스마트앱 프로그래밍 보고서

- 프로젝트 명 : HyunTalk

김정현

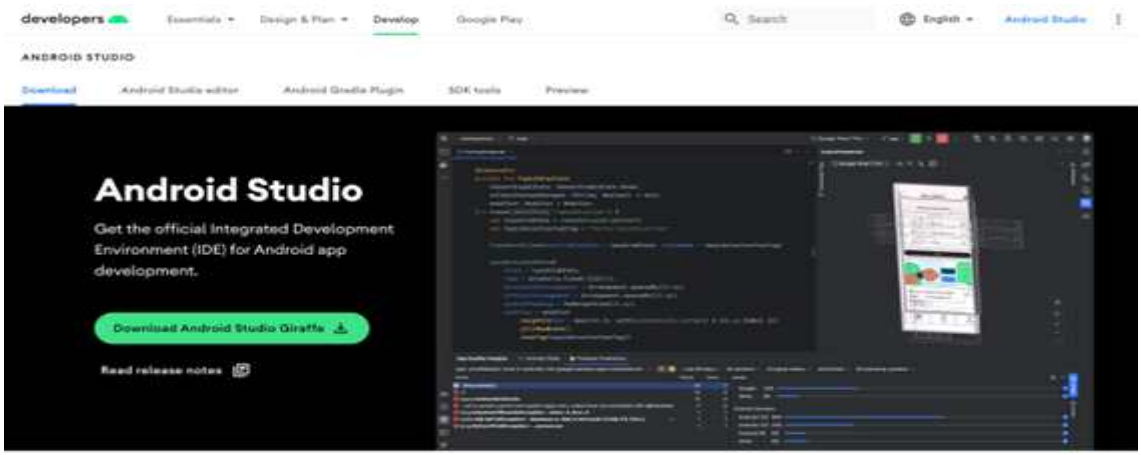
- 안드로이드 개발 준비(개발환경설정)

- 개발환경설정

1. Android Studio 최신버전 다운로드 및 설치
2. Android SDK 설치
3. Android Virtual Device(AVD) 생성
4. Figma를 이용하여 앱 화면 와이어프레임 작성
5. HyunTalk App 작성

- Android Studio 최신버전 다운로드 및 설치

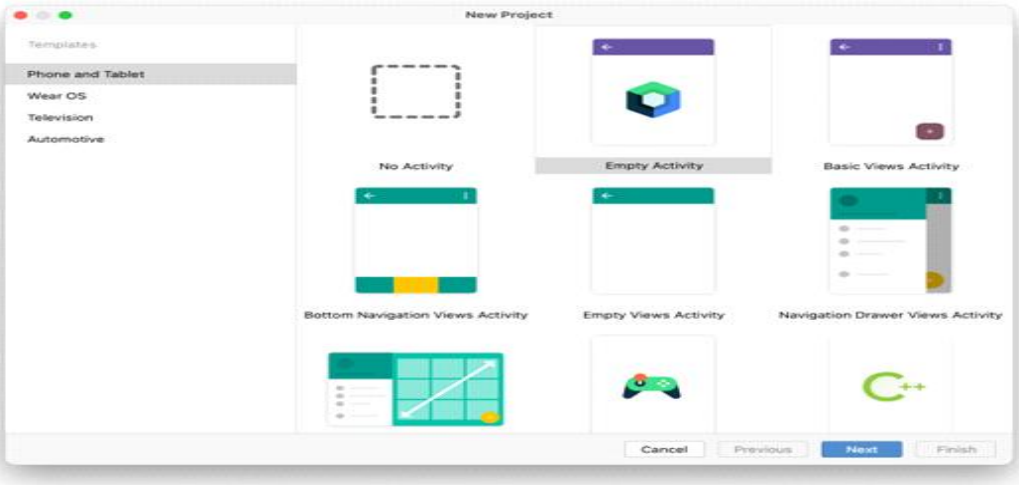
https://developer.android.com/studio 웹사이트 방문 후 운영체제에 적합한 프로그램 다운로드



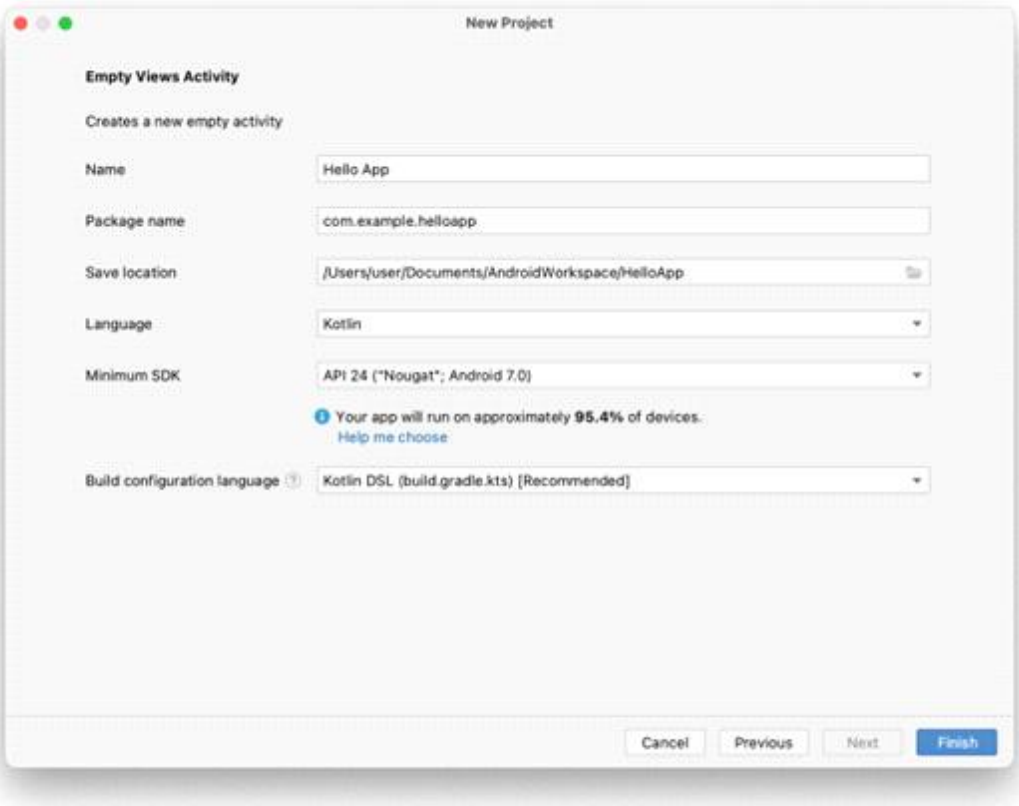
간단한 설치 후 안드로이드 스튜디오를 처음 실행하면 초기 설정을 진행한다.

