트 저장 UI 전반 및(Recycler view, Custom Dialog, Toolbar 등) 기능 구현 WithDiary 디자인 사용자 계정 관리

적용 기술 설명(개발 코드, 기능 등)

개발 언어 및 도구

- 안드로이드 스튜디오 및 자바
- 모바일 프로그래밍 강의에서 안드로이드 스튜디오를 기반으로 수업을 진행하였고, 이는 안드로이드 어플리케이션 제작에 매우 적합하다고 생각되어 선택하게 되었다.
- 파이어 베이스

  파이어 베이스에서 Database, Storage, Auth, Analytics 를 제공해주기 때문에, 서버 인프라를 고민할 필요가 없었다. 어플리케이션을 구현함에 있어 서버를 직접 만드는 수고를 덜어주고 손쉽게 코드를 통해 접근할 수 있기 때문에 파이어 베이스를 이용하게 되었다.
- 깃 허브

  Repository 의 완전한 복사본을 로컬 장비에 저장할 수 있으며, 처리 속도가 빠르고 작업 이력 관리가 가능한 깃 허브를 이용해 협업을 진행하였다.
  깃 허브 링크: https://github.com/pika96/WithDiary

개발 코드 및 기능

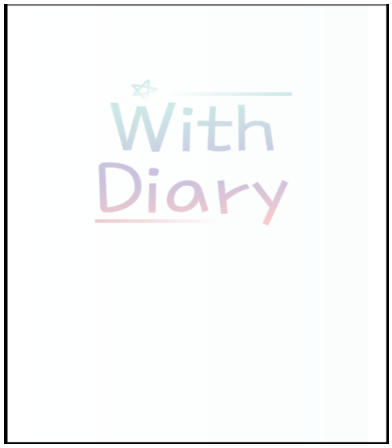
어플 아이콘

WithDiary 테마에 맞는 아이콘을 따로 만들어 제작하였다.

<그림 1> WithDiary 아이콘

인트로

처음 어플리케이션 실행 시 WithDiary 라는 화면이 띄워진다.

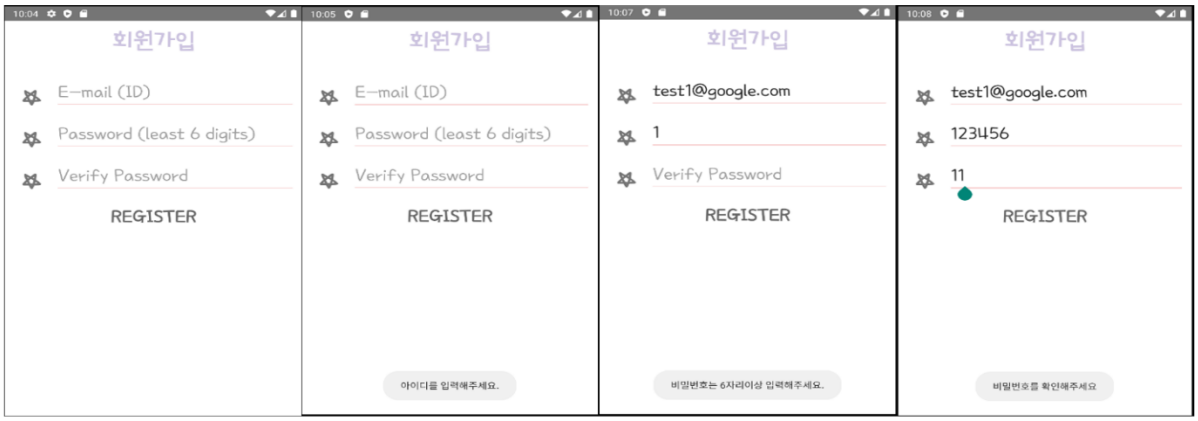


<그림 2> Intro 화면

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate( savedInstanceState );
    setContentView( R.layout.activity_intro );
    Handler handler = new Handler();
    handler.postDelayed(() → {
            Intent intent = new Intent(getApplicationContext(), Login.class);
            startActivity(intent);
            finish();
    }, delayMillis: 3000);
}
```

<그림 3> Intro 코드

회원가입



<그림 4> 회원 가입 예외 처리

```
firebaseAuth.createUserWithEmailAndPassword(Regist_ID, Regist_PW)
        .addOnCompleteListener( activity: this, (task) → {

