

모바일 프로그래밍 (나의 앱 만들기)

2020. 7.6

학과	컴퓨터공학과
학번	201433931 / 201533887
성명	팀장 : 황승환 / 팀원 : 표승훈



목 차

1. 나의 앱 만들기.....	3
1.1 애플리케이션 개요	3
1.2 애플리케이션 구성	4
1.3 애플리케이션 실행화면	5
1.4 데이터베이스 설계	11
1.5 애플리케이션 구현	14
2. 나의 앱 만들기 후기.....	41

1. 나의 앱 만들기

1.1 애플리케이션 개요

1) App 이름

『왔어독』

2) App 개요

반려견 100만시대에 채팅 어플에 수요도 급증하고 있는 현재 어플리케이션 트렌드에 맞추어

저희는 반려견 SNS 채팅 어플이라는 주제로 반려견 정보 공유 플랫폼을 만들어 보았습니다.

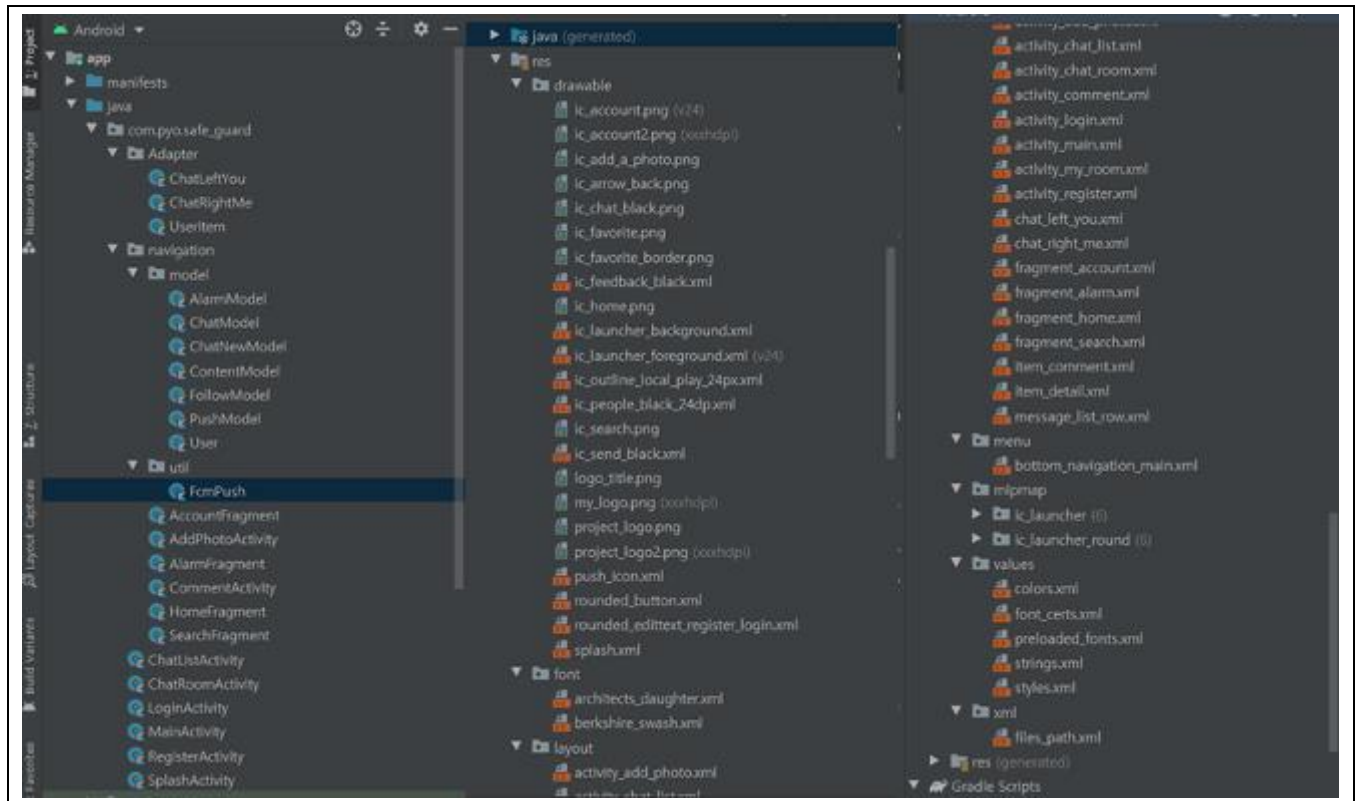
먼저 회원가입기능을 하고 다른 SNS 서비스와 비슷하게 자신의 반려견 사진과 글을 피드에 올릴 수 있도록 하였습니다. 게시글을 조회하고 등록, 프로필 이미지 수정, 사진들을 모아볼 수 있는 환경도 구축하였습니다. 또한 팔로우, 좋아요 기능과 피드에 대한 댓글 기능도 넣었습니다. 이러한 다른 사용자들의 반응을 알 수 있도록 알림 기능을 넣었습니다. 마지막으로 SNS 가입 회원들간의 소통을 위하여 회원들과의 1대1 채팅 기능을 넣었습니다. 이 모든 것을 firebase DB를 연결하여 정보들을 저장하고 관리할 수 있도록 하였습니다.

□ 주요기능

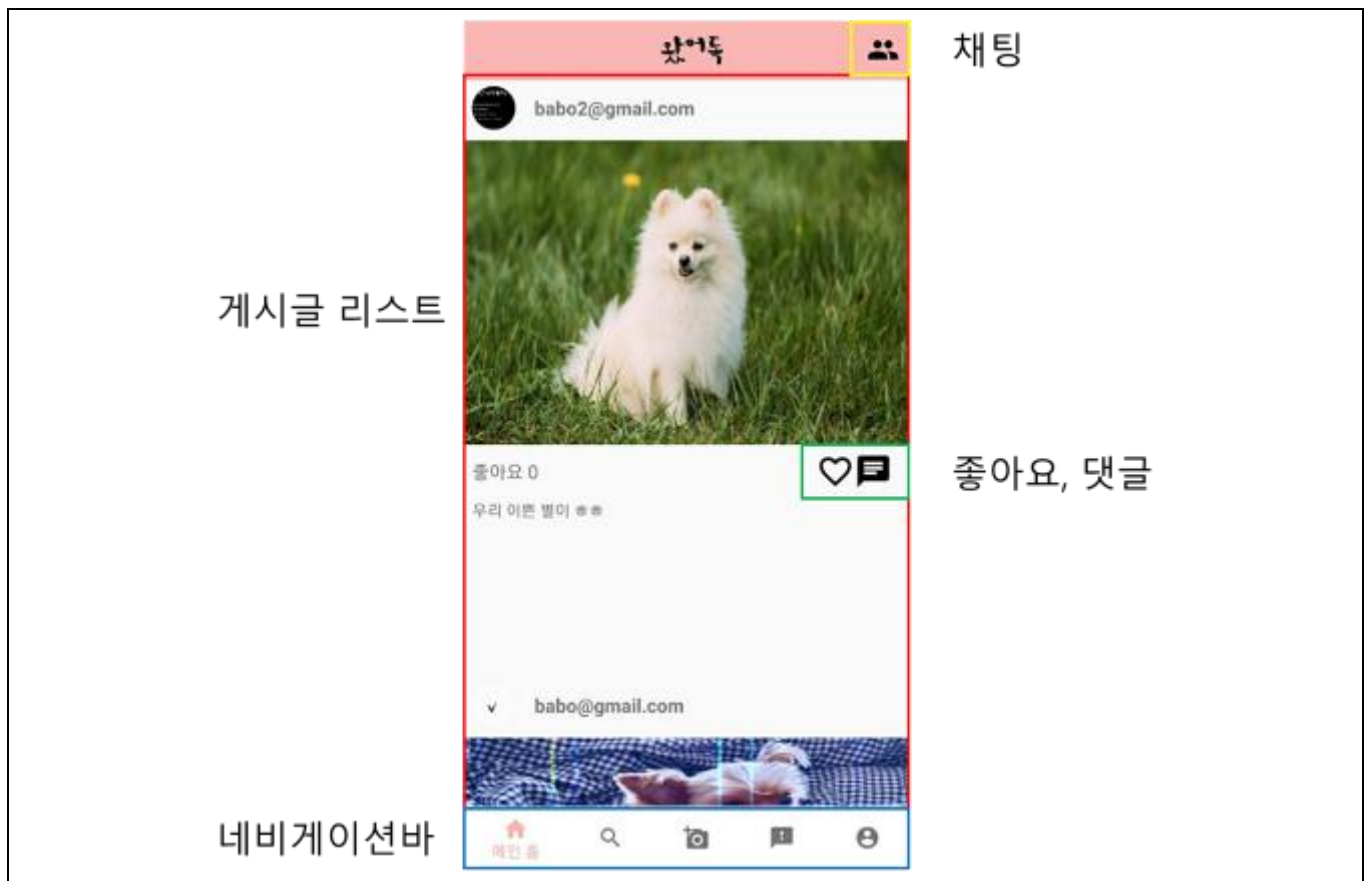


1.2 애플리케이션 구성

1) 프로젝트 구성도(프로젝트 구성도를 개발 툴에서 캡처해서 넣을 것)