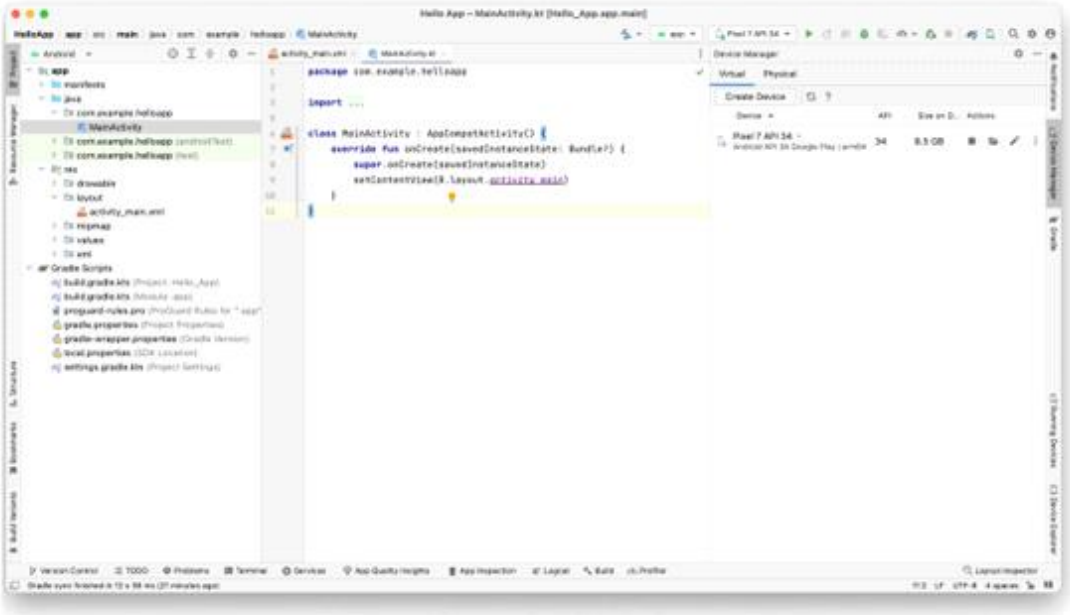
- Android Studio 실행 및 프로젝트 생성

안드로이드 스튜디오 실행 후 Phone and Tablet 선택하고 템플릿들 중에서 Empty Activity 또는 Empty View Activity 템플릿을 선택한다. Empty Activity 템플릿은 Jetpack Compose 기능이 포함된 Compose App을 생성하는 것으로써 Compose 기반의 UI 디자인을 수행하기 위해서는 이 템플릿을 선택한다. 이 템플릿은 하나의 액티비티 화면에 “Hello Android!”를 출력하는 코드를 가지고 있다.
Empty View Activity 템플릿은 “Hello World” 텍스트를 화면에 출력하는 기존의 간단한 앱을 생성하는 템플릿으로 MainActivity.kt(또는 MainActivity.java), activity_main.xml 파일이 자동으로 생성된다.
Next를 클릭하면 프로젝트 설정 화면이 다음과 같이 표시된다.



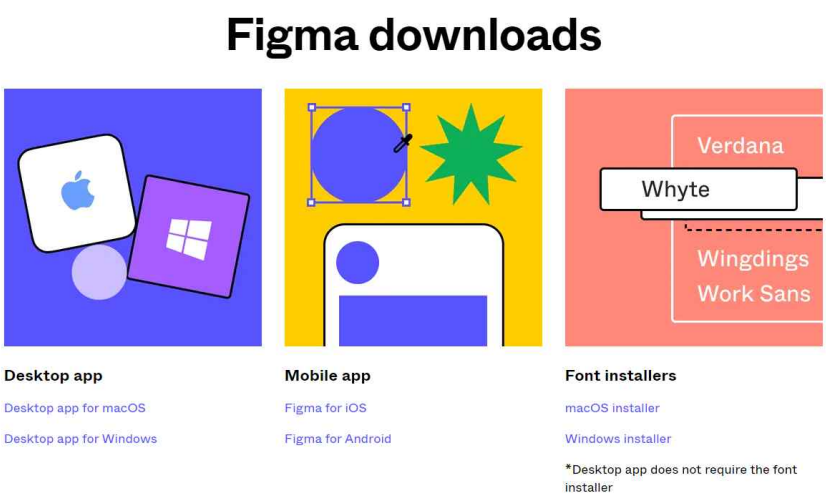
Name : 프로젝트명
Package name : 패키지명은 com.example.프로젝트명 형식으로 생성
Save location : 프로젝트 파일들이 생성되어 저장되는 위치
Minimum SDK : 생성하는 앱이 실행될 최소 안드로이드 버전을 선택
Finish를 클릭하면 안드로이드 스튜디오에 의해 프로젝트가 생성되고 빌드 된다. 이 과정에서 앱 생성과 빌드에 필요한 파일들이 자동으로 다운로드 되고 설치된다.



