DKU 단국대학교 DANKOOK UNIVERSITY

REPORT

개발자 가이드

0. Build.gradle

Chaquopy를 이용하기 위해서는 프로젝트의 Gradle Version과 Android Gradle Plugin version을 낮출 필요가 있습니다. 기본적으로 요구하고 있는 버전은 다음과 같습니다.

Android Gradle Plugin Version 7.4.0
Android Gradle 7.5
Python 3.8
SDK 33

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
    id 'kotlin-kapt'
    id 'kotlin-android'
    id 'com.chaquo.python'
}
```

위의 코드는 플러그인을 설정하는 부분입니다. 다양한 플러그인이 프로젝트에 적용됩니다. 각 플러그인은 다음과 같은 역할을 수행합니다.

com.android.application: 안드로이드 애플리케이션 플러그인으로, 안드로이드 앱을 빌드하기 위한 기능을 제공합니다.
org.jetbrains.kotlin.android: 코틀린 안드로이드 플러그인으로, 코틀린 언어를 사용하여 안드로이드 앱을 개발하는 데 필요한 기능을 제공합니다.
kotlin-kapt: 코틀린 어노테이션 프로세서 플러그인으로, 코틀린 어노테이션 프로세서를 사용하여 코드를 생성하거나 수정하는 데 필요한 기능을 제공합니다.
kotlin-android: 안드로이드용 코틀린 플러그인으로, 안드로이드 앱을 개발하는 데 필요한 코틀린 관련 기능을 제공합니다.
com.chaquo.python: Chaquopy 플러그인으로, 안드로이드 앱에서 Python 코드를 실행하기 위한 기능을 제공합니다.

```
tasks.withType(org.jetbrains.kotlin.gradle.tasks.KotlinCompile).configureEach {
    kotlinOptions {
        jvmTarget = "1.8"
    }
}
```

위의 코드는 Kotlin 컴파일러 옵션을 설정하는 부분입니다. Kotlin 컴파일러에 대한 옵션으로 jvmTarget을 설정하여 컴파일된 코드의 대상 JVM 버전을 지정합니다. 이 경우, JVM 버전 1.8을 대상으로 컴파일됩니다.

```
android {
    // Android 관련 설정

    packagingOptions {
        // 앱 패키징 옵션 설정
        exclude 'META-INF/DEPENDENCIES'
        exclude 'META-INF/INDEX.LIST'
    }
```

위의 코드는 앱 패키징 옵션을 설정하는 부분입니다. packagingOptions 블록 안에 exclude 키워드를 사용하여 특정 파일이나 디렉토리를 패키징에서 제외할 수 있습니다. 위의 예시에서는 'META-INF/DEPENDENCIES'와 'META-INF/INDEX.LIST' 파일을 제외하도록 설정되어 있습니다. 이렇게 함으로써 앱의 크기를 줄이거나 충돌 문제를 방지할 수 있습니다.

```
sourceSets {
    main {
        python.srcDir "src/main/java/com/example/python"
    }
}
```

위의 코드는 소스셋 설정 부분입니다. sourceSets 블록 안에 main 블록을 정의하고, 그 안에 python.srcDir 속성을 설정하여 Python 소스 코드의 디렉토리를 지정합니다. 위의 예시에서는 "src/main/java/com/example/python" 디렉토리에 있는 Python 소스 코드를 사용하도록 설정되어 있습니다.
Python 3.8이 설치되어 있는 경로로 수정하여 주시면 됩니다.

```
namespace 'com.example.myapplication2'
compileSdk 33
```

위의 코드는 안드로이드 네임스페이스(namespace)와 컴파일 SDK 버전(compileSdk)을 설정하는 부분입니다. namespace는 애플리케이션의 고유한 식별자로 사용되며, compileSdk는 프로젝트를 컴파일할 때 사용할 안드로이드 SDK 버전을 지정합니다. 위의 예시에서는 'com.example.myapplication2'를 네임스페이스로 설정하고, SDK 버전 33을 컴파일에 사용하도록 설정되어 있습니다.