2) 메인 화면 구성도(메인화면을 캡처해서 넣고, 간단히 설명)





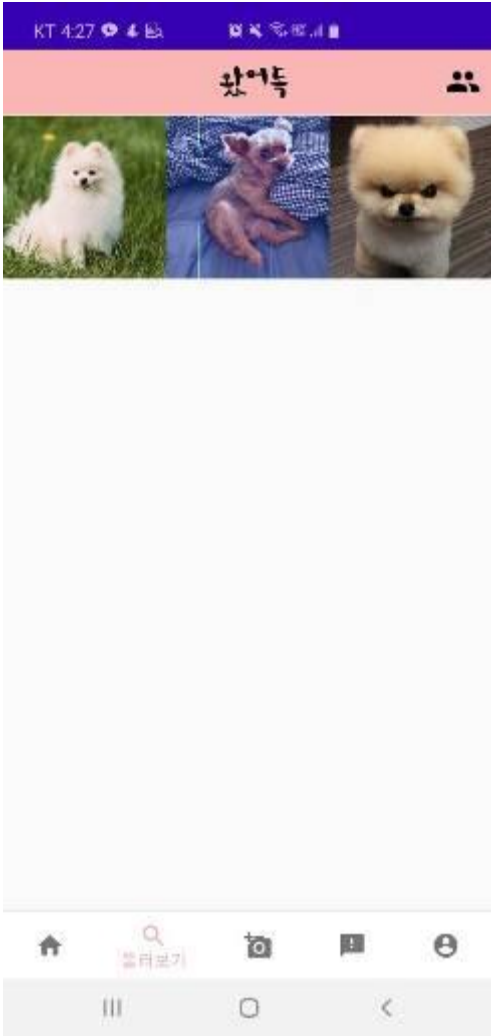

3) 실행 기능

주요기능	상세기능	비고
사용자 인증	로그인, 회원가입, 로그아웃	
SNS 회원 관리	회원 계정 프로필 보기	
피드에 글 올리기	자신의 반려견의 정보를 사진과 함께 게시	
메인 피드	회원들의 반려견 정보들을 자세하게 보기	
간략한 사진 정보 모아보기	회원들의 반려견 사진들을 한번에 모아보기	
알람 기능	자신에게 들어오는 좋아요, 팔로우, 댓글 알람 기능	
개인 프로필	자신의 계정 프로필정보를 볼 수 있음, 또한 상대방의 계정 또한 볼 수 있음 (팔로우, 팔로워, 게시글 수, 내가 게시한 것들)	
채팅 기능	SNS 가입 회원들과 1대1로 반려견 관련 정보를 소통할 수 있는 채팅 환경을 만듦	

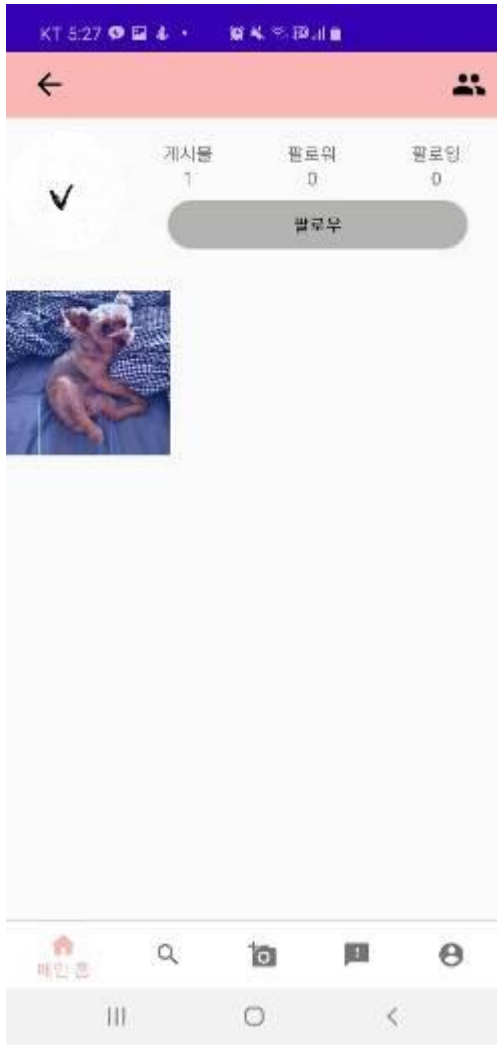

1.3 애플리케이션 실행 화면(실행순서 대로 모든 화면을 캡처해서 넣어주세요..)

화면1	화면2
	
Splash	로그인

화면3	화면4
	
회원가입	메인 홈 화면

화면5	화면6
	
둘러보기 화면	게시글 작성 화면

화면7	화면8
	
내 피드 화면	내 계정 화면

화면9	화면10
	
상대방 계정 화면	댓글 작성 화면

화면11	
	
사용자 리스트 화면	채팅방 화면

- 컬렉션명 : profileImages

<p>+ 컬렉션 시작</p> <p>alarms</p> <p>images</p> <p>profileImages</p> <p>users</p>	<p>+ 문서 추가</p> <p>0UbHySVEDQQ8JPf0hHwI1lSyz3J3</p> <p>LweAE3fJIaWlabacCI9hmtuFQ73</p> <p>SpIwR0hgC1MvQu4jApfoXcfo4Fj2</p>	<p>+ 컬렉션 시작</p> <p>+ 필드 추가</p> <p>image: "https://firebasest 65741.appspot.co alt=media&token=</p>
--	--	--

- 컬렉션명 : alarms

<p>+ 컬렉션 시작</p> <p>alarms</p> <p>images</p> <p>profileImages</p> <p>users</p>	<p>+ 문서 추가</p> <p>0IkX4lIf0vzNTfopHn1x</p> <p>Ar4VkpZUxvkj14xYh07</p> <p>x7sCS4ErgcEnknvYIU</p>	<p>+ 컬렉션 시작</p> <p>+ 필드 추가</p> <p>destinationUid: "0UI kind: 1 message: "별이가 나무 timestamp: 159387091 uid: "SplwR0hgC1MvQu userId: "babo@gmail.c</p>
--	--	---

- 컬렉션명 : follows

<p>+ 컬렉션 시작</p> <p>alarms</p> <p>follows</p> <p>images</p> <p>profileImages</p> <p>users</p>	<p>+ 문서 추가</p> <p>E2yCJp9u6VZjtCgAnMcIGTpsqze2</p> <p>NKz4JLzsQNZ6HkM284vx9CCSWFR2</p> <p>Wp4Yp9LHE7a9Zza9S5vY5D1kop82</p> <p>ZBffCDSg3BfghyGWrkrONWwXK92</p> <p>aov9YT6k1C08FHZ79fzpxGXxSE92</p> <p>heu2rqceyXeswh7GQ3gS8h8M04j2</p>	<p>+ 컬렉션 시작</p> <p>+ 필드 추가</p> <p>followerCount: 0</p> <p>followers</p> <p>followingCount: 1</p> <p>followings</p> <p>E2yCJp9u6VZjtCgAnMcIGTpsqze2: true</p>
---	--	--

2. Realtime Database : wassupdog-65741

- 레퍼런스명 : message




1.5 애플리케이션 구현

1) Splash (구현한 주요 기능에 대해 아래와 같이 설명하세요)

화면(UI)	설명
	<ul style="list-style-type: none"> - Splash 화면은 앱을 켤 때 ‘왓어독’ 마크가 가운데에 나오도록 구현하였습니다 - 1초가 지난 후 로그인 화면으로 이동합니다..
SplashActivity.kt	
<pre> class SplashActivity : AppCompatActivity() { override fun onCreate(savedInstanceState: Bundle?) { super.onCreate(savedInstanceState) SystemClock.sleep(ms: 1000) val intent = Intent(packageContext: this, LoginActivity::class.java) startActivity(intent) finish() } } </pre>	

2) 로그인

화면(UI)	설명
	<ul style="list-style-type: none"> - 로그인 화면은 왔어독 마크와 이메일, 비밀번호 입력 창, 로그인, 회원가입 버튼으로 이루어져 있습니다. - 이메일 혹은 비밀번호 입력창을 비운 상태로 로그인 버튼을 누르면 경고창이 뜹니다. - 올바른 이메일과 비밀번호를 입력하면 메인 화면으로 이동합니다. - 회원가입 버튼을 누르면 회원가입 화면으로 이동합니다.