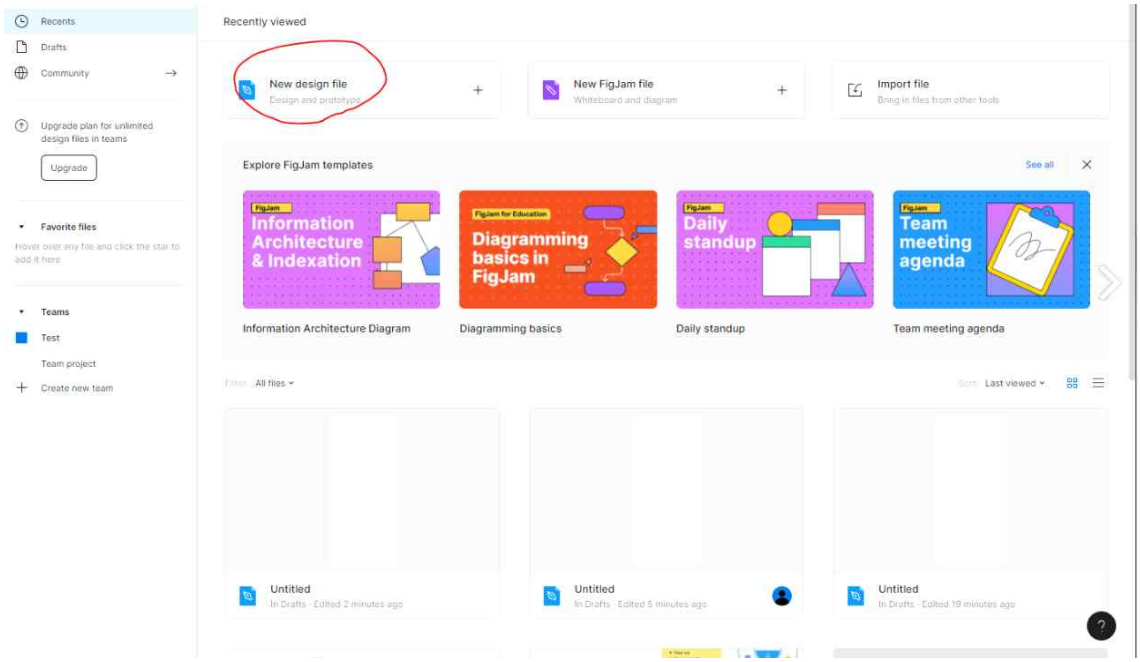


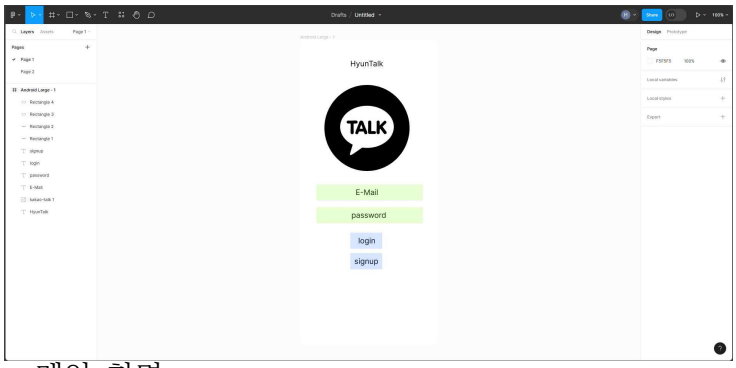
- Figma를 이용하여 초기 화면 와이어프레임 구성

- Step 1. https://www.figma.com/downloads/ 웹사이트 방문 후 운영체제에 적합한 프로그램 다운로드

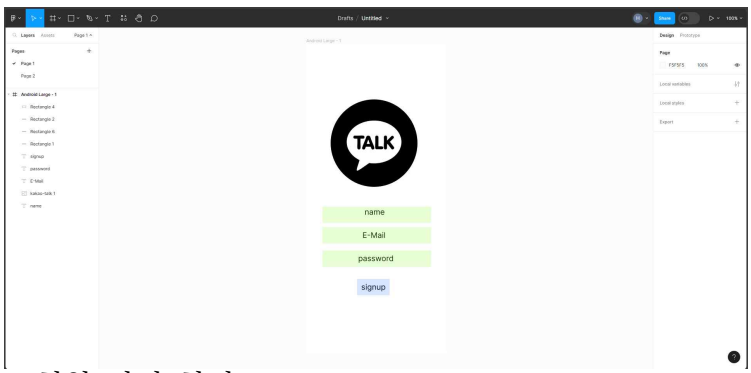


- Step 2. New design file 선택후 프레임을 Android Large를 선택후 화면 와이어 프레임 작성

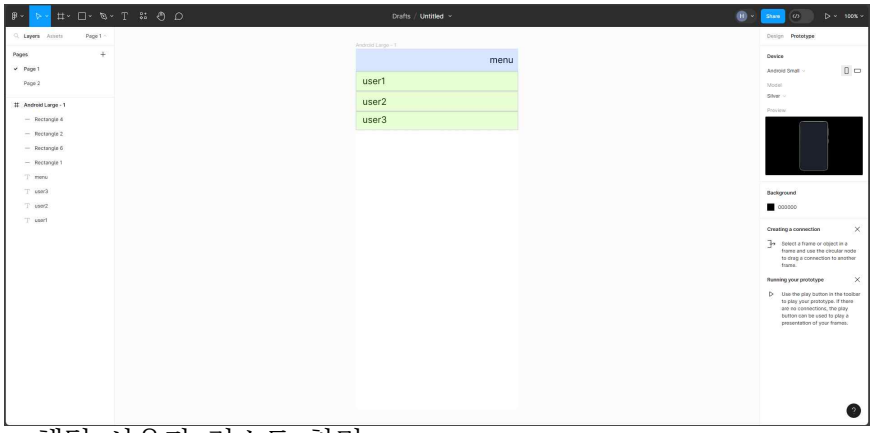




- 메인 화면



- 회원 가입 화면



- 채팅 사용자 리스트 화면

- Step 3. 오른쪽에 있는 Inspect를 눌러 Android로 변경해주면 Android의 xml 코드를 볼 수 있다.

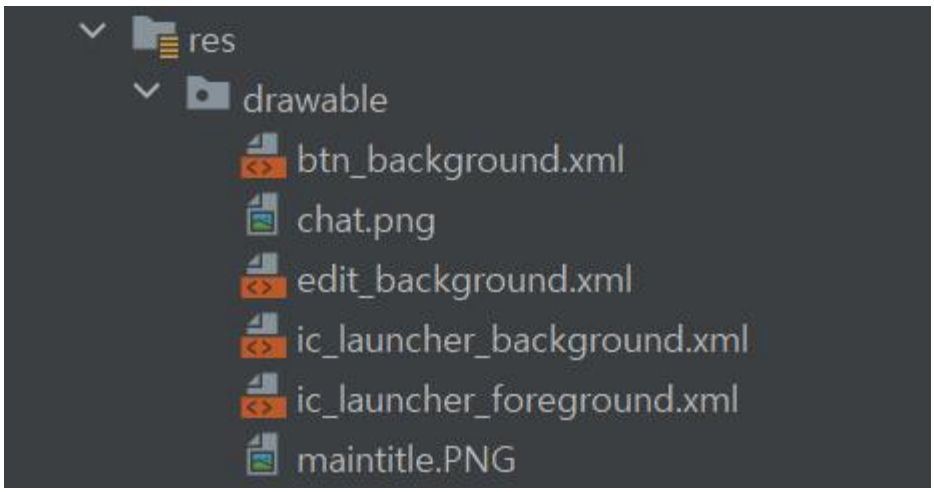
- 1. 로그인 Activity 만들기 (초기 앱 구성)

다음과 같은 입력이 가능한 로그인 기능 구현과 메인 창을 만들어보자.

- Step 1. 우선 build.gradle(Module:프로젝트명:app)에 코드를 작성한다.
이 코드는 findViewById 없이 View 객체에 접근하기 위한 설정이다.

```
            getDefaultProguardFile("proguard-android-optimize.txt"),
            "proguard-rules.pro"
        )
    }
}

compileOptions {
    sourceCompatibility = JavaVersion.VERSION_1_8
    targetCompatibility = JavaVersion.VERSION_1_8
}
kotlinOptions {
    jvmTarget = "1.8"
}

buildFeatures {
    viewBinding = true
}
}
```

- Step 2. 원하는 App 구성에 필요한 이미지 파일을 drawable 폴더에 복사한다.



여기서 chat.png, maintitle.png 는 메인 화면의 이미지 파일이고
btn_background.xml, edit_background.xml은 메시지 입력, 출력 디자인 xml 파일이다.