```
dataBinding {
    enabled = true
}
```

위의 코드는 데이터 바인딩(Data Binding)을 활성화하는 부분입니다. 데이터 바인딩은 XML 레이아웃 파일에서 뷰와 데이터를 바인딩하여 UI와 데이터를 쉽게 연결할 수 있는 안드로이드의 기능입니다. enabled 속성을 true로 설정함으로써 데이터 바인딩을 활성화합니다.

```
buildFeatures {
    viewBinding true
}
```

위의 코드는 뷰 바인딩(View Binding)을 활성화하는 부분입니다. 뷰 바인딩은 XML 레이아웃 파일에 정의된 뷰를 바인딩 클래스로 생성하여 사용할 수 있도록 하는 안드로이드의 기능입니다. viewBinding 속성을 true로 설정함으로써 뷰 바인딩을 활성화합니다.

```
defaultConfig {
    applicationId "com.example.myapplication2"
    minSdk 33
    targetSdk 33
    versionCode 1
    versionName "1.0"

    testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"

    ndk {
        abiFilters "armeabi-v7a", "x86_64"
    }
```

위의 코드는 기본 설정(defaultConfig)을 지정하는 부분입니다. applicationId는 애플리케이션의 패키지 이름을 지정하고, minSdk와 targetSdk는 애플리케이션이 지원하는 최소 SDK 버전과 타겟 SDK 버전을 지정합니다. versionCode와 versionName은 애플리케이션의 버전 관련 정보를 설정합니다. testInstrumentationRunner는 테스트를 실행할 때 사용하는 instrumentation runner를 지정합니다. ndk 블록은 Native Development Kit(NDK)를 사용하여 C/C++ 코드를 지원하는 경우 해당 ABI(애플리케이션 바이너리 인터페이스) 필터를 설정합니다.

```
// JVM Toolchain 설정
compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
}
kotlinOptions {
    jvmTarget = '1.8'
    languageVersion = "1.5"
}
```

위의 코드는 JVM 툴체인(JVM Toolchain)을 설정하는 부분입니다. compileOptions 블록에서 sourceCompatibility와 targetCompatibility를 Java 8로 설정하여 Java 8 코드를 사용할 수 있도록 합니다. kotlinOptions 블록에서는 Kotlin 컴파일러에 대한 옵션을 설정합니다. jvmTarget을 '1.8'로 설정하여 JVM 대상을 Java 8로 지정하고, languageVersion을 "1.5"로 설정하여 Kotlin 언어의 버전을 1.5로 지정합니다.