LoginActivity.kt

```
class LoginActivity : AppCompatActivity() {

    private lateinit var auth: FirebaseAuth

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        // 파이어베이스 통합 관리 객체 생성
        auth = FirebaseAuth.getInstance()

        signUp.setOnClickListener { it: View!
            startActivity(Intent( packageContext: this, RegisterActivity::class.java))
            finish()
        }

        loginBtn.setOnClickListener { it: View!
            doLogin()
        }

    }
}
```

```
// 이메일로 로그인
private fun doLogin() {
    if (sign_Up_Email.text.toString().isEmpty()) {
        sign_Up_Email.error = "이메일을 입력해주세요."
        sign_Up_Email.requestFocus()
        return
    }

    if (!Patterns.EMAIL_ADDRESS.matcher(sign_Up_Email.text.toString()).matches()) {
        sign_Up_Email.error = "유효한 이메일을 입력해주세요."
        sign_Up_Email.requestFocus()
        return
    }

    if (sign_Up_Password.text.toString().isEmpty()) {
        sign_Up_Password.error = "패스워드를 입력해주세요."
        sign_Up_Password.requestFocus()
        return
    }


    auth.signInWithEmailAndPassword(sign_Up_Email.text.toString(), sign_Up_Password.text.toString())
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                val user : FirebaseUser? = auth.currentUser
                updateUI(user)
            } else {
                Toast.makeText(baseContext, text: "로그인에 실패했습니다." , Toast.LENGTH_LONG).show()
                updateUI( currentUser: null)
            }
        }
    }
}
```

```
public override fun onStart() {
    super.onStart()
    val currentUser : FirebaseUser? = auth.currentUser
    updateUI(currentUser)
}

private fun updateUI(currentUser: FirebaseUser?) {

    if (currentUser == null) {
        //
    } else {
        startActivity(Intent( packageContext: this, MainActivity::class.java))
        finish()
    }
}
}
```


3) 회원 가입

화면(UI)	설명
	<ul style="list-style-type: none"> - 이름, 이메일, 비밀번호, 비밀번호 확인을 입력할 수 있는 입력할 수 있습니다. - 입력창이 한 곳이라도 비면 해당 부분을 입력하라는 경고창이 뜹니다. - 비밀번호와 비밀번호 확인 입력이 서로 다르면 경고창이 뜹니다. - 모든 입력창을 올바르게 입력하고 등록완료 버튼을 누르면 파이어스토어에 사용자의 uid, 이름이 저장되고 회원가입이 완료되어 메인 화면으로 이동합니다. - ‘이미 아이디가 있으신가요?’ 를 누르면 다시 로그인 화면으로 이동합니다.

RegisterActivity.kt

```

class RegisterActivity : AppCompatActivity() {

    lateinit var auth : FirebaseAuth
    lateinit var firestore : FirebaseFirestore

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_register)
        auth = FirebaseAuth.getInstance()
        firestore = FirebaseFirestore.getInstance()

        register_button_register.setOnClickListener { it: View!
            signUpUser()
        }

        already_have_account_text_view.setOnClickListener { it: View!
            val intent = Intent( packageContext: this, LoginActivity::class.java)
            startActivity(intent)
        }
    }
}

```

```
// 이메일 회원가입
```

```
private fun signUpUser() {
    if (email_edittext_register.text.toString().isEmpty()) {
        email_edittext_register.error = "이메일을 입력해주세요."
        email_edittext_register.requestFocus()
        return
    }

    if (!Patterns.EMAIL_ADDRESS.matcher(email_edittext_register.text.toString()).matches()) {
        email_edittext_register.error = "유효한 이메일을 입력해주세요."
        email_edittext_register.requestFocus()
        return
    }

    if (password_edittext_register.text.toString().isEmpty()) {
        password_edittext_register.error = "패스워드를 입력해주세요."
        password_edittext_register.requestFocus()
        return
    }

    if (username_edittext_register.text.toString().isEmpty()) {
        username_edittext_register.error = "이름을 입력해주세요."
        username_edittext_register.requestFocus()
        return
    }

    if (pwcheck_edittext_register.text.toString().isEmpty()) {
        pwcheck_edittext_register.error = "패스워드 확인을 입력해주세요."
        pwcheck_edittext_register.requestFocus()
        return
    }
}
```

```
if(password_edittext_register.text.toString() != pwcheck_edittext_register.text.toString()) {
    Toast.makeText( context this, text: "패스워드와 패스워드 확인이 일치하지 않습니다.", Toast.LENGTH_LONG).show()
    return
}

auth.createUserWithEmailAndPassword(email_edittext_register.text.toString(), password_edittext_register.text.toString())
    .addOnCompleteListener { task ->
        if (task.isSuccessful) {
            saveUserToFirestore()
        } else {
            // 에러 메시지 출력
            Toast.makeText(baseContext, text: "회원인증에 실패했습니다. 다시 시도해주세요.", Toast.LENGTH_SHORT).show()
        }
    }
}
```


```
// 파이어스토어에 유저 정보 저장
private fun saveUserToFirestore() {
    val uid : String = FirebaseAuth.getInstance().uid ?: ""

    val db : CollectionReference = FirebaseFirestore.getInstance().collection( collectionPath: "users")

    val user = User(uid, username_edittext_register.text.toString())

    db.document(uid)
        .set(user)
        .addOnSuccessListener { it: Void!
            val intent = Intent( packageContext: this, MainActivity::class.java)
            intent.flags = Intent.FLAG_ACTIVITY_CLEAR_TASK.or(Intent.FLAG_ACTIVITY_NEW_TASK)
            startActivity(intent)
        }
        .addOnFailureListener { it: Exception
        }
    }
}
```

4) 메인 홈 - ①

화면(UI)	설명
	<ul style="list-style-type: none"> - 사용자들이 등록한 게시글을 볼 수 있습니다. - ♥ 버튼을 누르면 좋아요의 개수가 증가합니다. - 💬 버튼을 누르면 댓글을 작성할 수 있습니다. - 🏠 메인 홈 로그인 했을 때의 첫 메인 화면입니다. - 🔍 버튼을 누르면 사용자들이 등록한 사진들을 한 눈에 볼 수 있는 '둘러보기' 화면으로 넘어갑니다. - 📷 버튼을 누르면 게시글 작성이 가능합니다. - 📢 버튼을 누르면 본인에게 온 좋아요, 댓글, 팔로우 알림을 확인할 수 있는 '내 피드' 화면으로 넘어갑니다. - 👤 본인이 작성한 게시글, 팔로잉/팔로워 수 확인, 프로필 사진 변경, 로그아웃이 가능한 '내 계정' 화면으로 이동합니다. - 👤 버튼을 누르면 전체 사용자를 볼 수 있습니다.

MainActivity.kt

```

class MainActivity : AppCompatActivity(), BottomNavigationView.OnNavigationItemSelectedListener {
    private val multiplePermissionsCode = 1000
    //원하는 퍼미션을 이곳에 추가하면 된다.
    private val requiredPermissions = arrayOf(
        Manifest.permission.READ_PHONE_STATE,
        Manifest.permission.READ_EXTERNAL_STORAGE
    )

    override fun onNavigationItemSelected(p0: MenuItem): Boolean {
        setToolbarDefault()
        when(p0.itemId) {
            R.id.action_home -> {
                var dogFragment = HomeFragment()
                supportFragmentManager.beginTransaction().replace(R.id.main_content, dogFragment).commit()
                return true
            }
            R.id.action_search -> {
                var searchFragment = SearchFragment()
                supportFragmentManager.beginTransaction().replace(R.id.main_content, searchFragment).commit()
                return true
            }
            R.id.action_add_photo -> {
                checkPermissions() //권한허용
                if(ContextCompat.checkSelfPermission( context: this,
                    Manifest.permission.READ_EXTERNAL_STORAGE) == PackageManager.PERMISSION_GRANTED){
                    startActivity(Intent( packageContext: this,
                        AddPhotoActivity::class.java))
                }
                return true
            }
        }
    }
}

```

```

        R.id.action_alarm-> {
            var alarmFragment = AlarmFragment()
            supportFragmentManager.beginTransaction().replace(R.id.main_content, alarmFragment).commit()
            return true
        }

        R.id.action_account -> {
            var userFragment = AccountFragment()
            var bundle = Bundle()
            var uid : String? = FirebaseAuth.getInstance().currentUser?.uid

            bundle.putString("destinationUid", uid)
            userFragment.arguments = bundle
            supportFragmentManager.beginTransaction().replace(R.id.main_content, userFragment).commit()
            return true
        }
    }

    return false
}

fun setToolbarDefault(){
    toolbar_btn_chat.visibility = View.VISIBLE
    toolbar_btn_back.visibility = View.GONE
    toolbar_title_image.visibility = View.VISIBLE
}

```

```

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    bottom_navigation.setOnNavigationItemSelectedListener(this)

    // 첫 화면 홈으로 고정
    bottom_navigation.selectedItemId = R.id.action_home

    // 채팅방으로 이동
    toolbar_btn_chat.setOnClickListener { it: View?
        val intent = Intent( packageContext: this, ChatListActivity::class.java)
        startActivity(intent)
    }
}

//퍼미션 체크 및 권한 요청 함수
private fun checkPermissions() {
    //거절되었거나 아직 수락하지 않은 권한(퍼미션)을 저장할 문자열 배열 리스트
    var rejectedPermissionList = ArrayList<String>()

    //필요한 퍼미션들을 하나씩 고집어내서 현재 권한을 받았는지 체크
    for(permission : String in requiredPermissions){
        if(ContextCompat.checkSelfPermission( context: this, permission) != PackageManager.PERMISSION_GRANTED) {
            //만약 권한이 없다면 rejectedPermissionList에 추가
            rejectedPermissionList.add(permission)
        }
    }

    //거절된 퍼미션이 있다면...
    if(rejectedPermissionList.isNotEmpty()){
        //권한 요청!
        val array : Array<String?> = arrayOfNulls<String>(rejectedPermissionList.size)
        ActivityCompat.requestPermissions( activity: this, rejectedPermissionList.toArray(array), multiplePermissionsCode)
    }
}