            Intent send_intent = new Intent();
            if(task.isSuccessful()){
                Toast.makeText( context: DB.this, text: "회원가입 완료", Toast.LENGTH_SHORT).show();
                setResult(Return_OK, send_intent);
                finish();
            }else{
                Toast.makeText( context: DB.this, text: "회원가입 실패", Toast.LENGTH_SHORT).show();
                setResult(Return_fail, send_intent);
                finish();
            }
        });
```

<그림 5> 회원 가입 코드

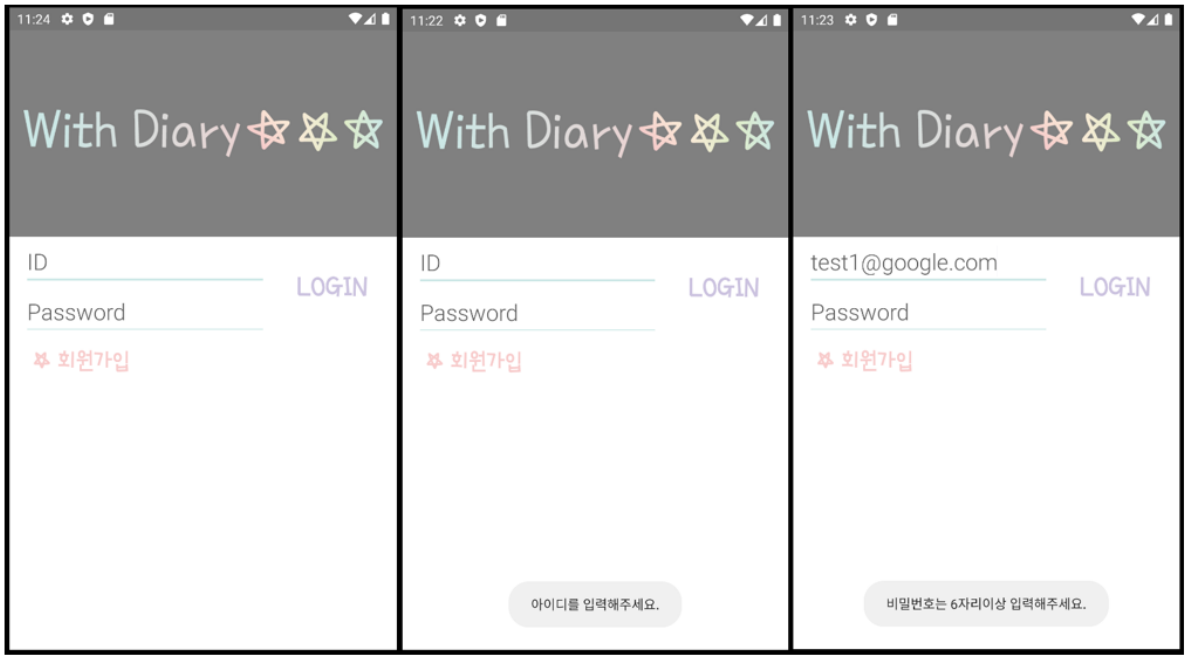
아이디가 입력되지 않았을 때, 비밀번호 6 자리 이상일 때, 비밀번호가 일치하지 않을 때, 이미 있는 아이디일 때 등 예외처리를 해주었다.

firebase 에서 지원하는 createUserWithEmailAndPasswrod 메소드를 통해 ID 와 PW 를 입력하여 회원가입을 구현하였다.

<그림 6> firebase Auth 사용자 계정 관리

회원가입을 완료하게 되면 firebase Auth 에 저장된다. 이후 저장된 계정은 로그인, 회원탈퇴 등 사용자 계정을 관리할 때 사용하게 된다.

로그인



<그림 7> 로그인 예외 처리

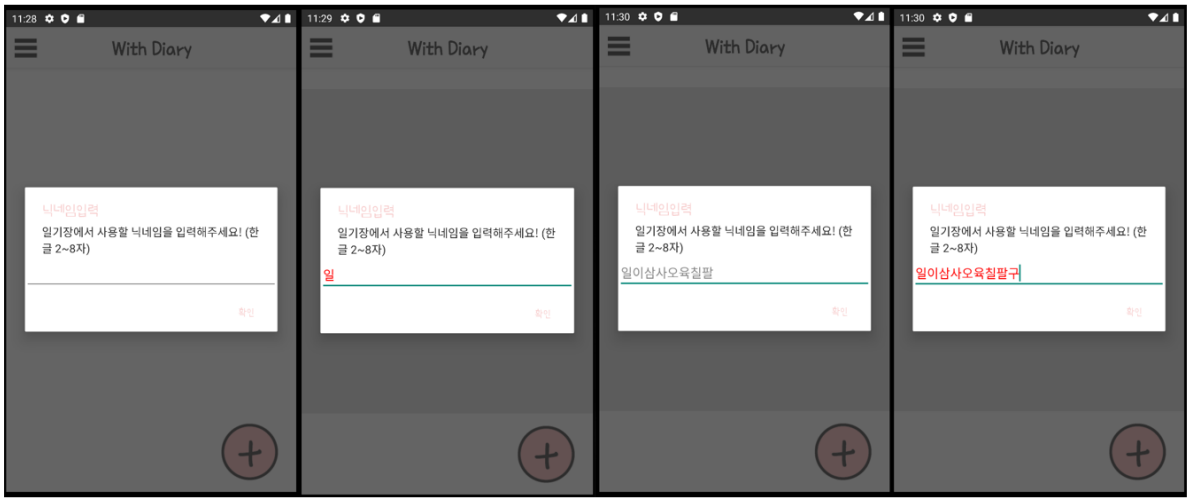
```
firebaseAuth.signInWithEmailAndPassword(ID, PW).addOnCompleteListener( activity: this, (task) → {
        if(task.isSuccessful()){
            Toast.makeText( context: Login.this,  text: "로그인 성공", Toast.LENGTH_SHORT).show();
            Intent intent = new Intent( packageContext: Login.this, Make_diary_Activity.class );
            startActivity(intent);
            finish();
        }else{
            Toast.makeText( context: Login.this,  text: "로그인 실패", Toast.LENGTH_SHORT).show();
        }
});
```

<그림 8> 로그인 코드

로그인 기능 또한 회원가입과 마찬가지로 아이디와 비밀번호에 대한 예외 처리를 해주었다.

Firebase 에서 제공하는 signInWithEmailAndPassword 메소드를 통해 입력한 ID 와 PW 로 로그인기능을 구현하였다.

닉네임입력



<그림 9> 닉네임 입력 예외 처리

닉네임은 2~8 글자 사이로 지을 수 있으며 그 이외의 글자 수는 빨간색 글씨로 표시되며 확인 버튼이 활성화되지 않는다.

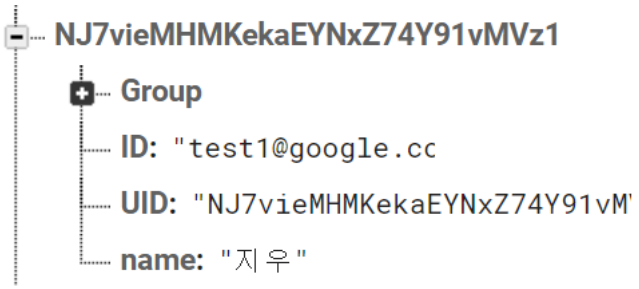
```
if (TextUtils.isEmpty( firebaseUser.getDisplayName() )) {

    final EditText InputName = new EditText(  context: Make_diary_Activity.this );
    FrameLayout.LayoutParams params = new FrameLayout.LayoutParams( ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.WRAP_CONTENT );
    params.leftMargin = 25dp;
    params.rightMargin = 25dp;
    AlertDialog.Builder getNameDialog = new AlertDialog.Builder( context, R.style.MyAlertDialogStyle );
    getNameDialog.setTitle( "닉네임입력" ).setMessage( "일기장에서 사용할 닉네임을 입력해주세요! (한글 2~8자)" );
    InputName.setLayoutParams( params );
    InputName.setSingleLine( true );
    getNameDialog.setView( InputName );
    getNameDialog.setCancelable( false );