```
tasks.withType(JavaCompile) {
    sourceCompatibility = '1.8'
    targetCompatibility = '1.8'
}
```

위의 코드는 Java 컴파일 태스크의 소스와 타겟의 호환성을 Java 8로 설정하는 부분입니다. JavaCompile 태스크를 가져와서 sourceCompatibility와 targetCompatibility를 각각 '1.8'로 설정합니다.

implementation 'com.google.auth:google-auth-library-oauth2-http:0.25.0': Google OAuth2 인증을 위한 라이브러리를 추가합니다.

implementation 'com.google.api:gax:1.62.0': Google API 클라이언트를 위한 라이브러리를 추가합니다.

implementation 'androidx.core:core-ktx:1.7.0': AndroidX Core KTX 라이브러리를 추가합니다. AndroidX Core 라이브러리의 Kotlin 확장 기능을 제공합니다.

implementation 'androidx.fragment:fragment-ktx:1.4.0': AndroidX Fragment KTX 라이브러리를 추가합니다. AndroidX Fragment 라이브러리의 Kotlin 확장 기능을 제공합니다.
implementation 'com.google.android.material:material:1.4.0': Material Design 구성 요소를 포함한 Android Material 라이브러리를 추가합니다.

implementation 'me.relex:circleindicator:2.1.6': 이미지 슬라이더에 사용할 수 있는 CircleIndicator 라이브러리를 추가합니다.

androidTestImplementation 'androidx.test:core:1.5.0': Android Instrumented 테스트를 위한 AndroidX Test Core 라이브러리를 추가합니다.

implementation("org.jetbrains.kotlinx:kotlinx-serialization-json:1.3.0"): Kotlin serialization을 지원하는 kotlinx-serialization-json 라이브러리를 추가합니다.

def nav_version = "2.3.5": 네비게이션 구성 요소 버전을 지정하기 위해 변수 nav_version을 정의합니다.

implementation "androidx.navigation:navigation-fragment-ktx:$nav_version": AndroidX Navigation Fragment KTX 라이브러리를 추가합니다. 네비게이션 구성 요소를 위한 Kotlin 확장 기능을 제공합니다.

implementation "androidx.navigation:navigation-ui-ktx:$nav_version": AndroidX Navigation UI KTX 라이브러리를 추가합니다. 네비게이션 구성 요소의 UI 요소를 위한 Kotlin 확장 기능을 제공합니다.

annotationProcessor "com.github.bumptech.glide:compiler:4.12.0": Glide 이미지 로딩 라이브러리의 컴파일러 애노테이션 프로세서를 추가합니다.

implementation 'com.github.bumptech.glide:glide:4.12.0': Glide 이미지 로딩 라이브러리를 추가합니다. 안드로이드에서 이미지 로딩과 디스플레이를 단순화

하는 데 사용됩니다.

annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0': Glide 컴파일러 애노테이션 프로세서를 추가합니다. Glide 사용 시 발생하는 애노테이션 처리에 필요합니다.

implementation 'androidx.fragment:fragment-ktx:1.5.7': AndroidX Fragment KTX 라이브러리의 1.5.7 버전을 추가합니다.

implementation 'com.squareup.moshi:moshi-kotlin:1.12.0': Moshi 라이브러리를 추가합니다. JSON 직렬화 및 역직렬화를 위한 라이브러리입니다.

implementation("com.squareup.moshi:moshi:1.13.0"): Moshi는 JSON 직렬화 및 역직렬화를 위한 모던한 JSON 라이브러리입니다. 이 라이브러리를 사용하면 JSON 데이터와 Kotlin 객체 간의 변환을 간편하게 처리할 수 있습니다. Moshi는 JSON 데이터를 Kotlin 객체로 변환하는 기능인 역직렬화(Deserialization)와 Kotlin 객체를 JSON 데이터로 변환하는 기능인 직렬화(Serialization)를 제공합니다.
구체적으로 implementation("com.squareup.moshi:moshi:1.13.0") 구문은 Maven 중앙 저장소에서 Moshi 라이브러리의 1.13.0 버전을 다운로드하여 프로젝트에 종속성으로 추가합니다. 이를 통해 프로젝트에서 Moshi 라이브러리의 클래스와 기능을 사용할 수 있게 됩니다.
Moshi 라이브러리는 안드로이드 앱 개발에서 특히 JSON 데이터와의 상호작용이 필요한 경우에 유용하게 사용될 수 있습니다. 예를 들어, 외부 API로부터 수신한 JSON 데이터를 파싱하여 앱에서 사용하는 Kotlin 객체로 변환하거나, Kotlin 객체를 JSON 형식으로 직렬화하여 서버로 전송하는 등의 작업을 Moshi를 통해 수행할 수 있습니다.

implementation 'com.squareup.retrofit2:retrofit:2.9.0': Retrofit 라이브러리를 추가합니다. RESTful API 통신을 위한 라이브러리입니다.

implementation 'com.squareup.retrofit2:converter-gson:2.9.0': Gson 변환기를 사용하여 Retrofit과 함께 JSON 데이터를 처리하기 위한 라이브러리를 추가합니다.

implementation "androidx.navigation:navigation-fragment-ktx:2.5.3": 이 구문은 AndroidX Navigation 라이브러리의 Fragment KTX 모듈을 추가합니다. Fragment KTX 모듈은 Android 앱에서 Navigation Component를 사용하여 Fragment 간의 탐색을 지원합니다. 버전 2.5.3은 사용하는 Navigation 라이브러리의 버전을 지정하는 것이며, 여기서는 2.5.3 버전을 사용합니다.