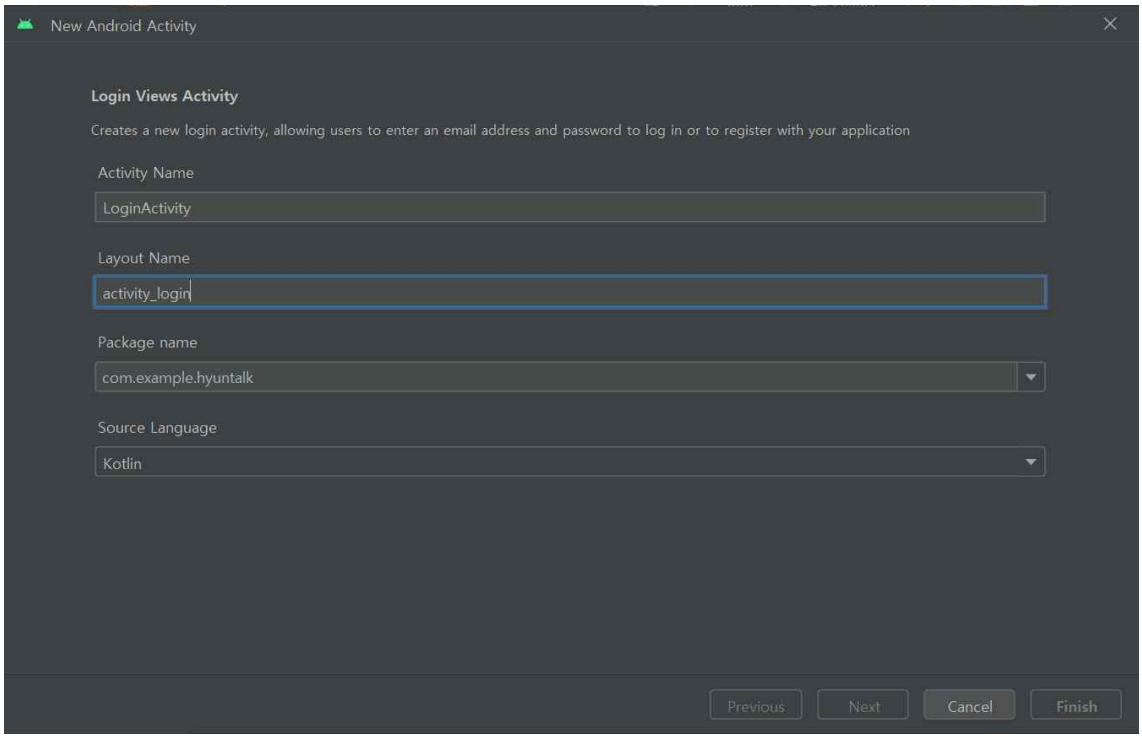
- btn_background의 코드 (내부 색상 설정)

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <corners android:radius="15dp"/>
    <solid android:color="@color/green"/>
</shape>
```

- edit_background.xml의 코드 (외곽선 둥글게, 굵기, 색상 설정)

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <corners android:radius="20dp"/>
    <stroke
        android:width="4dp"
        android:color="@color/green" />
</shape>
```

- Step 3. LoginActivity라는 이름의 액티비티를 생성한다.

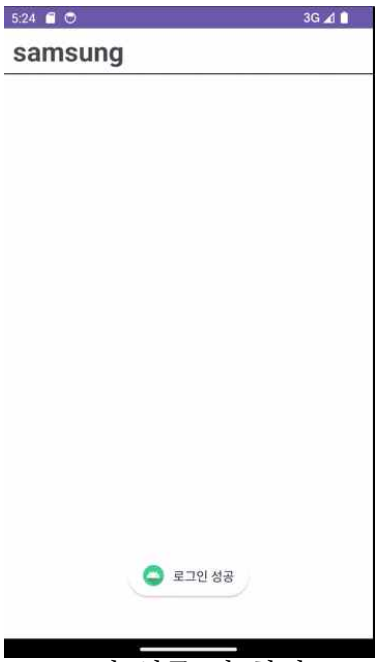


- Step 4. LoginActivity.kt 에 코드를 입력한다. (주석 및 하단 설명 참고)

```
class LoginActivity : AppCompatActivity() {

    lateinit var binding: ActivityLoginBinding

    lateinit var mAuth: FirebaseAuth

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityLoginBinding.inflate(layoutInflater)
        setContentView(binding.root)

        // 인증 초기화
        mAuth = Firebase.auth

        // login button event
        binding.loginBtn.setOnClickListener { it: View!
            val email = binding.emailEdit.text.toString()
            val password = binding.passwordEdit.text.toString()

            login(email, password)
        }

        // signup button event
        binding.signupBtn.setOnClickListener { it: View!
            val intent: Intent = Intent( packageContext: this@LoginActivity, SignupActivity::class.java)
            startActivity(intent)
        }
    }
    // login
    private fun login(email: String, password: String){
        mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(this) {
            task -> if (task.isSuccessful){
                val intent: Intent = Intent( packageContext: this@LoginActivity, MainActivity::class.java)
            startActivity(intent)
            Toast.makeText( context: this,  text: "로그인 성공", Toast.LENGTH_SHORT).show()
            finish()
        } else {
            Toast.makeText( context: this,  text: "로그인 실패", Toast.LENGTH_SHORT).show()
            Log.d( tag: "Login",  msg: "Error: ${task.exception}")
        }
        }
    }
}
```

1. binding : ActivityLoginBinding으로 뷰 바인딩 객체를 생성한다.
2. binding = ActivityLoginBinding.inflate(layoutInflater)로 뷰 바인딩 객체를 초기화 한다.
3. 로그인기능 함수를 생성하여 이메일과 비밀번호를 입력받는다.
4. 로그인과 회원가입 버튼 이벤트로 각각의 다음 Activity로 넘어가는 버튼을 만든다.
5. intent로 회원가입 Activity로 이동 객체를 생성하고 이동한다.

6. mAuth 인증 객체를 생성하고 초기화 한다.
7. mAuth 인증 서비스 기능을 구현하여 등록된 이메일과 패스워드를 검증한다.
8. 검증에 성공시 “로그인 성공” 메시지를 출력, 실패시 “로그인 실패” 메시지를 출력 한다.



- 로그인 성공 시 화면

- Step 5. Activity_login.xml에 다음과 같이 레이아웃을 지정해준다.

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".LoginActivity">

    <ImageView
        android:id="@+id/maintitle"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="70dp"
        android:src="@drawable/maintitle" />
```

```
<ImageView
    android:id="@+id/logo_image"
    android:layout_width="match_parent"
    android:layout_height="185dp"
    android:layout_below="@id/maintitle"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="30dp"
    android:src="@drawable/chat" />

<EditText
    android:id="@+id/email_edit"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_below="@id/logo_image"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="20dp"
    android:layout_marginRight="20dp"
    android:background="@drawable/edit_background"
    android:hint="Email"
    android:inputType="textEmailAddress"
    android:paddingStart="15dp"/>

<EditText
    android:id="@+id/password_edit"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_below="@id/email_edit"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="10dp"
    android:layout_marginRight="20dp"
    android:background="@drawable/edit_background"
    android:hint="password"
    android:inputType="numberPassword"
    android:paddingStart="15dp"/>
```

```
    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/login_btn"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/password_edit"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp"
        android:background="@drawable/btn_background"
        android:text="로그인"
        android:textColor="@android:color/white"
        android:textSize="20dp" />

    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/signup_btn"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/login_btn"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp"
        android:background="@drawable/btn_background"
        android:text="회원 가입"
        android:textColor="@android:color/white"
        android:textSize="20dp" />
</RelativeLayout>
```

- Step 6. AndroidManifest.xml 홈 화면 변경

MainActivity에 있는 intent-filter를 LoginActivity로 옮겨준다.

이렇게 하면 앱 실행 시 LoginActivity가 홈 화면으로 보인다.