```

```

}

//권한 요청 결과 함수
override fun onRequestPermissionsResult(requestCode: Int,
                                       permissions: Array<String>, grantResults: IntArray) {
    when (requestCode) {
        multiplePermissionsCode -> {
            if(grantResults.isNotEmpty()) {
                for((i :Int , permission :String) in permissions.withIndex()) {
                    if(grantResults[i] != PackageManager.PERMISSION_GRANTED) {
                        //권한 획득 실패
                        Log.i( tag: "TAG", msg: "The user has denied to $permission")
                        Log.i( tag: "TAG", msg: "I can't work for you anymore then. ByeBye!")
                    }
                }
            }
        }
    }
}

```


```

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    if(requestCode == AccountFragment.PICK_PROFILE_FROM_ALBUM && resultCode == Activity.RESULT_OK){
        var imageUri :Uri? = data?.data
        var uid :String? = FirebaseAuth.getInstance().currentUser?.uid
        var storageRef :StorageReference = FirebaseStorage.getInstance().reference.child( pathString: "userProfileImages").child(uid!!)
        storageRef.putFile(imageUri!!).continueWithTask { task: Task<UploadTask.TaskSnapshot> ->
            return@continueWithTask storageRef.downloadUrl
        }.addOnSuccessListener { uri ->
            var map = HashMap<String, Any>()
            map["image"] = uri.toString()
            FirebaseFirestore.getInstance().collection( collectionPath: "profileImages").document(uid).set(map)
        }
    }
}

```

5) 메인 홈 - ②

화면(UI)	설명
	<ul style="list-style-type: none"> - 메인 홈 - ① 과 동일한 기능입니다. - 상대방 계정 프로필 이미지를 누르면 상대방 계정 페이지로 이동합니다.

HomeFragment.kt

```
class HomeFragment : Fragment() {
    var firestore: FirebaseFirestore? = null
    var uid: String? = null
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        var view: View? = LayoutInflater.from(activity).inflate(R.layout.fragment_home, container, attachToRoot: false)
        firestore = FirebaseFirestore.getInstance()
        uid = FirebaseAuth.getInstance().currentUser?.uid

        view.detailviewfragment_recyclerview.adapter = DetailViewRecyclerViewAdapter()
        view.detailviewfragment_recyclerview.layoutManager = LinearLayoutManager(activity)
        return view
    }

    inner class DetailViewRecyclerViewAdapter : RecyclerView.Adapter<RecyclerView.ViewHolder>() {
        var contentModels: ArrayList<ContentModel> = arrayListOf()
        var contentUidList: ArrayList<String> = arrayListOf()

        init {

            firestore?.collection( collectionPath: "images")?.orderBy( field: "timestamp")
                ?.addSnapshotListener { querySnapshot, firebaseFirestoreException ->
                    contentModels.clear()
                    contentUidList.clear()
                    //Sometimes, This code return null of querySnapshot when it signout
                    if (querySnapshot == null) return@addSnapshotListener
                }
        }
    }
}
```

```

        for (snapshot : DocumentSnapshot! in querySnapshot!!.documents) {
            var item : ContentModel? = snapshot.toObject(ContentModel::class.java)
            contentModels.add(item!!)
            contentUidList.add(snapshot.id)
        }
        notifyDataSetChanged()
    }
}

override fun onCreateViewHolder(p0: ViewGroup, p1: Int): RecyclerView.ViewHolder {
    var view : View! = LayoutInflater.from(p0.context).inflate(R.layout.item_detail, p0, attachToRoot: false)
    return CustomViewHolder(view)
}

inner class CustomViewHolder(view: View) : RecyclerView.ViewHolder(view)

override fun getItemCount(): Int {
    return contentModels.size
}

override fun onBindViewHolder(p0: RecyclerView.ViewHolder, p1: Int) {
    var viewholder : View = (p0 as CustomViewHolder).itemView

    // 유저 아이디
    viewholder.detailviewitem_profile_textview.text = contentModels!![p1].userId

    // 이미지
    Glide.with(p0.itemView.context).load(contentModels!![p1].imageUrl).into(viewholder.detailviewitem_imageview_content)

```

```

// 콘텐츠 설명
viewholder.detailviewitem_explain_textview.text = contentModels!![p1].explain

// 프로필 이미지
FirebaseFirestore.getInstance()
    .collection( collectionPath: "profileImages")
    .document(contentModels[p1].uid!!)
    .get()
    .addOnCompleteListener { task ->
        if(task.isSuccessful){
            var url : Any? = task.result!!["image"]
            Glide.with(p0.itemView.context).load(url).apply(RequestOptions().circleCrop()).into(viewholder.detailviewitem_profile_image)
        }
    }

// 좋아요
viewholder.detailviewitem_favoritecounter_textview.text = "좋아요 " + contentModels!![p1].favoriteCount

// 버튼 클릭시
viewholder.detailviewitem_favorite_imageview.setOnClickListener { it: View!
    favoriteEvent(p1)
}

// 좋아요 색깔 이벤트
if(contentModels!![p1].favorites.containsKey(uid)){

```



```

if(contentModels!![p1].favorites.containsKey(uid)){
    // 좋아요 클릭 o
    viewHolder.detailviewitem_favorite_imageview.setImageResource(R.drawable.ic_favorite)
}else{
    // 좋아요 클릭 x
    viewHolder.detailviewitem_favorite_imageview.setImageResource(R.drawable.ic_favorite_border)
}

// 이미지 클릭 시 발생 이벤트
viewHolder.detailviewitem_profile_image.setOnClickListener { it: View!
    var fragment = AccountFragment()
    var bundle = Bundle()
    bundle.putString("destinationUid", contentModels[p1].uid)
    bundle.putString("userId", contentModels[p1].userId)
    fragment.arguments = bundle
    activity?.supportFragmentManager?.beginTransaction()?.replace(R.id.main_content,fragment)?.commit()
}
viewholder.detailviewitem_comment_imageview.setOnClickListener { v ->
    var intent = Intent(v.context,CommentActivity::class.java)
    intent.putExtra( name: "contentUid", contentUidList[p1])
    intent.putExtra( name: "destinationUid", contentModels[p1].uid)
    startActivity(intent)
}
}
}

```

```

fun favoriteEvent(position : Int){
    var tsDoc : DocumentReference? = firestore?.collection( collectionPath: "images")?.document(contentUidList[position])
    firestore?.runTransaction { transaction ->

        var contentDTO : ContentModel? = transaction.get(tsDoc!!).toObject(ContentModel::class.java)

        if(contentDTO!!.favorites.containsKey(uid)){
            // 버튼을 클릭했을 때
            contentDTO?.favoriteCount = contentDTO?.favoriteCount - 1
            contentDTO?.favorites.remove(uid)
        }else{
            // 버튼을 클릭하지 않았을 때
            contentDTO?.favoriteCount = contentDTO?.favoriteCount + 1
            contentDTO?.favorites[uid!!] = true
            favoriteAlarm(contentModels[position].uid!!)
        }
        transaction.set(tsDoc,contentDTO) ^runTransaction
    }
}

```

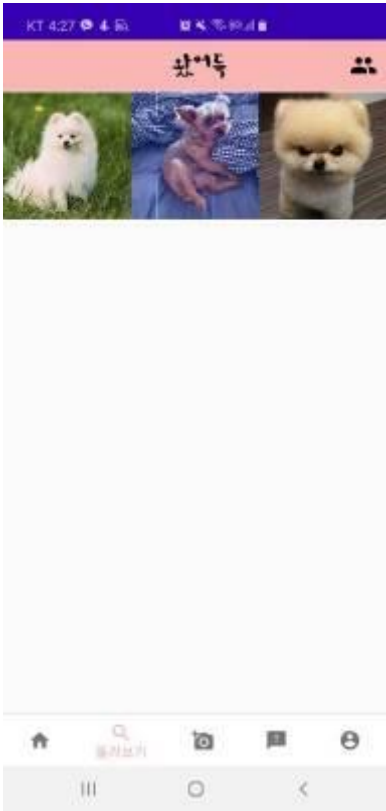
```

}
fun favoriteAlarm(destinationUid : String){
    var alarmDTO = AlarmModel()
    alarmDTO.destinationUid = destinationUid
    alarmDTO.userId = FirebaseAuth.getInstance().currentUser?.email
    alarmDTO.uid = FirebaseAuth.getInstance().currentUser?.uid
    alarmDTO.kind = 0
    alarmDTO.timestamp = System.currentTimeMillis()
    FirebaseFirestore.getInstance().collection( collectionPath: "alarms").document().set(alarmDTO)

    var message :String = FirebaseAuth.getInstance()?.currentUser?.email + "좋아요를 눌렀습니다."
    FcmPush.instance.sendMessage(destinationUid, title: "왔어독",message)
}
}

```

6) 둘러보기

화면(UI)	설명
	<ul style="list-style-type: none"> - 🔍 버튼을 눌렀을 때의 화면입니다. - 파이어스토어에 저장된 모든 이미지 url을 불러와 사용자들이 등록한 게시글을 한눈에 확인할 수 있습니다.