    InputName.addTextChangedListener( new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {

        }

        @Override

        public void onTextChanged(CharSequence s, int start, int before, int count) {

            if (InputName.getText().toString().length() >= 2 && InputName.getText().toString().length() <= 8) {
                button.setEnabled(true);
                InputName.setTextColor( Color.GRAY);
            } else {
                InputName.setTextColor(Color.RED );
                button.setEnabled(false);
            }
        }
```

<그림 10> 닉네임 입력 코드

확인 버튼 비활성화와 색깔을 넣어 주기 위해 커스텀 다이얼로그를 사용하였다. 코드에서 확인할 수 있듯이 조건이 맞지 않으면 setEnabled 을 사용하여 확인버튼을 비활성화하고 setTextColor 를 사용하여 GRAY 와 RED 로 글자 색깔을 바꿔주는 것을 확인할 수 있다.

```
databaseReference = firebaseDatabase.getReference();
Map<String, Object> childUpdates = new HashMap<>();
Map<String, Object> UserValues = null;
User user = new User(ID, Name, cur_UID);
UserValues = user.toMap();

childUpdates.put("/User/" + cur_UID, UserValues);
databaseReference.updateChildren(childUpdates);
```

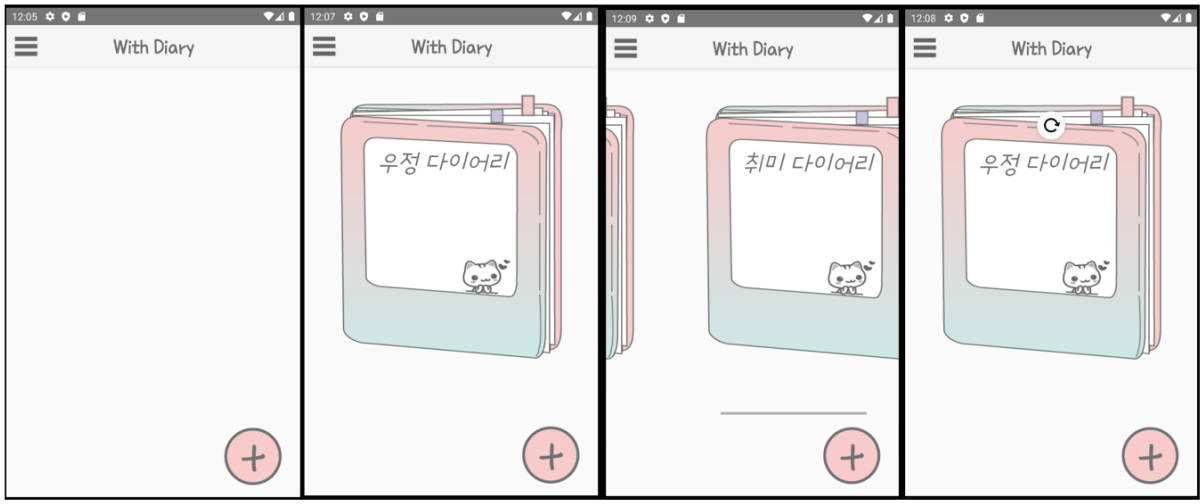
<그림 11> 닉네임 DB 저장



<그림 12> DB 에 저장된 사용자 정보

닉네임을 입력하고 확인 버튼을 누르게 되면 사용자 정보들이 DB 에 저장된다. User 클래스를 따로 만들어 비밀번호를 제외한 아이디, 닉네임, UID 등 사용자 정보를 저장해주는 코드를 확인할 수 있다. 실제로 DB 에 저장된 사용자 정보를 확인할 수 있다.

일기장 선택 화면



<그림 13> 일기장 선택 화면

그림 8 은 순서대로 초기 상태, 일기장 한 개, 일기장 두 개 이상, 새로고침 기능을 나타낸다. 처음 회원가입을 하고 아무런 그룹에 속해 있지 않는 상태에서는 빈칸으로 나타난다. 일기장을 하나 이상 추가했을 경우 옆으로 스와이프하며 일기장을 고를 수 있다. 새로고침 기능은 일기장 목록이 추가 또는 삭제되거나 변경되었을 경우 업데이트된 목록을 불러올 수 있다.

```
listview = findViewById( R.id.make_diary_listView );
MyListDecoration decoration = new MyListDecoration();
LinearLayoutManager layoutManager = new LinearLayoutManager( context: this, LinearLayoutManager.HORIZONTAL, reverseLayout: false );
listview.setLayoutManager( layoutManager );
adapter = new MyAdapter( context: this, grouplist);

listview.setAdapter( adapter );
listview.addItemDecoration( decoration );


adapter.setOnItemClickListener((v, position) -> {
```

<그림 14> 일기장 선택 화면 RecyclerView 코드

Static Adapter 클래스에 RecyclerView 를 상속하여 구현하였다. adapter 클래스에 grouplist 를 넣어주어 그룹 정보를 표시하고 LinearLayoutManager.HORIZONTAL 을 적용하여 가로로 스와이프 할 수 있도록 구현하였다. onItemClick 을 통해 일기장 아이템

클릭 시 이벤트가 발생하게 하였으며, ViewHolder 에 리스너를 달아 아이템 각각의 포지션을 받아올 수 있게 하였다.