```
        <activity
            android:name=".LoginActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            />
        </activity>
        <activity
            android:name=".MainActivity"
            android:exported="true" />
```

- 2. 회원가입 Activity 만들기

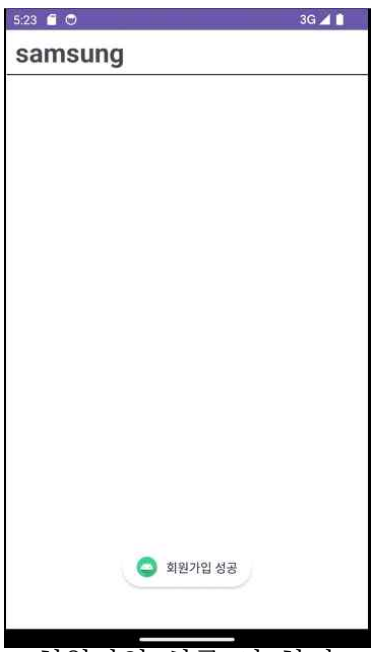
다음과 같은 입력이 가능한 회원가입 액티비티를 만들어보자.

- Step 1. 1단계의 LoginActivity와 동일하게 SignupActivity라는 이름의 액티비티를 생성하고 필요한 코드를 작성한다. (주석 및 하단 설명 참고)

```
class SignupActivity : AppCompatActivity() {

    lateinit var binding: ActivitySignupBinding

    lateinit var  mAuth: FirebaseAuth

    private lateinit var mDbRef: DatabaseReference

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivitySignupBinding.inflate(layoutInflater)
        setContentView(binding.root)

        //인증 초기화
        mAuth = Firebase.auth

        //
        mDbRef = Firebase.database.reference

        binding.signupBtn.setOnClickListener {
            val name = binding.nameEdit.text.toString().trim()
            val email = binding.emailEdit.text.toString().trim()
            val password = binding.passwordEdit.text.toString().trim()
```

```
            signup(name, email, password)
        }
    }

    // signup
    private fun signup(name: String, email:String, password:String) {
        mAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(this) {
            task -> if(task.isSuccessful) {
                Toast.makeText( context: this,  text: "회원가입 성공", Toast.LENGTH_SHORT).show()
            val intent: Intent = Intent( packageContext: this@SignupActivity, MainActivity::class.java)
            startActivity(intent)
            addUserToDatabase(name, email, mAuth.currentUser?.uid!!)
        } else {
            Toast.makeText( context: this,  text: "회원가입 실패", Toast.LENGTH_SHORT).show()
            Log.d( tag: "SignUp",  msg: "Error: ${task.exception}")
        }
        }
    }
    private fun addUserToDatabase(name: String, email: String, uId: String){
        mDbRef.child( pathString: "user").child(uId).setValue(User(name, email, uId))
    }
}
```

1. mAuth:FirebaseAuth로 인증 서비스 객체를 생성한다.
2. mAuth = Firebase.auth로 인증 서비스를 초기화한다.
3. signup이라는 이름, 이메일, 비밀번호를 입력받는 회원가입 등록 함수를 만들어준다.
4. 회원가입에 성공하면 회원가입 성공 메시지와 동시에 인증 서비스 신규 계정 등록을 해준다.
5. 중복된 이메일이거나 비밀번호가 6자리 이하시 회원가입 실패가 된다.
6. addUserToDatabase 함수로 데이터베이스에 저장한다. (이름, 이메일, uid(인증 데이터에 저장된 정보))
7. mDbRef.child("user").child(uId).setValue(User(name,email,uId))에서 user안에 uid안에 사용자 정보를 저장한다.



- 회원가입 성공 시 화면

- Step 2. DB에 저장할 사용자 정보를 담을 모델 클래스 파일(User.kt)을 생성한다.

```kotlin
package com.example.hyuntalk

data class User(
    var name: String,
    var email: String,
    var uId: String
){
    constructor(): this( name: "", email: "", uId: "")
}
```

- Realtime Database에 저장된 사용자 정보

- Step 3. Activity_signup.xml에 다음과 같이 레이아웃을 지정해준다.

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SignupActivity">

    <ImageView
        android:id="@+id/logo_image"
        android:layout_width="wrap_content"
        android:layout_height="350dp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="70dp"
        android:src="@drawable/chat" />
```

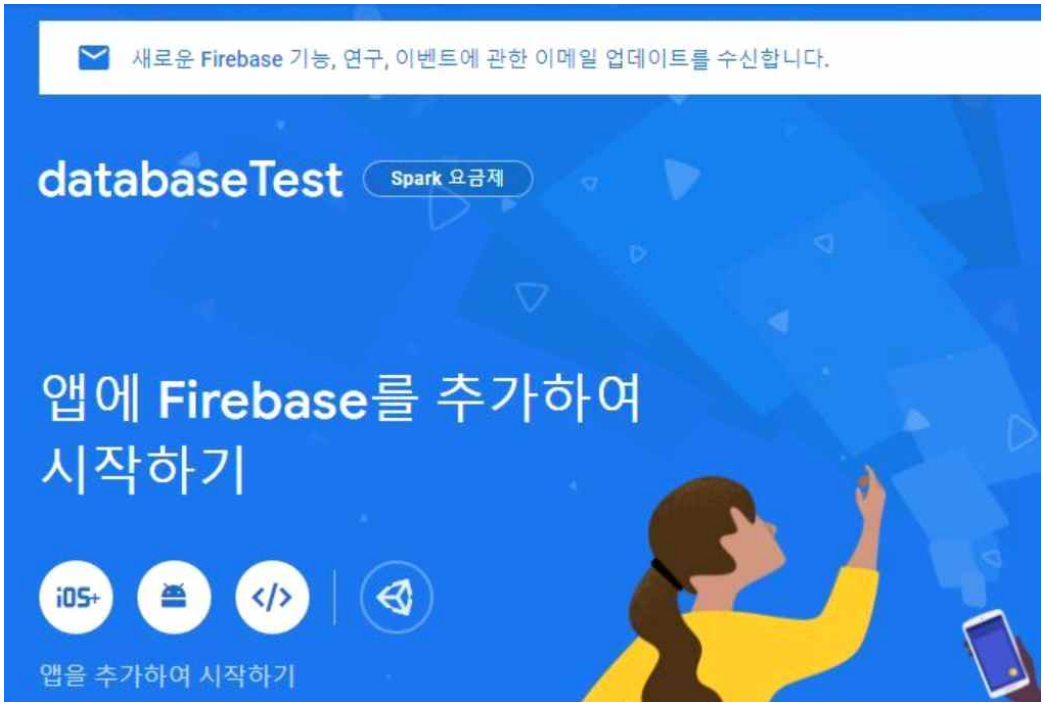
```
    <EditText
        android:id="@+id/name_edit"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_below="@id/logo_image"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="20dp"
        android:layout_marginRight="20dp"
        android:background="@drawable/edit_background"
        android:hint="Name"
        android:inputType="textEmailAddress"
        android:paddingStart="15dp" />
    <EditText
        android:id="@+id/email_edit"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_below="@id/name_edit"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="10dp"
        android:layout_marginRight="20dp"
        android:background="@drawable/edit_background"
        android:hint="Email"
        android:inputType="textEmailAddress"
        android:paddingStart="15dp" />

    <EditText
        android:id="@+id/password_edit"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_below="@id/email_edit"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="10dp"
        android:layout_marginRight="20dp"
        android:background="@drawable/edit_background"
        android:hint="password"
        android:inputType="numberPassword"
        android:paddingStart="15dp" />
```

```
    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/signup_btn"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/password_edit"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp"
        android:background="@drawable/btn_background"
        android:text="회원 가입"
        android:textColor="@android:color/white"
        android:textSize="20sp" />

</RelativeLayout>
```