implementation "androidx.navigation:navigation-ui-ktx:2.5.3": 이 구문은 AndroidX Navigation 라이브러리의 UI KTX 모듈을 추가합니다. UI KTX 모듈은 Android 앱에서 Navigation Component를 사용하여 UI 요소(예: Toolbar, BottomNavigationView)와의 상호작용을 간편하게 처리할 수 있는 확장 함수와 속성을 제공합니다. 버전 2.5.3은 사용하는 Navigation 라이브러리의 버전을 지정하는 것이며, 여기서는 2.5.3 버전을 사용합니다.

implementation "androidx.appcompat:appcompat:1.6.1": AndroidX AppCompat 라이브러리의 1.6.1 버전을 추가합니다. 이 라이브러리는 Android의 호환성을 지원하기 위해 사용됩니다.

implementation "androidx.constraintlayout:constraintlayout:2.1.4": AndroidX ConstraintLayout 라이브러리의 2.1.4 버전을 추가합니다. ConstraintLayout은 안드로이드 앱의 레이아웃을 구성하기 위한 라이브러리로 사용됩니다.

implementation 'com.google.cloud:google-cloud-dialogflow:2.6.0': Google Cloud Dialogflow 라이브러리의 2.6.0 버전을 추가합니다. 이 라이브러리는 대화형 인터페이스를 개발하기 위해 Dialogflow를 사용하는 데 도움을 줍니다.

implementation 'joda-time:joda-time:2.10.12': Joda-Time 라이브러리의 2.10.12 버전을 추가합니다. Joda-Time은 Java에서 날짜와 시간을 처리하기 위한 강력한 라이브러리입니다.

implementation 'androidx.viewpager2:viewpager2:1.1.0-beta02': AndroidX ViewPager2 라이브러리의 1.1.0-beta02 버전을 추가합니다. ViewPager2는 안드로이드 앱에서 화면 간에 슬라이드되는 페이지를 구현하는 데 사용됩니다.

implementation("com.squareup.okhttp3:okhttp:4.9.3"): OkHttp 라이브러리의 4.9.3 버전을 추가합니다. OkHttp는 안드로이드 앱에서 HTTP 요청을 보내고 응답을 받는 데 사용되는 라이브러리입니다.

implementation "androidx.lifecycle:lifecycle-extensions:2.2.0": AndroidX Lifecycle Extensions 라이브러리의 2.2.0 버전을 추가합니다. 이 라이브러리는 Android 앱의 라이프사이클 관리를 지원하기 위해 사용됩니다.

implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.6.1': AndroidX Lifecycle Runtime KTX 라이브러리의 2.6.1 버전을 추가합니다. 이 라이브러리는 Android 앱의 라이프사이클과 관련된 Kotlin 확장 함수 및 속성을 제공합니다.

단국대학교 DANKOOK UNIVERSITY

1. MainActivity.kt

소개

MainActivity 는 앱의 메인 화면을 담당하는 역할을 합니다. 이 클래스는 다음과 같은 기능을 수행합니다:

- 이미지 로딩
- API 와의 통신
- 화면 전환
- 설정 대화상자 표시

사용 방법

MainActivity 클래스를 사용하기 위해서는 다음과 같은 단계가 있습니다.

1. MainActivity 클래스를 앱의 소스 코드에 추가합니다.
2. 필요한 라이브러리와 종속성을 추가합니다.
3. MainActivity 클래스의 필드와 메소드를 사용하여 앱의 기능을 구현합니다.

필드

MainActivity 클래스에는 다음과 같은 필드들이 있습니다:

- `apiKey`: OpenAI API 키를 저장하는 문자열 필드입니다.
- `imageView`: 이미지를 표시하는 ImageView 객체입니다.
- `newImageButton`: 새 이미지를 가져오기 위한 버튼입니다.
- `startOverButton`: 다시 시작하기 위한 버튼입니다.
- `titleTextView`: 제목을 표시하는 TextView 객체입니다.
- `ProgressBar`: 진행 상태를 표시하는 ProgressBar 객체입니다.
- `FrameLayout`: 프레임 레이아웃 객체입니다.
- `url`, `prompt`, `size`, `numImages`, `responseFormat`: API 와 통신하기 위한 필드들입니다.
- `isStoryGenerated`: 이야기가 생성되었는지 여부를 나타내는 불리언 필드입니다.
- `pythonJob`: Python 작업을 나타내는 Job 객체입니다.

단국대학교 DANKOOK UNIVERSITY

메소드

MainActivity 클래스에는 다음과 같은 주요 메소드들이 있습니다:

- `onCreate(savedInstanceState: Bundle?)`: 액티비티가 생성될 때 호출되는 메소드입니다. 초기화 작업 및 이벤트 처리 등을 수행합니다.
- `isDataFileExists()`: 데이터 파일이 존재하는지 확인하는 메소드입니다.
- `readDataFromFile()`: 파일에서 데이터를 읽어오는 메소드입니다.
- `parseApiKey(data: String): String`: 데이터에서 API 키를 추출하는 메소드입니다.
- `parseAttributes(data: String): Pair<String, String>`: 데이터에서 속성을 추출하는 메소드입니다.
- `parseJson(jsonString: String?): Map<String, Any>`: JSON 문자열을 파싱하여 맵 객체로 변환하는 메소드입니다.
- `showDialog()`: 설정 대화상자를 표시하는 메소드입니다.
- `loadData()`: 데이터를 로드하는 메소드입니다.
- `loadImage()`: 이미지를 로드하는 메소드입니다.

위의 메소드들은 MainActivity 클래스의 기능을 구현하기 위해 사용됩니다.