```
databaseReference = firebaseDatabase.getReference(DBPath);
databaseReference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

        group_list.clear();
        for(DataSnapshot snapshot : dataSnapshot.getChildren()){
            String group = snapshot.getValue(String.class);
            String key = snapshot.getKey();

            group_list.add(group);
            group_key.add(key);
        }
```

<그림 15> 그룹 리스트 가져오는 코드

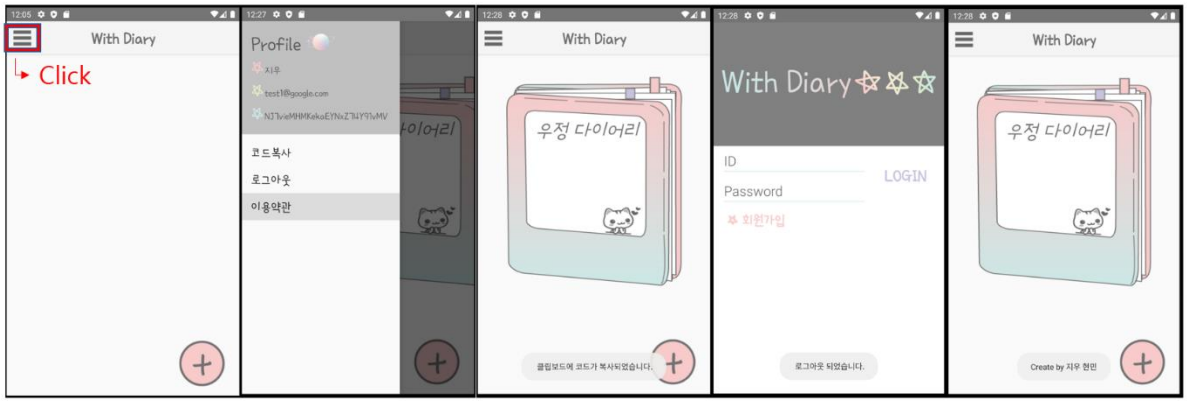
일기장(그룹)을 생성할 때 그룹 정보는 DB 에 저장한다. 사용자는 일기장을 여러 개 가질 수 있으므로 각 그룹이름을 ArrayList<String> group_list 에 저장하여 가져온다. firebase 에서 지원하는 addValueEventListener 를 활용하여 DB 에 저장 되어있는 정보를 가져올 수 있다.

```
    swipeRefreshLayout=findViewById( R.id.swipe2 );
    swipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {

        @Override
        public void onRefresh() {
            getGrouplist(UID);
            swipeRefreshLayout.setRefreshing(false);
        }
    });
}
```

<그림 16> 일기장 새로고침 코드

setOnRefreshListener 를 활용하여 아래로 스크롤하면 새로고침 되어 그룹 정보를 새로 불러올 수 있게 구현하였다. getGrouplist()는 그룹 정보를 가져와 화면에 출력해주는 메소드이다.

네비게이션 드로어

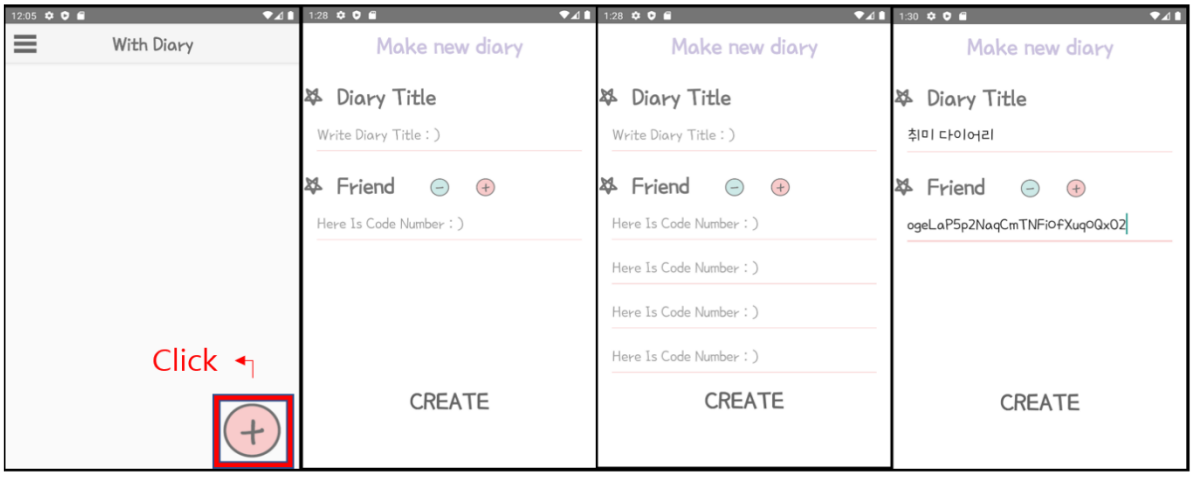


<그림 17> 네비게이션 드로어 기능

일기장 선택 화면에서 툴바 왼쪽 상단의 버튼을 클릭하면, 두 번째 사진처럼 화면이 나타난다. 사용자의 프로필을 보여주는데 위에서부터 닉네임, 이메일, 초대코드 순서로 보여준다.

네비게이션 드로어에는 코드 복사, 로그아웃, 이용약관 이렇게 3 가지의 메뉴가 있다. 코드복사 메뉴를 클릭하면 바로 위에 보이는 초대코드를 클립보드에 복사할 수 있다. 이는 다른 사람에게 초대코드를 보낼 때 유용하게 쓰일 수 있다. 로그아웃은 현재 로그인한 계정을 로그아웃 할 수 있는 메뉴이다. 이용약관 메뉴 클릭을 통해 어플리케이션을 만든 사람을 확인할 수 있다.

일기장 생성



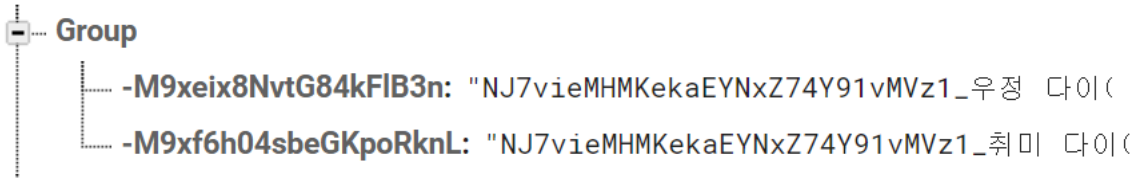
<그림 18> 일기장(그룹) 생성 기능

오른쪽 하단의 + 버튼을 클릭하게 되면 일기장을 새로 생성할 수 있는 화면이 나온다. 사용자는 Diary Title 에서 일기장 제목을 입력할 수 있고, Friend 에서 같이 일기장을 사용할 친구의 초대코드를 입력할 수 있다. 맨 오른쪽에 있는 사진이 예시이다. Friend 옆에 있는 – 와 + 버튼을 눌러 최대 4 명의 친구를 추가할 수 있다.