- 3. Firebase와 연동하기

- Step 1. 로그인과 회원가입을 이용하기위해 Firebase의 인증 서비스를 이용한다. 우선 http://firebase.google.com/ 으로 접속하여 프로젝트를 만든다.

- Step 2. 만들어진 프로젝트를 누르고 안드로이드 앱 추가를 하고 초기설정을 한다.



- Step 3. google-services.json을 다운로드하여 app폴더 안에 넣고 build.gradle (프로젝트명) plugins와 build.gradle(module: 프로젝트명.app)안에 코드를 입력한다.

```
buildscript {
    dependencies {
        classpath("com.google.gms:google-services:4.4.0")
    }
}
// Top-level build file where you can add configuration options common to all sub-projects/modules.
plugins {
    id("com.android.application") version "8.1.4" apply false
    id("org.jetbrains.kotlin.android") version "1.9.0" apply false
    id("com.google.gms.google-services") version "4.4.0" apply false
}
```

- build.gradle(프로젝트명) plugins

```
plugins { this: PluginDependenciesSpecScope
    id("com.android.application")
    id("org.jetbrains.kotlin.android")
    id("com.google.gms.google-services")
}

android { this: BaseAppModuleExtension
    namespace = "com.example.hyuntalk"
    compileSdk = 34

    defaultConfig { this: ApplicationDefaultConfig
        applicationId = "com.example.hyuntalk"
    buildFeatures { this: ApplicationBuildFeatures
        viewBinding = true
    }
}
dependencies { this: DependencyHandlerScope

    implementation("androidx.core:core-ktx:1.9.0")
    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.10.0")
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")
    implementation(platform("com.google.firebase:firebase-bom:32.6.0"))
    implementation("com.google.firebase:firebase-analytics")
    implementation("com.google.firebase:firebase-auth-ktx:22.3.0")
    implementation("com.google.android.gms:play-services-auth:20.7.0")
    implementation("com.google.firebase:firebase-database:20.3.0")
    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
}
```

- build.gradle(module: 프로젝트명.app) [인증 서비스 라이브러리도 같이 추가]
이후 콘솔로 이동버튼을 누르면 앱에 Firebase가 추가 된다.

- Step 4. 사용자 데이터 저장을 위해 Realtime Database를 생성하고 초기 설정을 한다.

- 4. 사용자 리스트 화면 생성하기

다음과 같이 회원 가입하여 저장된 채팅 사용자 리스트를 보여주는 액티비티를 만들어보자.

- Step 1. 사용자 이름을 보여주는 TextView와 사용자를 구분해주는 선 View가 담긴 user_layout.xml를 생성한다.

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/name_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="5dp"
        android:text="user1"
        android:textSize="30sp"
        android:textStyle="bold" />

    <View
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:layout_below="@+id/name_text"
        android:layout_marginTop="5dp"
        android:background="@android:color/black" />
</RelativeLayout>
```

- Step 2. 사용자 리스트 어댑터(UserAdapter.kt)를 생성한다.

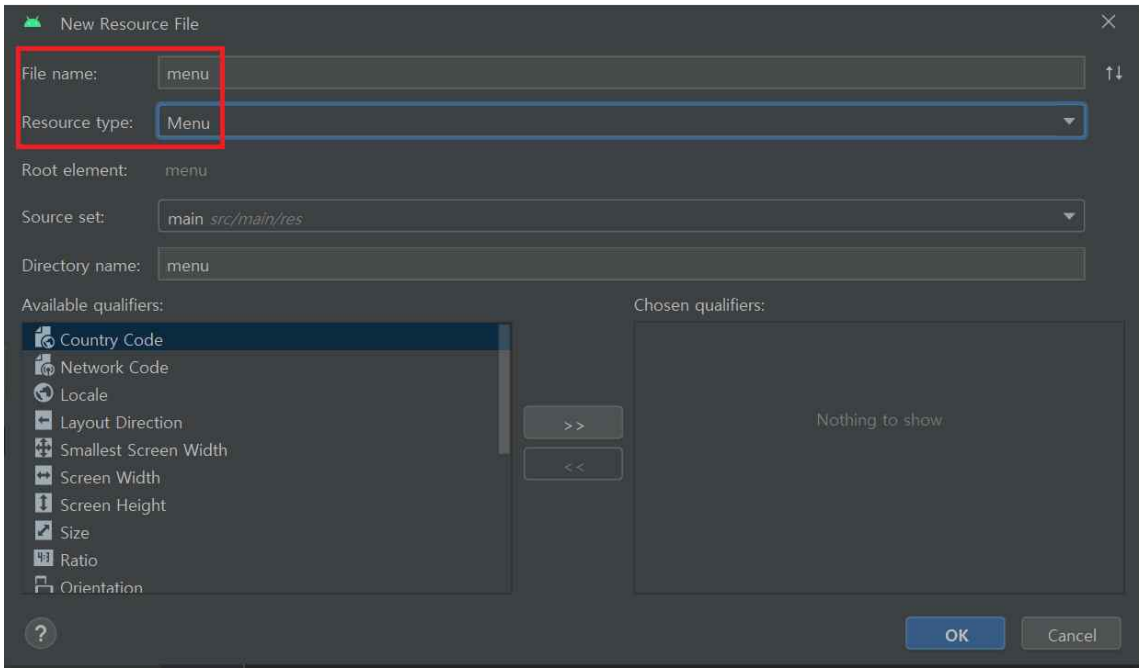
```kotlin
class UserAdapter(private val context: Context,private val userList: ArrayList<User>):
    RecyclerView.Adapter<UserAdapter.UserViewHolder>(){