- startPythonJob(prompt: String): Python 작업을 시작하는 메소드입니다. 주어진 prompt 를 기반으로 Python 스크립트를 실행하여 이야기를 생성합니다.
- cancelPythonJob(): 현재 실행 중인 Python 작업을 취소하는 메소드입니다. 작업이 취소되면 이야기 생성이 중단됩니다.
- showLoading(): 로딩 상태를 표시하는 메소드입니다. ProgressBar 를 보여주고, 이미지 및 버튼 등을 비활성화합니다.
- hideLoading(): 로딩 상태를 숨기는 메소드입니다. ProgressBar 를 숨기고, 이미지 및 버튼 등을 활성화합니다.

이벤트 처리 MainActivity 클래스에는 다음과 같은 이벤트 처리를 위한 메소드들이 있습니다:

- onNewImageClick(view: View): "새 이미지 가져오기" 버튼 클릭 이벤트를 처리하는 메소드입니다. 이 메소드는 새로운 이미지를 가져오기 위해 loadImage() 메소드를 호출합니다.
- onStartOverClick(view: View): "다시 시작하기" 버튼 클릭 이벤트를 처리하는 메소드입니다. 이 메소드는 앱을 초기 상태로 되돌리기 위해 필요한 초기화 작업을 수행합니다.

데이터 저장 MainActivity 클래스에는 다음과 같은 데이터 저장을 위한 메소드들이 있습니다:

- saveDataToFile(data: String): 데이터를 파일에 저장하는 메소드입니다. 주어진 데이터를 파일에 기록합니다.
- updateDataFile(): 데이터 파일을 업데이트하는 메소드입니다. 필요한 경우 데이터 파일을 새로 다운로드하고 저장합니다.

네트워크 통신 MainActivity 클래스에는 다음과 같은 네트워크 통신을 위한 메소드들이 있습니다:

- fetchImage(url: String): 주어진 URL 에서 이미지를 가져오는 메소드입니다. 이 메소드는 비동기적으로 이미지를 다운로드하고, 가져온 이미지를 imageView 에 표시합니다.
- fetchPrompt(): API 로부터 새로운 prompt 를 가져오는 메소드입니다. 이 메소드는 비동기적으로 API 에 요청을 보내고, 응답으로 받은 prompt 를 사용하여 이야기를 생성합니다.

추가적인 UI 관련 메소드 MainActivity 클래스에는 다음과 같은 추가적인 UI 관련 메소드들이 있습니다:

- showGeneratedStory(story: String): 생성된 이야기를 표시하는 메소드입니다. 주어진 이야기를 TextView 에 표시합니다.
- clearGeneratedStory(): 이야기를 지우고 초기 상태로 되돌리는 메소드입니다. TextView 를 비웁니다.

XML 의 디자인 같은 경우 다음과 같이 2 가지로 나눌 수 있습니다.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:background="#FFFFFF"
    android:layout_height="match_parent">

    <FrameLayout
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:id="@+id/first_fragment_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:visibility="gone">

        <fragment
            android:id="@+id/nav_host_fragment"
```

```
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/nav_graph" />

</FrameLayout>

<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scaleType="centerCrop" />

<ProgressBar
    android:id="@+id/loadingProgressBar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true" />

<TextView
    android:id="@+id/titleTextView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Into the portal"
    android:textSize="24sp"
    android:textStyle="bold"
    android:gravity="center"
    android:layout_alignParentTop="true"
    android:layout_marginTop="16dp" />



<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_horizontal"
    android:layout_alignParentBottom="true"
    android:layout_marginBottom="10dp"
    android:layout_marginTop="32dp"
    android:orientation="vertical">

    <Button
        android:id="@+id/newImageButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="새로운 세상"
        android:layout_marginBottom="16dp" />

    <Button
        android:id="@+id/startButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="기존의 세상"
        android:layout_marginBottom="32dp" />

</LinearLayout>
```

```
    <ImageView
        android:id="@+id/settingsButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_settings"
        android:layout_alignParentEnd="true"
        android:layout_alignParentTop="true"
        android:layout_marginEnd="16dp"
        android:layout_marginTop="16dp" />

</RelativeLayout>
```