SearchFragment.kt

```

class SearchFragment : Fragment() {
    var firestore : FirebaseFirestore? = null
    var fragmentView : View? = null

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {
        fragmentView = LayoutInflater.from(activity).inflate(R.layout.fragment_search, container, attachToRoot: false)
        firestore = FirebaseFirestore.getInstance()
        fragmentView?.searchfragment_recyclerview?.adapter = UserFragmentRecyclerViewAdapter()
        fragmentView?.searchfragment_recyclerview?.LayoutManager =
            GridLayoutManager(activity, spanCount: 3)
        return fragmentView
    }

    inner class UserFragmentRecyclerViewAdapter : RecyclerView.Adapter<RecyclerView.ViewHolder>(){
        var contentDTOS : ArrayList<ContentModel> = arrayListOf()
        init {
            firestore?.collection( collectionPath: "images")?.addSnapshotListener { querySnapshot, firebaseFirestoreException ->
                //Sometimes, This code return null of querySnapshot when it signout
                if(querySnapshot == null) return@addSnapshotListener

                //Get data
                for(snapshot : DocumentSnapshot! in querySnapshot.documents){
                    contentDTOS.add(snapshot.toObject(ContentModel::class.java!!))
                }
                notifyDataSetChanged()
            }
        }
    }
}

```

```

        override fun onCreateViewHolder(p0: ViewGroup, p1: Int): RecyclerView.ViewHolder {
            var width :Int = resources.displayMetrics.widthPixels / 3

            var imageView = ImageView(p0.context)
            imageView.layoutParams = LinearLayoutCompat.LayoutParams(width,width)
            return CustomViewHolder(imageview)
        }

        inner class CustomViewHolder(var imageView: ImageView) : RecyclerView.ViewHolder(imageview) {

        }

        override fun getItemCount(): Int {
            return contentDTOs.size
        }

        override fun onBindViewHolder(p0: RecyclerView.ViewHolder, p1: Int) {
            var imageView :ImageView = (p0 as CustomViewHolder).imageView
            Glide.with(p0.itemView.context).load(contentDTOs[p1].imageUrl).apply(RequestOptions().centerCrop()).into(imageview)
        }
    }
}

```

7) 게시물 작성

화면(UI)	설명
	<ul style="list-style-type: none"> - 카메라 버튼을 눌렀을 때의 화면입니다. - '저장공간' 에 대한 권한을 허용하겠다는 메시지가 나타납니다. - 허용을 하면 내 갤러리를 통해 이미지를 업로드할 수 있습니다. - 간단한 이미지에 대한 설명을 작성할 수 있습니다. - '이미지 업로드' 버튼을 누르면 게시물 작성이 완료됩니다.
AddPhotoActivity.kt	

```

class AddPhotoActivity : AppCompatActivity() {
    var PICK_IMAGE_FROM_ALBUM = 0
    var storage : FirebaseStorage? = null
    var photoUri : Uri? = null
    var auth : FirebaseAuth? = null
    var firestore : FirebaseFirestore? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_add_photo)

        storage = FirebaseStorage.getInstance()
        auth = FirebaseAuth.getInstance()
        firestore = FirebaseFirestore.getInstance()

        // 앨범 열기
        var photoPickerIntent = Intent(Intent.ACTION_PICK)
        photoPickerIntent.type = "image/*"
        startActivityForResult(photoPickerIntent, PICK_IMAGE_FROM_ALBUM)

        // 이미지 업로드 이벤트
        addphoto_btn_upload.setOnClickListener { it: View!
            contentUpload()
        }
    }
}

```

```

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if(requestCode == PICK_IMAGE_FROM_ALBUM){
        if(resultCode == Activity.RESULT_OK){
            //이미지 경로
            photoUri = data?.data
            addphoto_image.setImageURI(photoUri)
        }else{
            finish()
        }
    }
}
}

```

```

fun contentUpload(){
    // 파일 이름
    var timestamp : String = SimpleDateFormat( pattern: "yyyyMMdd_HH:mm:ss").format(Date())
    var imageFileName : String = "IMAGE_" + timestamp + ".png"

    var storageRef : StorageReference? = storage?.reference?.child( pathString: "images")?.child(imageFileName)


    // 파일 업로드(Promise : 구글 권장 방식)
    storageRef?.putFile(photoUri!!)?.continueWithTask { task: Task<UploadTask.TaskSnapshot> ->
        return@continueWithTask storageRef.downloadUrl
    }?.addOnSuccessListener { uri ->
        var contentModels = ContentModel()

        // 이미지의 다운로드 url
        contentModels.imageUrl = uri.toString()
        // 유저의 uid
        contentModels.uid = auth?.currentUser?.uid
        // 유저 아이디
        contentModels.userId = auth?.currentUser?.email
        // 콘텐츠 설명
        contentModels.explain = addphoto_edit_explain.text.toString()
        // 시간
        contentModels.timestamp = System.currentTimeMillis()

        firestore?.collection( collectionPath: "images")?.document()?.set(contentModels)
        setResult(Activity.RESULT_OK)
        finish()
    }
}

```

8) 내 피드

화면(UI)	설명
	<ul style="list-style-type: none"> - 버튼을 눌렀을 때의 화면입니다. - 자신의 게시글에 좋아요, 댓글이 달렸을 때 알림을 확인할 수 있습니다. - 자신의 계정을 팔로우 했을 때 알림을 확인할 수 있습니다.

AlarmFragment.kt

```
class AlarmFragment : Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        var view :View? = LayoutInflater.from(activity).inflate(R.layout.fragment_alarm, container, attachToRoot: false)
        view?.alarmfragment_recyclerview.adapter = AlarmRecyclerviewAdapter()
        view?.alarmfragment_recyclerview.layoutManager = LinearLayoutManager(activity)

        return view
    }

    inner class AlarmRecyclerviewAdapter : RecyclerView.Adapter<RecyclerView.ViewHolder>() {
        var alarmModelList: ArrayList<AlarmModel> = arrayListOf()

        init {
            val uid :String? = FirebaseAuth.getInstance().currentUser?.uid

            FirebaseFirestore.getInstance().collection( collectionPath: "alarms").whereEqualTo( field: "destinationUid", uid)
                .addSnapshotListener { querySnapshot, firebaseFirestoreException ->
                    alarmModelList.clear()
                    if (querySnapshot == null) return@addSnapshotListener

                    for (snapshot : DocumentSnapshot! in querySnapshot.documents) {
                        alarmModelList.add(snapshot.toObject(AlarmModel::class.java!!))
                    }
                    notifyDataSetChanged()
                }
        }
    }
}
```

```

override fun onCreateViewHolder(p0: ViewGroup, p1: Int): RecyclerView.ViewHolder {
    var view :View = LayoutInflater.from(p0.context).inflate(R.layout.item_comment, p0, attachToRoot: false)

    return CustomViewHolder(view)
}

inner class CustomViewHolder(view: View) : RecyclerView.ViewHolder(view)

override fun getItemCount(): Int {
    return alarmModellist.size
}

override fun onBindViewHolder(p0: RecyclerView.ViewHolder, p1: Int) {
    var view :View = p0.itemView

    FirebaseFirestore.getInstance().collection( collectionPath: "profileImages")
        .document(alarmModellist[p1].uid!!).get().addOnCompleteListener { task ->
        if (task.isSuccessful) {
            val url :Any? = task.result!!["image"]
            Glide.with(view.context).load(url).apply(RequestOptions().circleCrop())
                .into(view.commentviewitem_imageview_profile)
        }
    }
}

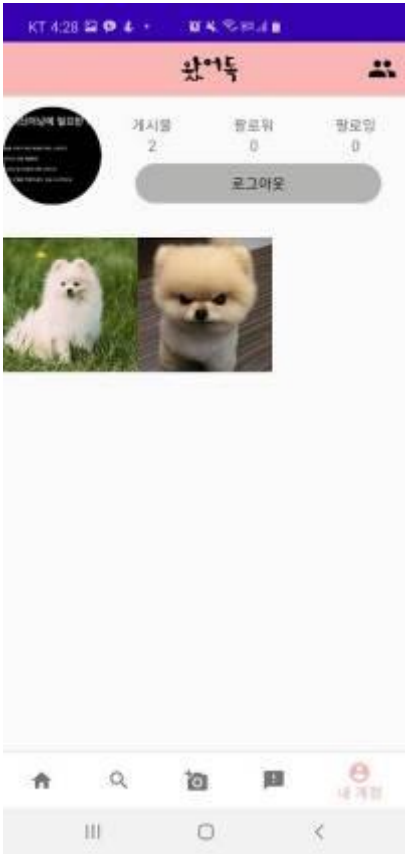

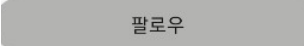
```

```

when (alarmModellist[p1].kind) {
    0 -> {
        val str_0 :String = alarmModellist[p1].userId + " 님이 " + "좋아요를 눌렀습니다."
        view.commentviewitem_textview_profile.text = str_0
    }
    1 -> {
        val str_0 :String =
            alarmModellist[p1].userId + " 님이 " + "댓글을 남겼습니다."
        view.commentviewitem_textview_profile.text = str_0
    }
    2 -> {
        val str_0 :String = alarmModellist[p1].userId + " 님이 " + "팔로우를 시작했습니다."
        view.commentviewitem_textview_profile.text = str_0
    }
}
view.commentviewitem_textview_comment.visibility = View.INVISIBLE
}
}