    // 화면설정
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): UserAdapter.UserViewHolder {
        val view: View = LayoutInflater.from(context).inflate(R.layout.user_layout, parent, false)

        return UserViewHolder(view)
    }

    // 데이터 설정
    override fun onBindViewHolder(holder: UserAdapter.UserViewHolder, position: Int) {
        val currentUser = userList[position]
        holder.nameText.text = currentUser.name

        holder.itemView.setOnClickListener {
            val intent = Intent(context, ChatActivity::class.java)

            // 넘길 데이터
            intent.putExtra("name",currentUser.name)
            intent.putExtra("uId",currentUser.uId)
            context.startActivity(intent)
        }
    }

    override fun getItemCount(): Int {
        return userList.size
    }
    class UserViewHolder(itemView: View): RecyclerView.ViewHolder(itemView){

        val nameText: TextView = itemView.findViewById(R.id.name_text)
    }
}
```

1. 유저 생성 시에 Context와 ArrayList를 넘겨받는다.
2. onCreateViewHolder로 화면을 연결한다.
3. onBindViewHolder로 데이터를 연결한다.
4. getItemCount로 데이터 개수를 돌려준다.

- Step 3. 사용자 리스트를 보여주는 RecyclerView를 activity_main.xml에 생성한다.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/user_recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:listitem="@layout/user_layout" />

</LinearLayout>
```

- Step 4. 로그아웃 기능을 구현하기 위해 메뉴를 생성한다.



- Step 5. menu.xml에 로그아웃 기능 item을 생성한다.

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/log_out"
        android:title="Log Out" />
</menu>
```

- Step 6. MainActivity에 코드를 작성한다. (주석 및 하단 설명 참고)

```
class MainActivity : AppCompatActivity() {

    lateinit var binding: ActivityMainBinding
    lateinit var adapter: UserAdapter

    private lateinit var mAuth: FirebaseAuth
    private lateinit var mDbRef: DatabaseReference

    private lateinit var userList: ArrayList<User>

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        // 인증 초기화
        mAuth = Firebase.auth

        // db 초기화
        mDbRef = Firebase.database.reference

        // List 초기화
        userList = ArrayList()

        adapter = UserAdapter(this, userList)
        binding.userRecyclerView.layoutManager = LinearLayoutManager( context: this)
        binding.userRecyclerView.adapter = adapter

        // 사용자 정보 가져오기
        mDbRef.child( pathString: "user").addValueEventListener(object:ValueEventListener{
            override fun onDataChange(snapshot: DataSnapshot) {
                for(postSnapshot in snapshot.children){
                    // 유저 정보
                    val currentUser = postSnapshot.getValue(User::class.java)

                    if(mAuth.currentUser?.uid != currentUser?.uId){
                        userList.add(currentUser!!)
                    }
                }
                adapter.notifyDataSetChanged()
            }
            override fun onCancelled(error: DatabaseError){
                // 실패시 실행
            }
        })
    }//onCreate

    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        menuInflater.inflate(R.menu.menu, menu)
        return super.onCreateOptionsMenu(menu)
```

```
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        if(item.itemId == R.id.log_out){
            mAuth.signOut()
            val intent = Intent( packageContext: this@MainActivity, LoginActivity::class.java)
            startActivity(intent)
            finish()
            return true
        }
        return true
    }
}
```

1. mAuth = Firebase.auth로 인증 서비스 객체를 초기화한다.
2. mDbRef = Firebase.database.reference로 실시간 데이터베이스 객체를 초기화한다.
3. addValueEventListener로 해당 경로의 데이터를 가져온다.
4. 사용자 정보를 가져오고 실패 시 실행하는 onCancelled를 추가한다.
5. onCreateOptionsMenu로 생성할 메뉴 지정 함수를 만들어주고 onOptionsItemSelected로 메뉴 아이템 선택 기능 구현 함수를 만들어 준다.
6. 메뉴에는 인증 서비스 로그아웃이 구현되어 있다.

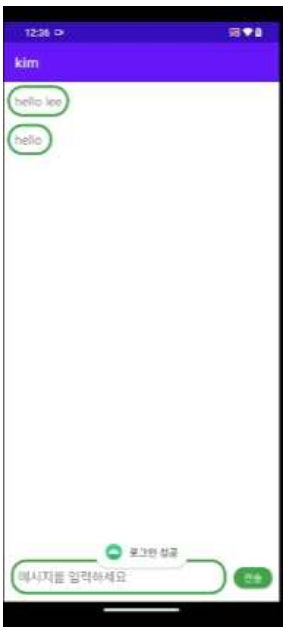
- 메뉴 버튼의 로그아웃 기능 화면

- Step 7. 사용자 리스트 어댑터를 생성한다.

```
class UserAdapter(private val context: Context,private val userList: ArrayList<User>):
    RecyclerView.Adapter<UserAdapter.UserViewHolder>(){

    // 화면설정
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): UserAdapter.UserViewHolder {
        val view: View = LayoutInflater.from(context).inflate(R.layout.user_layout, parent,
            attachToRoot: false)

        return UserViewHolder(view)
    }

    // 데이터 설정
    override fun onBindViewHolder(holder: UserAdapter.UserViewHolder, position: Int) {
        val currentUser = userList[position]
        holder.nameText.text = currentUser.name

        holder.itemView.setOnClickListener { it: View!
            val intent = Intent(context, ChatActivity::class.java)

            // 넘길 데이터
            intent.putExtra( name: "name",currentUser.name)
            intent.putExtra( name: "uId",currentUser.uId)

            context.startActivity(intent)
        }
    }

    override fun getItemCount(): Int {
        return userList.size
    }
    class UserViewHolder(itemView: View): RecyclerView.ViewHolder(itemView){

        val nameText: TextView = itemView.findViewById(R.id.name_text)
    }
}
```

1. 유저 리스트의 데이터를 담는다.
2. 화면에 받은 데이터를 보여준다.
3. 아이템 클릭 이벤트를 생성하고 넘겨줄 상대방 이름과 uId를 담는다.

- 5. 채팅 화면 만들기



- 보낸 사람 채팅 화면 - 받는 사람 채팅 화면

다음과 같은 채팅방의 화면을 만들어보자

- Step 1. ChatActivity를 생성하고 코드를 작성한다. (주석 및 하단 설명 참고)

```
class ChatActivity : AppCompatActivity() {
    private lateinit var receiverName: String
    private lateinit var receiverUid: String

    //바인딩 객체
    private lateinit var binding: ActivityChatBinding

    lateinit var mAuth: FirebaseAuth //인증 객체
    lateinit var mDbRef: DatabaseReference //DB 객체

    private lateinit var receiverRoom: String //받는 대화방
    private lateinit var senderRoom: String //보낸 대화방

    private lateinit var messageList: ArrayList<Message>

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityChatBinding.inflate(layoutInflater)
        setContentView(binding.root)
```