```
String Diary_Title = get_intent.getExtras().getString( key: "Diary_Title");
UID_list = get_intent.getStringArrayListExtra( name: "UID");

for(int i = 0; i < UID_list.size(); i++) {

    String UID = UID_list.get(i);

    databaseReference = firebaseDatabase.getReference();
    databaseReference = databaseReference.child("/User/" + UID);

    databaseReference.child("Group").push().setValue(Diary_Title);

}
finish();
```

<그림 19> 그룹 생성 코드

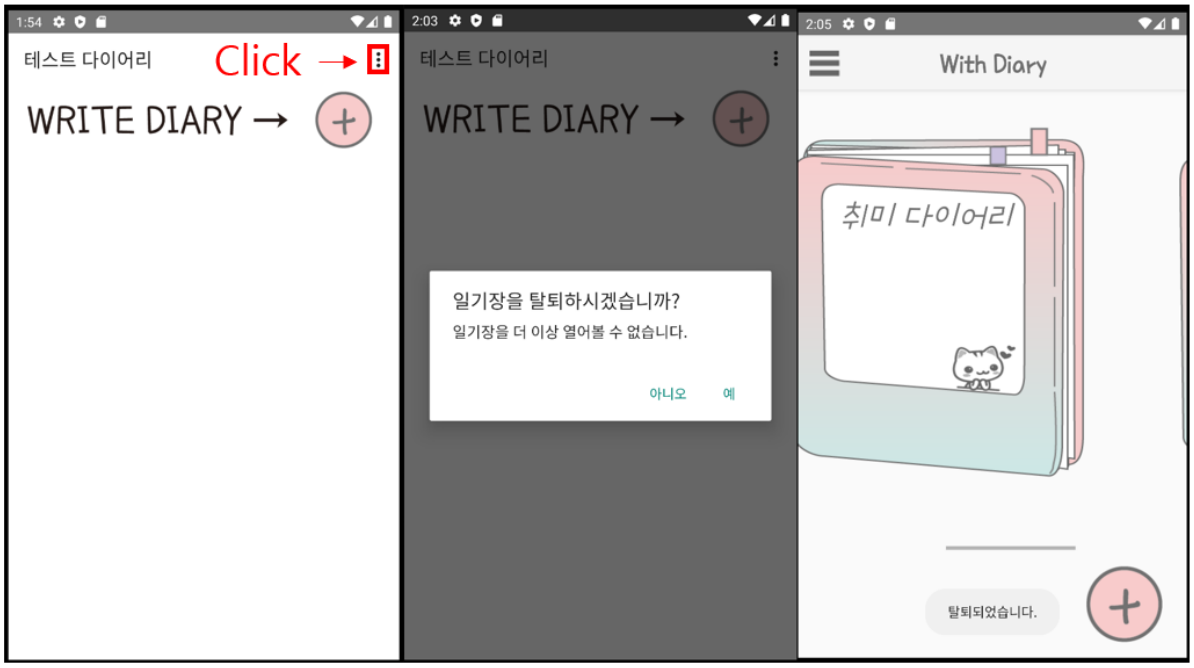
그룹 정보는 firebase DB 에 저장이 된다. 각 사용자마다 그룹 정보를 가지고 있어야함으로 추가된 사용자마다 각각 그룹 정보를 넣어준다.



<그림 20> 그룹 정보가 저장된 DB

Firebase 를 보면 각 사용자마다 그룹 정보가 저장된 것을 확인할 수 있다.

일기장 탈퇴(삭제)



<그림 21> 일기장 탈퇴

일기장에 들어가면 위에 툴바를 통해 일기장을 탈퇴할 수 있다. 일기장을 탈퇴하겠냐는 다이얼 로그가 뜨고 예 버튼을 누르면 탈퇴가 완료된다.

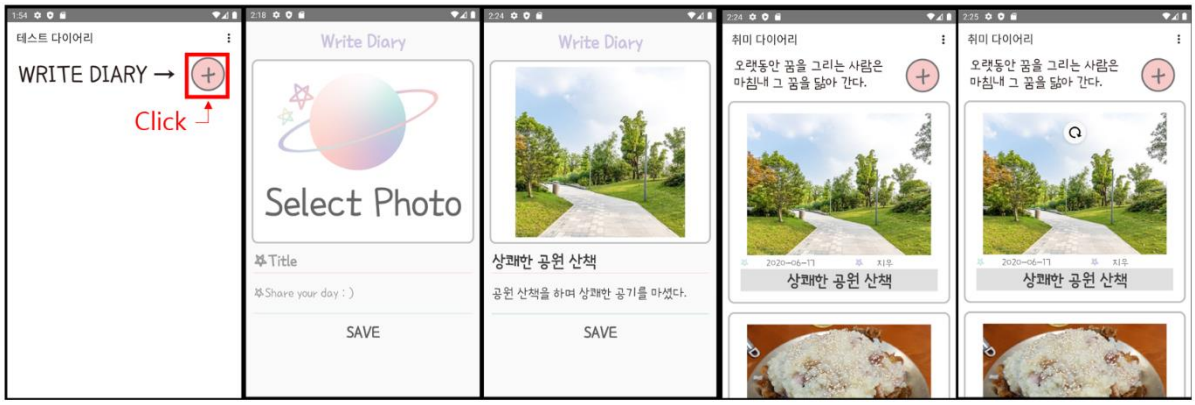
```
DBPath = "User/" + curUID + "/Group/" + groupkey + "/";

databaseReference = firebaseDatabase.getReference(DBPath);
databaseReference.removeValue();
finish();
break;
```

<그림 22> 일기장 탈퇴 코드

<그림 23> 일기장 정보 firebase 삭제

Firebase 를 통해 해당 일기장 정보가 있는 경로를 찾아가 removeValue 를 이용하여 DB 에 정보를 없앤다.

일기 생성



<그림 24> 일기 생성 기능

오른쪽 상단 + 버튼을 누르면 일기 생성 화면으로 넘어간다. 사진과 일기 제목, 일기 내용을 입력한 후 SAVE 버튼을 누르면, 보이는 것처럼 일기장에 일기가 업로드 된다. 일기장에는 날짜, 작성자 제목 이미지가 표시된다. 또한 일기를 추가, 삭제 또는 변경할 때 아래로 스와이프하여 새로고침을 통해 업데이트 할 수 있다.