```

9) 내 계정 / 상대방 계정

화면(UI)	설명
	<ul style="list-style-type: none"> -  버튼을 눌렀을 때의 화면입니다. - 자신이 올린 게시물과 팔로워, 팔로잉 수를 확인할 수 있습니다. - 프로필 이미지를 누르면 내 갤러리로 이동하고 프로필 이미지를 변경할 수 있습니다. - 로그아웃 버튼을 누르면 현재 로그인 된 계정이 로그아웃 됩니다. - 상대방 계정 화면은 로그아웃 버튼이  버튼으로 변경되어 있다는 점을 빼면 동일합니다. - 상대방이 올린 게시물과 팔로워, 팔로잉 수를 확인할 수 있습니다. - 팔로우 버튼을 누르면 상대방의 팔로워 수가 증가하고 자신의 계정은 팔로잉 수가 증가합니다. - 팔로우 버튼을 누르면 해당 사용자에게 팔로우 알림이 전송됩니다. - 팔로우 버튼을 누르고 난 후 팔로우 버튼이 '팔로우 취소' 버튼으로 변경됩니다. - 팔로우 취소버튼을 누르면 팔로우가 취소됩니다.

AccountFragment.kt

```

class AccountFragment : Fragment() {
    var fragmentView : View? = null
    var firestore : FirebaseFirestore? = null
    var uid : String? = null
    var auth : FirebaseAuth? = null
    var currentUserUid : String? = null
    companion object {
        var PICK_PROFILE_FROM_ALBUM = 10
    }
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {
        fragmentView = LayoutInflater.from(activity).inflate(R.layout.fragment_account, container, attachToRoot: false)
        uid = arguments?.getString( key: "destinationUid")
        firestore = FirebaseFirestore.getInstance()
        auth = FirebaseAuth.getInstance()
        currentUserUid = auth?.currentUser?.uid

        if(uid == currentUserUid){
            // 마이 페이지
            fragmentView?.account_btn_follow_signout?.text = "로그아웃"
            fragmentView?.account_btn_follow_signout?.setOnClickListener { it: View?
                activity?.finish()
                startActivity(Intent(activity, LoginActivity::class.java))
                auth?.signOut()
            }
        }
    }
}

```

```

    }else{
        // 상대방 페이지
        fragmentView?.account_btn_follow_signout?.text = "팔로우"
        var mainactivity : MainActivity = (activity as MainActivity)
        // mainactivity?.toolbar_username?.text = arguments?.getString("userId")
        mainactivity?.toolbar_btn_back?.setOnClickListener { it: View!
            mainactivity.bottom_navigation.selectedItemId = R.id.action_home
        }
        mainactivity?.toolbar_title_image?.visibility = View.GONE
        // mainactivity?.toolbar_username?.visibility = View.VISIBLE
        mainactivity?.toolbar_btn_back?.visibility = View.VISIBLE
        fragmentView?.account_btn_follow_signout?.setOnClickListener { it: View!
            requestFollow()
        }
    }

    fragmentView?.account_recyclerview?.adapter = UserFragmentRecyclerViewAdapter()
    fragmentView?.account_recyclerview?.LayoutManager = GridLayoutManager(activity!!, spanCount: 3)

    if(uid == currentUserUid) {
        fragmentView?.account_iv_profile?.setOnClickListener { it: View!
            var photoPickerIntent = Intent(Intent.ACTION_PICK)
            photoPickerIntent.type = "image/*"
            activity?.startActivityForResult(photoPickerIntent,PICK_PROFILE_FROM_ALBUM)
        }
    }
    getProfileImage()
    getFollowerAndFollowing()
    return fragmentView
}

```

```

fun getFollowerAndFollowing(){
    firestore?.collection( collectionPath: "users")?.document(uid!!)?.addSnapshotListener { documentSnapshot, firebaseFirestoreException ->
        if(documentSnapshot == null) return@addSnapshotListener
        var followModels : FollowModel? = documentSnapshot.toObject(FollowModel::class.java)
        if(followModels?.followingCount != null){
            fragmentView?.account_tv_following_count?.text = followModels?.followingCount?.toString()
        }
        if(followModels?.followerCount != null){
            fragmentView?.account_tv_follower_count?.text = followModels?.followerCount?.toString()
            if(followModels?.followers?.containsKey(currentUserUid!!)){
                fragmentView?.account_btn_follow_signout?.text = "팔로우 취소"
                fragmentView?.account_btn_follow_signout?.background
                    ?.setColorFilter(ContextCompat.getColor(activity!!,R.color.colorLightGray),
                        PorterDuff.Mode.MULTIPLY)
            }else{
                if(uid != currentUserUid){
                    fragmentView?.account_btn_follow_signout?.text = "팔로우"
                    fragmentView?.account_btn_follow_signout?.background?.colorFilter = null
                }
            }
        }
    }
}
}

```



```

fun requestFollow(){
    // 나의 데이터 저장
    var tsDocFollowing : DocumentReference? = firestore?.collection( collectionPath: "users")?.document(currentUserId!!)
    firestore?.runTransaction { transaction ->
        var followModels : FollowModel? = transaction.get(tsDocFollowing!!).toObject(FollowModel::class.java)
        if(followModels == null){
            followModels = FollowModel()
            followModels!!.followingCount = 1
            followModels!!.followings[uid!!] = true

            transaction.set(tsDocFollowing, followModels!!)
            return@runTransaction
        }

        if(followModels?.followings.containsKey(uid)){
            // 팔로우 취소
            followModels?.followingCount = followModels?.followingCount - 1
            followModels?.followings.remove(uid)
        }else{
            // 팔로우 추가
            followModels?.followingCount = followModels?.followingCount + 1
            followModels?.followings[uid!!] = true
        }
        transaction.set(tsDocFollowing, followModels)
        return@runTransaction
    }
}

```

```

// 3자에게 데이터 저장
var tsDocFollower : DocumentReference? = firestore?.collection( collectionPath: "users")?.document(uid!!)
firestore?.runTransaction { transaction ->
    var followDTO : FollowModel? = transaction.get(tsDocFollower!!).toObject(FollowModel::class.java)
    if(followDTO == null){
        followDTO = FollowModel()
        followDTO!!.followerCount = 1
        followDTO!!.followers[currentUserId!!] = true
        followerAlarm(uid!!)
        transaction.set(tsDocFollower, followDTO!!)
        return@runTransaction
    }

    if(followDTO!!.followers.containsKey(currentUserId!!)){
        // 팔로워 취소
        followDTO!!.followerCount = followDTO!!.followerCount - 1
        followDTO!!.followers.remove(currentUserId!!)
    }else{
        // 팔로워 추가
        followDTO!!.followerCount = followDTO!!.followerCount + 1
        followDTO!!.followers[currentUserId!!] = true
        followerAlarm(uid!!)
    }
    transaction.set(tsDocFollower, followDTO!!)
    return@runTransaction
}

```

```

fun followerAlarm(destinationUid : String){
    var alarmModels = AlarmModel()
    alarmModels.destinationUid = destinationUid
    alarmModels.userId = auth?.currentUser?.email
    alarmModels.uid = auth?.currentUser?.uid
    alarmModels.kind = 2
    alarmModels.timestamp = System.currentTimeMillis()
    FirebaseFirestore.getInstance().collection( collectionPath: "alarms").document().set(alarmModels)

    var message :String = auth?.currentUser?.email + "팔로우를 시작했습니다."
    FcmPush.instance.sendMessage(destinationUid, title: "왔어득",message)
}

fun getProfileImage(){
    firestore?.collection( collectionPath: "profileImages")?.document(uid!!)?.addSnapshotListener { documentSnapshot, firebaseFirestoreException ->
        if(documentSnapshot == null) return@addSnapshotListener
        if(documentSnapshot.data != null){
            var url :Any? = documentSnapshot?.data!!["image"]
            Glide.with(activity!!).load(url).apply(RequestOptions().circleCrop()).into(fragmentView?.account_iv_profile!!)
        }
    }
}
}

```

```

inner class UserFragmentRecyclerViewAdapter : RecyclerView.Adapter<RecyclerView.ViewHolder>(){
    var contentModels : ArrayList<ContentModel> = arrayListOf()
    init {
        firestore?.collection( collectionPath: "images")?.whereEqualTo( field: "uid",uid)?.addSnapshotListener { querySnapshot, firebaseFirestoreException ->
            //Sometimes, This code return null of querySnapshot when it signout
            if(querySnapshot == null) return@addSnapshotListener

            //Get data
            for(snapshot : DocumentSnapshot! in querySnapshot.documents){
                contentModels.add(snapshot.toObject(ContentModel::class.java!!))
            }
            fragmentView?.account_tv_post_count?.text = contentModels.size.toString()
            notifyDataSetChanged()
        }
    }
}