```
    messageList = ArrayList()
    val messageAdapter: MessageAdapter = MessageAdapter( context: this, messageList)

    binding.chatRecyclerView.layoutManager = LinearLayoutManager( context: this)
    binding.chatRecyclerView.adapter = messageAdapter

    // 넘어온 데이터 변수에 담기
    receiverName = intent.getStringExtra( name: "name").toString()
    receiverUid = intent.getStringExtra( name: "uId").toString()

    mAuth = FirebaseAuth.getInstance()
    mDbRef = FirebaseDatabase.getInstance().reference

    //접속자 uId
    val senderUid = mAuth.currentUser?.uid

    //보낸사람 방
    senderRoom = receiverUid + senderUid

    //받는사람 방
    receiverRoom = senderUid + receiverUid

    // 액션바에 상대방 이름 보여주기
    supportActionBar?.title = receiverName
binding.sendBtn.setOnClickListener { it: View!                    ▲2 ▲2 ✓5 ^
    val message = binding.messageEdit.text.toString()
    val messageObject = Message(message, senderUid)

    // data 저장
    mDbRef.child( pathString: "chats").child(senderRoom).child( pathString: "message").push()
        .setValue(messageObject).addOnSuccessListener { it: Void!
        //저장 성공시
            mDbRef.child( pathString: "chats").child(receiverRoom).child( pathString: "message").push()
                .setValue(messageObject)
    }
    // 입력값 초기화
    binding.messageEdit.setText("")
}
// 메시지 가져오기
mDbRef.child( pathString: "chats").child(senderRoom).child( pathString: "messages")
    .addValueEventListener(object: ValueEventListener{
        override fun onDataChange(snapshot: DataSnapshot) {
            messageList.clear()

            for(postSnapshot in snapshot.children){
                val message = postSnapshot.getValue(Message::class.java)
                messageList.add(message!!)
            }
```

```
                    //적용
                    messageAdapter.notifyDataSetChanged()
                }

                override fun onCancelled(error: DatabaseError) {
                }
            })
    }
}
```

1. intent를 이용해 넘어온 상대방 이름과 uId를 데이터 변수에 저장한다.
2. 액션 바에 상대방 이름을 보여준다.
3. 보낸 사람 대화 방의 값을 받는 사람 uId와 보낸 사람 uId로 합쳐서 구현한다. (반대의 경우도 동일)
4. chat -> senderRoom -> message -> 메시지 저장
5. 데이터 추가 리스너를 생성하고 onDataChange 함수를 통해 데이터 변화를 감지한다.

- Step 2. Activity_chat.xml에 다음과 같이 레이아웃을 지정해준다.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ChatActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/chat_recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@+id/input_layout"
        android:layout_alignParentTop="true" />
```

```
    <LinearLayout
        android:id="@+id/input_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="10dp"
        android:orientation="horizontal"
        android:weightSum="100">

        <EditText
            android:id="@+id/message_edit"
            android:layout_width="wrap_content"
            android:layout_height="50dp"
            android:layout_marginStart="10dp"
            android:layout_weight="85"
            android:background="@drawable/edit_background"
            android:hint="메세지를 입력하세요"
            android:paddingStart="10dp" />
        <androidx.appcompat.widget.AppCompatButton
            android:id="@+id/send_btn"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_marginStart="10dp"
            android:layout_marginEnd="10dp"
            android:layout_weight="15"
            android:background="@drawable/btn_background"
            android:text="전송"
            android:textColor="@android:color/white" />
    </LinearLayout>

</RelativeLayout>
```

- 대화 내용을 보여줄 RecyclerView, 메시지를 입력하는 EditText, 메시지를 전송하는 Button을 생성한다.

- Step 3. 메시지 정보를 담을 클래스 파일을 생성한다.

```
package com.example.hyuntalk

data class Message(
    var message: String?,
    var sendId: String?
){
    constructor():this("","")
}
```

대화 내용과 접속자 uId 정보를 저장한다.

- Step 4. 받는 메시지 화면과 보낸 메시지 화면을 구성한다.

- 받은 메시지를 보여주는 TextView(receive.xml)

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/receive_message_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_margin="5dp"
        android:background="@drawable/edit_background"
        android:padding="10dp"
        android:text="받는 메시지"
        android:textSize="18sp" />

</RelativeLayout>
```

- 보낸 메시지를 보여주는 TextView(send.xml)

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/send_message_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_margin="5dp"
        android:background="@drawable/btn_background"
        android:padding="10dp"
        android:text="보내는 메시지"
        android:textColor="@android:color/white"
        android:textSize="18sp" />

</RelativeLayout>
```

- Step 5. 메시지 어댑터를 생성한다.

```
class MessageAdapter(private val context: Context, private val messageList: ArrayList<Message>):
RecyclerView.Adapter<RecyclerView.ViewHolder>(){
    private val receive = 1 //받는 타입
    private val send = 2 //보내는 타입

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder {
        return if(viewType == 1){ //받는 화면
            val view: View = LayoutInflater.from(context).inflate(R.layout.receive, parent,
                attachToRoot: false)
            ReceiveViewHolder(view)
        } else {
            val view: View = LayoutInflater.from(context).inflate(R.layout.send, parent,
                attachToRoot: false)
            SendViewHolder(view)
        }
    }

    override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {
        // 현재 메시지
        val currentMessage = messageList[position]

        // 보내는 데이터
        if(holder.javaClass == SendViewHolder::class.java){
            val viewHolder = holder as SendViewHolder
            viewHolder.sendMessage.text = currentMessage.message
        } else { //받는 데이터
            val viewHolder = holder as ReceiveViewHolder
            viewHolder.receiveMessage.text = currentMessage.message
        }
    }

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

    override fun getItemViewType(position: Int): Int {
        // 메시지 값
        val currentMessage = messageList[position]

        return if(FirebaseAuth.getInstance().currentUser?.uid.equals(currentMessage.sendId)){
            send
        } else {
            receive
        }
    }
    // 보낸 사람
    class SendViewHolder(itemView: View): RecyclerView.ViewHolder(itemView){
        val sendMessage: TextView = itemView.findViewById(R.id.send_message_text)
    }
```

```
    // 받는 사람
    class ReceiveViewHolder(itemView: View): RecyclerView.ViewHolder(itemView){
        val receiveMessage: TextView = itemView.findViewById(R.id.receive_message_text)
    }
}
```

1. 보낸 사람과 받는 사람의 ViewHolder를 생성한다.
2. ViewHolder 타입 리턴하는 함수인 getItemViewType를 사용한다.
3. onCreateViewHolder로 화면을 연결해준다.
4. onBindViewHolder로 데이터를 연결해준다.

- Realtime Database에 사용자간의 대화가 저장된다.