```
datalist tmp_datalist = new datalist(writeUser, Date, Title, Diary, Imagepath, firebaseUser.getUid());

DBPath = "Group/" + cur_groupname + "/";
databaseReference = firebaseDatabase.getReference(DBPath);
databaseReference.push().setValue(tmp_datalist);
uploadFile(cur_groupname, Date, Title);
```

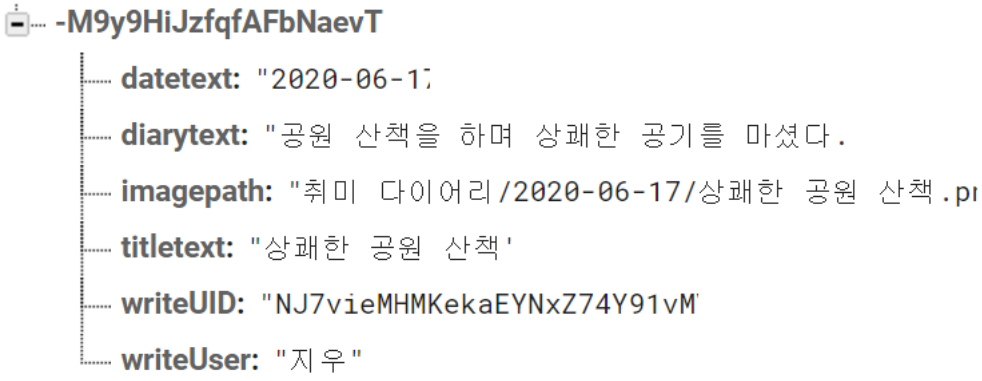
<그림 25> 일기 생성 코드

```
StorageReference storageReference = firebaseStorage.getReference().child(savePath + filename);


storageReference.putFile(filePath) UploadTask
        .addOnSuccessListener((OnSuccessListener) (taskSnapshot) → {
                progressDialog.dismiss();

                Toast.makeText(getApplicationContext(), text: "업로드 완료!", Toast.LENGTH_SHORT).show();
        }) StorageTask<UploadTask.TaskSnapshot>

        .addOnFailureListener((e) → {
                progressDialog.dismiss();
                Toast.makeText(getApplicationContext(), text: "업로드 실패!", Toast.LENGTH_SHORT).show();
        }) StorageTask<UploadTask.TaskSnapshot>

        .addOnProgressListener((OnProgressListener) (taskSnapshot) → {
                double progress = (100 * taskSnapshot.getBytesTransferred()) / taskSnapshot.getTotalByteCount();
                progressDialog.setMessage("Uploaded " + ((int) progress) + "% ...");
        });
```

<그림 26> 이미지 업로드 코드

일기 내용을 저장할 수 있는 Datalist 클래스를 만들었다. Datalist 클래스는 일기 작성자, 일기 내용, 일기 작성시간, 이미지 경로, 사용자 UID 를 저장한다. 해당 정보들은 firebaseDatabase 에서 지원하는 setValue 를 통해 DB 에 저장된다.

이미지는 Storage 에 저장이 되는데 firebaseStorage 에서 제공하는 putFile 을 통해 경로를 입력 받아 해당 이미지를 저장한다.



<그림 27> 파이어 베이스 DB 일기 정보

<그림 28> 파이어 베이스 Storage 저장 정보

일기 선택(보기)

해당 일기를 선택하면 그림과 같이 이미지, 제목, 날짜, 일기 내용이 표시된다.

일기의 내용이 길 경우 스크롤을 해서 일기의 내용을 볼 수 있으며, 외부 라이브러리를 사용하여 오버스크롤이 된 경우를 부드럽게 처리해주었다. (setUpOverScroll)

<그림 29> 일기 선택 화면

```
storageReference.getDownloadUrl().addOnCompleteListener((task) → {
        if(task.isSuccessful()){
            Glide.with( activity: Select_Diary.this)
                    .load(task.getResult())
                    .into(Read_imageview);
        }else{
            Toast.makeText( context: Select_Diary.this, task.getException().getMessage(), Toast.LENGTH_SHORT).show();
        }
});
```

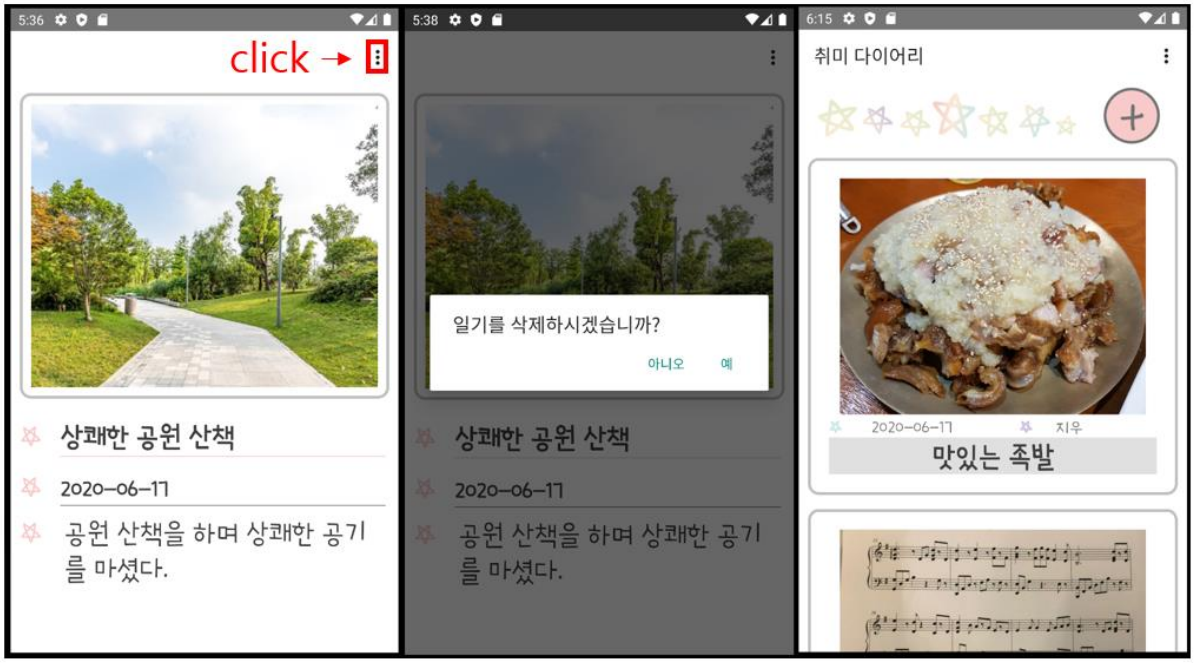
<그림 30> 이미지 출력 코드