해당 코드에서 이어하기 버튼만 바꾸고 진행하면
2 가지 선택 화면이 뜨게 됩니다.

다음은 설정 창 dialog.XML 입니다

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="API Key 입력"
        android:textSize="18sp"
        android:textStyle="bold"
        android:gravity="center"
        android:layout_marginBottom="16dp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:gravity="center_vertical">

        <EditText
            android:id="@+id/apiKeyEditText"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="API Key 입력" />

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <Button
                android:id="@+id/checkButton"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="코드 체크" />

        <Button
            android:id="@+id/pasteButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="붙여넣기" />

    </LinearLayout>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="center">

    <Button
        android:id="@+id/confirmButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="확인"
        android:layout_marginTop="16dp"
        android:layout_marginRight="8dp" />

    <Button
        android:id="@+id/cancelButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="취소"
        android:layout_marginTop="16dp"
        android:layout_marginLeft="8dp" />

</LinearLayout>

</LinearLayout>
```

이렇게 작성하시면 대략적인 위치를 나타내어
기본적으로 사용되는 간편한 디자인으로 이용하실 수 있습니다

- 기본적으로 사용되는 간편한 디자인으로 이용하실 수 있습니다

일부 코드의 재사용 부분이 있기 때문에, 해당 부분들은 따로 다시 구체적으로는 언급하지 않고 넘어가도록 하겠습니다.

MakeStoryFragment.kt 개발자 가이드

소개

이 개발자 가이드는 안드로이드 앱의 MakeStoryFragment.kt 클래스에 대한 설명과 사용 방법을 제공합니다. MakeStoryFragment 는 이야기 생성 기능을 담당하는 프래그먼트입니다. 이 클래스는 다음과 같은 기능을 수행합니다:

- 사용자 입력 받기
- API 와의 통신
- 결과 표시

사용 방법

MakeStoryFragment.kt 클래스를 사용하기 위해서는 다음과 같은 단계를 따르세요:

1. MakeStoryFragment.kt 파일을 앱의 소스 코드에 추가합니다.
2. 필요한 라이브러리와 종속성을 추가합니다.
3. MakeStoryFragment 를 액티비티나 다른 프래그먼트에서 호출하고 표시합니다.
4. 사용자 입력을 받고 이야기 생성을 시작합니다.
5. API 와 통신하여 생성된 이야기를 받아옵니다.
6. 결과를 적절한 방식으로 표시합니다.

필드

MakeStoryFragment.kt 클래스에는 다음과 같은 필드들이 있을 수 있습니다:

- userInput: 사용자 입력을 저장하는 문자열 필드입니다.
- storyTextView: 이야기를 표시하는 TextView 객체입니다.
- progressBar: 진행 상태를 표시하는 ProgressBar 객체입니다.
- apiClient: API 와 통신을 처리하는 클라이언트 객체입니다.
- responseFormat: API 응답의 형식을 지정하는 문자열 필드입니다.
- storyGeneratedListener: 이야기 생성 완료 이벤트를 처리하는 리스너 객체입니다.

메소드

MakeStoryFragment.kt 클래스에는 다음과 같은 주요 메소드들이 있을 수 있습니다:

- onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): 프래그먼트의 뷰를 생성하고 초기화합니다.
- onViewCreated(view: View, savedInstanceState: Bundle?): 뷰가 생성된 후 초기화 작업을 수행합니다.
- getUserInput(): 사용자 입력을 받아옵니다.
- generateStory(): 사용자 입력을 기반으로 이야기 생성을 시작합니다.
- showProgressBar(): 진행 상태를 표시하는 프로그래스 바를 보여줍니다.
- hideProgressBar(): 진행 상태를 표시하는 프로그래스 바를 숨깁니다.
- displayStory(story: String): 생성된 이야기를 텍스트 뷰에 표시합니다.
- handleStoryGenerationError(error: String): 이야기 생성 중 발생한 오류를 처리합니다.

해당 코드에서 사용된 xml 은 다음과 같습니다.

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/make_story_fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/topic_input"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="주제를 입력하세요"