```

```

override fun onCreateViewHolder(p0: ViewGroup, p1: Int): RecyclerView.ViewHolder {
    var width :Int = resources.displayMetrics.widthPixels / 3

    var imageview = ImageView(p0.context)
    imageview.LayoutParams = LinearLayoutCompat.LayoutParams(width,width)
    return CustomViewHolder(imageview)
}

inner class CustomViewHolder(var imageview: ImageView) : RecyclerView.ViewHolder(imageview) {

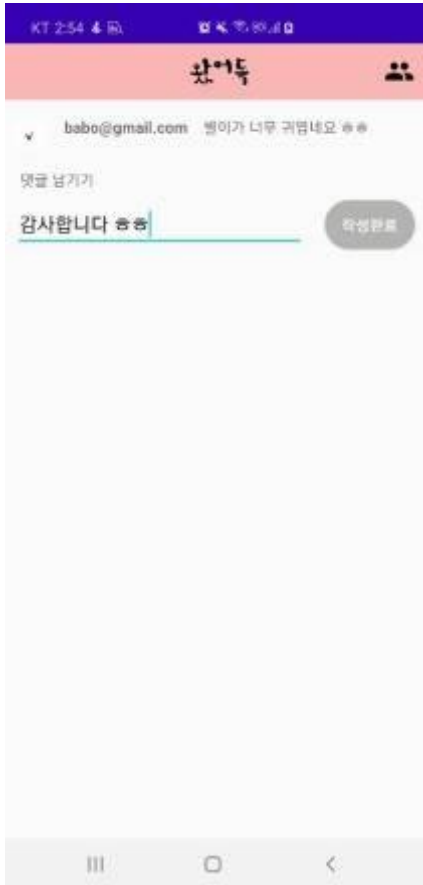

}

override fun getItemCount(): Int {
    return contentModels.size
}

override fun onBindViewHolder(p0: RecyclerView.ViewHolder, p1: Int) {
    var imageview :ImageView = (p0 as CustomViewHolder).imageview
    Glide.with(p0.itemView.context).load(contentModels[p1].imageUrl).apply(RequestOptions().centerCrop()).into(imageview)
}
}

```

10) 댓글 작성

화면(UI)	설명
	<ul style="list-style-type: none"> -  버튼을 누르면 댓글을 작성할 수 있고 작성된 댓글을 볼 수 있습니다. - 간단한 텍스트를 작성할 수 있는 텍스트 입력창이 있습니다. - 작성완료 버튼을 누르면 댓글 작성이 완료됩니다. - 댓글 작성이 완료되면 해당 사용자에게 알림이 전송됩니다. - 작성된 댓글은 '댓글 남기기' 텍스트 위쪽에서 확인할 수 있습니다.

CommentActivity.kt

```
class CommentActivity : AppCompatActivity() {

    var contentUid : String? = null
    var destinationUid : String? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_comment)
        contentUid = intent.getStringExtra( name: "contentUid")
        destinationUid = intent.getStringExtra( name: "destinationUid")

        comment_recyclerview.adapter = CommentRecyclerViewAdapter()
        comment_recyclerview.layoutManager = LinearLayoutManager( context: this)

        comment_btn_send?.setOnClickListener { lt View!
            var comment = ContentModel.Comment()
            comment.userId = FirebaseAuth.getInstance().currentUser?.email
            comment.uid = FirebaseAuth.getInstance().currentUser?.uid
            comment.comment = comment_edit_message.text.toString()
            comment.timestamp = System.currentTimeMillis()

            FirebaseFirestore.getInstance().collection( collectionPath: "images").document(contentUid!!).collection( collectionPath: "comments").document().set(comment)
            commentAlarm(destinationUid!!,comment_edit_message.text.toString())
            comment_edit_message.setText("")
        }
    }
}
```

```

fun commentAlarm(destinationUid : String, message : String){
    var alarmDTO = AlarmModel()
    alarmDTO.destinationUid = destinationUid
    alarmDTO.userId = FirebaseAuth.getInstance().currentUser?.email
    alarmDTO.kind = 1
    alarmDTO.uid = FirebaseAuth.getInstance().currentUser?.uid
    alarmDTO.timestamp = System.currentTimeMillis()
    alarmDTO.message = message
    FirebaseFirestore.getInstance().collection( collectionPath: "alarms").document().set(alarmDTO)

    var msg : String = FirebaseAuth.getInstance().currentUser?.email + " " + "댓글을 남겼습니다." + " of " + message
    FcmPush.instance.sendMessage(destinationUid, title: "왔어특",msg)
}

inner class CommentRecyclerviewAdapter : RecyclerView.Adapter<RecyclerView.ViewHolder>(){

    var comments : ArrayList<ContentModel.Comment> = arrayListOf()
    init {
        FirebaseFirestore.getInstance()
            .collection( collectionPath: "images")
            .document(contentUid!!)
            .collection( collectionPath: "comments")
            .orderBy( field: "timestamp")
            .addSnapshotListener { querySnapshot, firebaseFirestoreException ->
                comments.clear()
                if(querySnapshot == null)return@addSnapshotListener

                for(snapshot : DocumentSnapshot! in querySnapshot.documents!!){
                    comments.add(snapshot.toObject(ContentModel.Comment::class.java!!))
                }
                notifyDataSetChanged()
            }
    }
}

```

```

override fun onCreateViewHolder(p0: ViewGroup, p1: Int): RecyclerView.ViewHolder {
    var view : View! = LayoutInflater.from(p0.context).inflate(R.layout.item_comment,p0, attachToRoot: false)
    return CustomViewHolder(view)
}

private inner class CustomViewHolder(view : View) : RecyclerView.ViewHolder(view)



override fun getItemCount(): Int {
    return comments.size
}

override fun onBindViewHolder(p0: RecyclerView.ViewHolder, p1: Int) {
    var view : View = p0.itemView
    view.commentviewitem_textview_comment.text = comments[p1].comment
    view.commentviewitem_textview_profile.text = comments[p1].userId

    FirebaseFirestore.getInstance()
        .collection( collectionPath: "profileImages")
        .document(comments[p1].uid!!)
        .get()
        .addOnCompleteListener { task ->
            if(task.isSuccessful){
                var url : Any? = task.result!!["image"]
                Glide.with(p0.itemView.context).load(url).apply(RequestOptions().circleCrop()).into(view.commentviewitem_imageview_profile)
            }
        }
}
}

```

11) 사용자 리스트

화면(UI)	설명
	<ul style="list-style-type: none"> -  버튼을 누르면 전체 사용자를 볼 수 있습니다. - 현재 왔어독 서비스를 사용하고 있는 전체 사용자들을 볼 수 있습니다. (본인은 제외) - 전체 사용자 리스트 중 한 명을 선택하면 해당하는 사용자와 1:1 채팅을 할 수 있는 채팅방으로 이동합니다.

ChatListActivity.kt

```

class ChatListActivity : AppCompatActivity() {

    private val TAG = ChatListActivity::class.java.simpleName

    val db = FirebaseFirestore.getInstance()
    var auth = FirebaseAuth.getInstance()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_chat_list)

        val adapter = GroupAdapter<GroupViewHolder>()

        recyclerview_list.adapter = adapter

        db.collection( collectionPath: "users")
            .get()
            .addOnSuccessListener { result ->
                for(document : QueryDocumentSnapshot! in result) {
                    // 자기 자신은 채팅 리스트에 출력하지 않을
                    if(document.get("uid").toString() != auth.currentUser?.uid)
                        adapter.add(UserItem(document.get("username").toString(), document.get("uid").toString()))
                    Log.d(TAG, document.get("username").toString())
                    Log.d(TAG, msg: "${document.id} => ${document.data}")
                }
                recyclerview_list.adapter = adapter
            }
            .addOnFailureListener { exception ->
                Log.d(TAG, msg: "Error getting documents", exception)
            }
    }
}

```

```

adapter.setOnItemClickListener { item, view ->

    Log.d(TAG, (item as UserItem).name)
    Log.d(TAG, (item as UserItem).uid)


    val name : String = (item as UserItem).name
    val uid : String = (item as UserItem).uid

    val intent = Intent( packageContext: this, ChatRoomActivity::class.java)
    intent.putExtra( name: "name", name)
    intent.putExtra( name: "yourUid", uid)
    startActivity(intent)

}

}
    
```

12) 1:1 채팅

화면(UI)	설명
	<p>동욱</p> <ul style="list-style-type: none"> - 사용자 이름을 눌렀을 때의 화면입니다. - 이곳에서 1:1 메시지를 주고 받을 수 있습니다. - 내가 보낸 메시지는 '나' 로 표시되며 오른쪽에 나타납니다. - 상대방이 보낸 메시지는 '동욱'과 같이 이름으로 표시되고, 왼쪽에 나타납니다. - 메시지 입력칸에 메시지를 입력한 후 '전송' 버튼을 누르면 메시지가 전송됩니다.