```
databaseReference = firebaseDatabase.getReference(DBPath);
databaseReference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        data_list.clear();

        for(DataSnapshot snapshot : dataSnapshot.getChildren()){

            datalist datalist = snapshot.getValue(datalist.class);
            String key = snapshot.getKey();

            data_list.add(datalist);
            diary_key.add(key);
        }
```

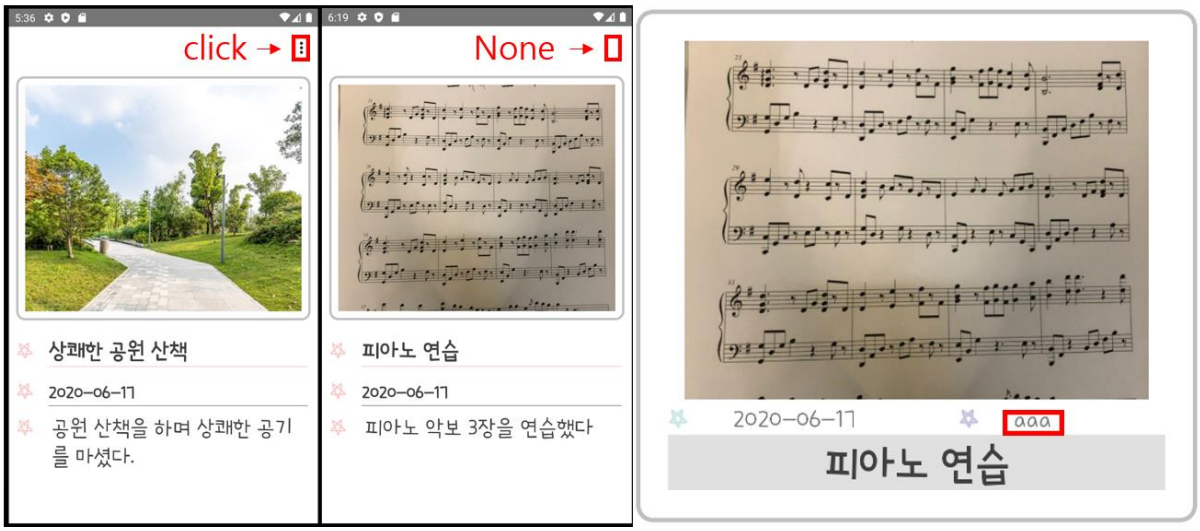
<그림 31> 일기 내용 코드

일기 내용은 firebase 가 지원하는 addValueEventListener 메소드를 통해 가져온다. 가져온 데이터는 datalist 클래스를 새로 만들어 저장한다. 해당 데이터는 textview 에 넣어준다. 이미지는 getDownloadUrl 메소드로 가져와 Glide 를 통해 imageView 에 넣어준다.

일기 삭제



<그림 32> 일기 삭제 기능



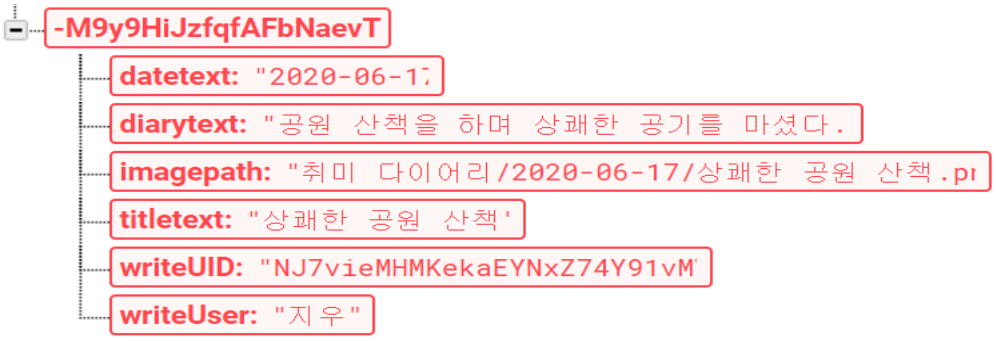
<그림 33> 일기 작성자가 아닌 경우

일기 선택을 한 뒤 상단 툴바를 이용하여 삭제를 할 수 있다. 그룹에서 작성한 일기에는 오른쪽 그림과 같이 작성자가 표시되는데 이 때 현재 사용자와 일기 작성자가 다르면 툴바를 보이지 않게 하여, 일기를 작성한 사람 만이 일기를 삭제할 수 있게 하였다.