ChatRoomActivity.kt

```

class ChatRoomActivity : AppCompatActivity() {

    private lateinit var auth : FirebaseAuth
    private lateinit var db : FirebaseFirestore
    private val TAG = ChatRoomActivity::class.java.simpleName

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_chat_room)

        auth = FirebaseAuth.getInstance()
        db = FirebaseFirestore.getInstance()

        val adapter = GroupAdapter<GroupViewHolder>()

        val myUid : String? = auth.uid
        val yourUid : String? = intent.getStringExtra( name: "yourUid")
        val name : String? = intent.getStringExtra( name: "name")

        val database : FirebaseDatabase = FirebaseDatabase.getInstance()
        val myRef : DatabaseReference = database.getReference( path: "message")
        var readRef : DatabaseReference = database.getReference( path: "message").child(myUid.toString()).child(yourUid)

```

```

    val childEventListener = object : ChildEventListener {
        override fun onCancelled(error: DatabaseError) {

        }

        override fun onChildMoved(p0: DataSnapshot, previousChildName: String?) {

        }

        override fun onChildChanged(p0: DataSnapshot, previousChildName: String?) {

        }

        override fun onChildAdded(p0: DataSnapshot, previousChildName: String?) {
            Log.d(TAG, msg: "p0 : " + p0)

            val msg = p0.getValue(ChatNewModel::class.java)?.message
            Log.d(TAG, "p0 : " + msg)

            val model : ChatModel? = p0.getValue(ChatModel::class.java)
            Log.d(TAG, msg: "model:" + model)

            val msg : String = model?.message.toString()
            val who : String? = model?.who
            val yourUid : String? = model?.yourUid

            Log.d(TAG, msg: "msg : " + msg)

```

```

        if(who == "me") {
            adapter.add(ChatRightMe(msg))
        } else {
            db.collection( collectionPath: "users")
                .get()
                .addOnSuccessListener { result ->
                    for(document : QueryDocumentSnapshot! in result) {
                        // 상대방 이름 출력
                        if(document.get("uid") == yourUid) {
                            adapter.add(ChatLeftYou(msg, document.get("username").toString()))
                            return@addOnSuccessListener
                        }
                    }
                }
                .addOnFailureListener { exception ->
                    Log.d(TAG, msg: "Error getting documents", exception)
                }
        }
    }

    override fun onChildRemoved(p0: DataSnapshot) {

    }
}

recyclerview_chat_room.adapter = adapter
readRef.addChildEventListener(childEventListener)

```

```

//      val myRef_list = database.getReference("message-user-list")

button_chat_room.setOnClickListener { it: View!

    val message :String  = editText_chat_room.text.toString()

    val chat = ChatNewModel(myUid.toString(), yourUid, message, System.currentTimeMillis(), who: "me")
    myRef.child(myUid.toString()).child(yourUid).push().setValue(chat)

    val chat_get = ChatModel(yourUid, myUid.toString(), message, System.currentTimeMillis(), who: "you")
    myRef.child(yourUid.toString()).child(myUid.toString()).push().setValue(chat_get)

//      myRef_list.child(myUid.toString()).child(yourUid).setValue(chat)

    editText_chat_room.setText("")

}
}

```


2. 나의 앱 만들기 후기

이름: 황승환

	<p>나의 소감</p> <p>이번 학기에 졸업프로젝트를 진행하면서 모바일로 작품을 준비하는 다른 팀들을 보면서 모바일 앱을 만들고 싶다는 생각을 계속 하고 있었는데, 이번 모바일 프로그래밍 수업에서 기말 프로젝트로 간단하지만 저만의 앱을 구축해볼 수 있어서 정말 좋은 경험이었다고 생각합니다.</p> <p>제한된 시간에 처음 접해보는 코틀린으로 첫 모바일 프로젝트를 진행했기에 많은 어려움이 있어 처음 기획했던 GPS 위치 기반으로 사용자 리스트를 가져와 채팅을 하는 기능을 구현하지 못해서 아쉽지만 가장 구현해보고 싶었던 채팅 기능을 간단하게라도 만들어내는데 성공했다는 점에서는 만족할 만한 성과라고 생각합니다.</p> <p>비록 이번 프로젝트는 끝이 났지만, 차후 부족한 점을 보완하여 처음 기획했던 위치 기반 채팅 서비스를 만들어보고 싶고, 파이어베이스를 이용하는 것이 아니라 이번 졸업프로젝트를 통해 만든 웹 서버에 모바일을 연동하여 서비스를 확장 시켜보고 싶습니다.</p>
	<p>모바일 프로그래밍 강의를 통해 얻은 것</p> <p>첫번째로 팀장으로서 리더십을 기를 수 있는 계기가 됐습니다. 학교생활을 하면서 팀 프로젝트가 있을 때마다 대부분 팀원의 역할을 맡았습니다. 그러나 이번에는 팀장의 역할을 맡게 되었는데 처음에는 많이 어색하고 어려웠지만 팀원이 적극적으로 잘 따라와준 덕분에 저도 금방 적응하여 최선을 다 할 수 있었던 것 같습니다. 비록 짧은 시간이지만 리더로서 팀원과 어떻게 의사소통을 해야 하는지 배울 수 있는 계기가 됐던 수업인 것 같습니다.</p> <p>두번째로 코틀린과 파이어베이스 사용법에 대해 알게 된 것입니다. 모바일 프로그래밍 강의를 들으면서 처음엔 코틀린이 너무 어렵게 느껴졌습니다. 그러나 강의 자료와 블로그 자료들을 보면서 공부하고 프로젝트를 진행해가면서 코틀린의 단순한 문법, 간결한 코드, 람다식의 사용 등의 장점들이 보이기 시작했고 코틀린에 조금씩 익숙해졌습니다. 또한 프로젝트를 진행하면서 파이어베이스 Auth, Storage, Cloud Firestore, Realtime Database 등 다양한 기술들을 접해볼 수 있는 계기가 됐는데 이 경험은 앞으로 모바일 프로젝트를 진행할 때 굉장한 도움이 될 것이라고 생각합니다.</p> <p>세번째로 모바일 영역에 관심을 가지게 됐습니다. 모바일 프로그래밍 강의를 듣기전에는 PHP나 스프링부트를 이용한 웹 개발에만 관심이 있었습니다. 그러나 이번 강의 덕분에 모바일 영역에 관심을 가지게 됐고, 자바나 코틀린을 이용한 네이티브 앱 개발 뿐만 아니라 플러터, 아이오닉, 리액트 네이티브 등을 이용한 하이브리드 앱 개발에도 관심이 생겼습니다.</p> <p>마지막으로, 오류를 만나거나 발견했을 때에 해당 오류 메시지를 읽는 습관을 들였다는 점입니다. 이전에는 코드를 작성하며 오류가 발생하면 그 오류가 무엇을 의미하는지 읽지도 않은 채 구글에 그대로 검색해보는 좋지 않은 습관을 가지고 있었습니다. 그 결과, 오류에 대한 근본적인 원인을 알지 못하고 똑 같은 실수를 반복하는 불상사를 일으켰습니다. 그러나, 이번 프로젝트를 통해 이런 안 좋은 습관을 고칠 수 있게 됐습니다.</p>

이름: 표승훈

	<p>나의 소감</p> <p>일단 앱이라는 것을 만드는 것 자체에 대한 생소함이 가장 컸던 것 같습니다. 평소에 해보았던 웹프로그래밍 기술과는 다른 부분들이 많아서 접근하기가 조금 어려웠습니다. 특히 대중적인 언어인 java를 사용해서 안드로이드 프로그래밍을 해놓은 자료들이 대부분이었기에 kotlin을 사용해서 작업을 하는 것은 자료를 찾고 공부하는 것에 있어 어려움이 있었던 것은 사실입니다.</p> <p>그럼에도 불구하고 이렇게 길지 않은 시간에 자신이 필요한 것을 배워 하나의 구동하는 앱을 만들어 보는 것은 저 자신에게 뜻 깊은 경험이 되었습니다. 평소에 관심이 많은 주제에 대해서 알릴 수 있는 그런 부족하지만 처음에 그렸던 그림을 어느정도 표현해 낼 수 있어서 노력이 헛되지 않았다고 생각합니다.</p>
	<p>모바일 프로그래밍 강의를 통해 얻은 것</p> <p>강아지 SNS 채팅 앱이라는 것을 처음 배우는 kotlin이라는 언어로 도전하여 만들어보는 데는 모바일 프로그래밍 강의를 듣지 않았더라면 이런 시도를 과연 할 수 있었을까.. 생각이 듭니다. 그 만큼 쉽지 않았고 시간도 생각보다 타이트하게 느껴졌기에 좀더 집중할 수 있는 계기가 되었습니다. 사실은 좀더 구체적이고 GPS를 이용한 기술을 사용하고 싶었는데 그 부분까지는 해내지 못한 것이 아쉬운 점입니다. 이러한 부분은 강의가 끝나고서도 스스로 프로젝트를 하면서 배웠던 경험을 생각하며 더욱 발전할 수 있을 거라고 생각합니다.</p> <p>평소에 수동적이고 발전에 대한 큰 계기가 없었던 상황에서 이러한 도전 거리를 안겨주신 교수님의 강의에 감사하고 이 경험들이 단지 모바일 프로그래밍뿐 아니라 저의 개발자 생활에 도움이 될 거라고 확신합니다.</p>