```
DBPath = "Group/" + curGroup + "/" + key + "/";

databaseReference = firebaseDatabase.getReference(DBPath);
databaseReference.removeValue();
finish();
break;
```

<그림 34> 일기 삭제 코드

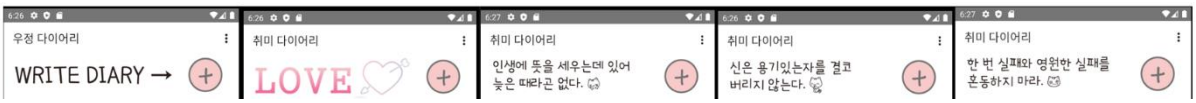
데이터 베이스에서 저장 되어있는 일기 정보 경로를 받아와 removeValue()메소드로 삭제한다.



<그림 35> 파이어 베이스 일기 삭제

일기 삭제를 하게 되면 파이어 베이스의 DB 에 저장 되어있는 해당 일기 정보 또한 삭제된다.

일기 랜덤 문구 출력



<그림 36> 일기 랜덤 문구

```
    if (recyclerAdapter.getItemCount() == 0) {

        emptyView.setVisibility(View.VISIBLE);
        notemptyView.setVisibility( View.GONE );
    }
    else {
        Random random = new Random(  );
        int num = random.nextInt(image.length);
        notemptyView.setImageResource( image[num] );
        emptyView.setVisibility(View.GONE);
        notemptyView.setVisibility( View.VISIBLE );
    }
    recyclerAdapter.notifyDataSetChanged();
}
```

<그림 37> 일기 랜덤 문구 코드

일기장을 클릭하면 상단에 랜덤으로 문구가 출력된다. 일기가 하나도 존재하지 않을 때는 첫 번째 그림에 보이듯이 WRITE DIARY 문구를 출력해준다.

코드에서 확인할 수 있듯이 랜덤으로 숫자를 받아 해당 문구를 출력해주고 있다.

기술적 문제점 해결 경험

구현 계획: 일기 화면에서 일기가 없을 때 이미지를 따로 띄우기

해결 경험: 일기 화면의 레이아웃은 Recycleview 를 사용하였는데 일기가 없을 때, 해당 레이아웃에 이미지를 넣고자 했다. 처음 접근했던 방법은 Recycleview 를 GONE 상태로 만들어 아예 보이지 않게 한 뒤 이미지를 띄우는 방식이었다. 하지만 새로고침이 가능하도록 Recycleview 안에 스와이프 기능을 넣어 놓은 상태였고 Recycleview 상태가 GONE 이 되면, 새로고침이 불가능하게 되었다. 해당 해결 방법을 구글링 했지만 따로 방법이 나오지 않았다. 따라서 Recycleview 의 상태를 Visible 하게 두고, 다른 UI 를 이용하여 일기가 없을 때 이미지를 넣어 주게 되었다.

구현 계획: 로그인 구현 후 해당 사용자 정보 받아오기

해결 경험: Firebase 에서 Auth 를 통해 로그인 방식을 구현하였다. 하지만 Firebase Auth 의 로그인 방식은 비동기식이기 때문에 가끔 로그인 후 사용자 정보가 null 인 경우가 발생하였다. 따라서 전체적인 로직을 로그인이 완전히 끝나길 기다린 후 완료가 된 다음 사용자 정보를 받아오도록 수정하였다.

구현 계획: Firebase 초기 설정 및 안드로이드 연동

해결 경험: Firebase 는 다양한 기능을 지원해주지만 초기 설정하는 부분이 여러가지 많았다. Firebase 를 처음 사용해 보았기 때문에 조금 더 헤맸다. Firebase 에서 키를 받아오고 안드로이드와 연동하는 부분이 있었는데 Firebase 의 공식 문서와 구글링을 통해 해결하였다.

구현 계획: Firebase Storage 에서 이미지 받아와 imageview 에 올리기

해결 경험: Firebase 를 이용하면서 DB 와 로그인 정보는 텍스트이기 때문에 조금 수월하였지만 이미지 정보를 받아오는 부분이 어려웠다. URL 로 받거나 이미지 자체를 받는 여러 방법이 있었는데 Firebase 공식 문서를 통해 Glide 를 이용하여 imageview 에 사진을 올리는 방법을 알게 되었다. 핸드폰에 직접 이미지를 저장하는 것이 아닌 캐시를 받아오는 방법이기 때문에 우리 프로젝트에 매우 적합했다. 하지만 해당 Glide 를 사용하기 위해 Build.gradle 을 수정하는 등 여러 가지 세팅이 필요하였다. 모두 처음해보는 것이라 매우 헷갈렸던 것 같다.

향후 추가 개발 방향 또는 계획

회원 탈퇴

사용자 회원을 탈퇴하는 기능을 구현하고 싶다. 사용자가 탈퇴하게 될 경우 이전에 작성한 일기장과 일기를 삭제할지를 아직 명확하게 정하지 않았기 때문에 구현에 있어서 순위를 맨 뒤로 두었다. 나중에 이런 부분을 명확하게 정하고 구현하고 싶다.

일기를 외부로 공유 및 저장

작성한 일기를 다른 외부 어플리케이션(ex. 카카오톡, facebook, instargram 등)으로 공유하는 기능을 추가할 예정이다. 또한 작성한 일기를 일정한 형식(ex. PDF, JPG 등)으로 저장할 수 있도록 구현할 예정이다.

결론

두 명이상이 같이 쓸 수 있는 WithDiary 어플리케이션을 만들어 보았다. 구현을 하면서 매우 간단하다고 생각했던 기능을 구현할 때 생각지도 못한 곳에서 에러가 나 오래 걸렸던 적이 있다. 또한 평소에 어플리케이션을 사용하면서 간단한 기능조차 하나하나 모두 개발자가 구현을 한 것이구나 라고 느꼈다. 구현하면서 여러 어려움이 있었지만 완성을 하고 보니 많은 성취감을 얻었다. 개발적인 측면에서는 Git 을 이용하여 협업을 해보았던 경험이 가장 뜻 깊었던 것 같다. 또한 하나의 어플리케이션을 처음부터 끝까지

완성도 있게 만든 것 또한 매우 좋은 경험이 되었다. 아직 100% 완벽하진 않지만 기회가 된다면 실제로 마켓에 올리는 과정까지 해보고 싶다.