        android:focusableInTouchMode="true"
        android:gravity="start"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/send_button"
        android:textColor="@color/black"/>

    <Button
        android:id="@+id/send_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="보내기"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <ProgressBar
        android:id="@+id/loadingProgressBar1"
        style="@android:style/Widget.ProgressBar.Large"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="gone"
```

```
app:layout_constraintBottom_toTopOf="@+id/guideline_horizontal_center"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ProgressBar
        android:id="@+id/loadingProgressBar2"
        style="@android:style/Widget.ProgressBar.Large"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="@+id/guideline_horizontal_center" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline_horizontal_center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.5" />

    <TextView
        android:id="@+id/summary_text"
        android:textColor="@color/black"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="5dp"
        android:layout_marginEnd="5dp"
        android:fontFamily="sans-serif"
        android:textSize="14sp"
        android:visibility="gone"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"

app:layout_constraintBottom_toTopOf="@+id/guideline_horizontal_center"
/>

    <ImageView
        android:id="@+id/dalle_image"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:scaleType="fitXY"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="@+id/guideline_horizontal_center" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:text="새로운 세상"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/button2"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="정착"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/button1" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

해당 디자인을 참고하여 제작하시면 조금 더 빠른 이해로 코드를 작성하실 수 있으십니다.

StoryFragment 개발자 가이드

소개

이 개발자 가이드는 안드로이드 앱의 StoryFragment 클래스에 대한 설명과 사용 방법을 제공합니다. StoryFragment 는 이야기를 표시하고 관리하는 프래그먼트입니다. 이 클래스는 다음과 같은 기능을 수행합니다:

- 이야기 데이터 로드
- 이야기 표시
- 이야기 상세 정보 표시

사용 방법

StoryFragment 클래스를 사용하기 위해서는 다음과 같은 단계를 따르세요:

1. StoryFragment.kt 파일을 앱의 소스 코드에 추가합니다.
2. 필요한 라이브러리와 종속성을 추가합니다.
3. StoryFragment 를 액티비티나 다른 프래그먼트에서 호출하고 표시합니다.
4. 이야기 데이터를 로드합니다.

5. 로드된 이야기를 표시합니다.
6. 이야기를 선택하고 상세 정보를 표시합니다.

필드

StoryFragment 클래스에는 다음과 같은 필드들이 있을 수 있습니다:

- `storyList`: 이야기 데이터를 저장하는 리스트 필드입니다.
- `recyclerView`: 이야기를 표시하는 RecyclerView 객체입니다.
- `adapter`: RecyclerView 와 데이터를 연결하는 어댑터 객체입니다.
- `itemClickListener`: 이야기 항목 클릭 이벤트를 처리하는 리스너 객체입니다.

메소드

StoryFragment 클래스에는 다음과 같은 주요 메소드들이 있을 수 있습니다:

- `onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?)`: 프래그먼트의 뷰를 생성하고 초기화합니다.
- `onViewCreated(view: View, savedInstanceState: Bundle?)`: 뷰가 생성된 후 초기화 작업을 수행합니다.
- `loadStories()`: 이야기 데이터를 로드합니다.
- `displayStories()`: 로드된 이야기를 RecyclerView 에 표시합니다.
- `showStoryDetails(story: Story)`: 선택한 이야기의 상세 정보를 표시합니다.
- `handleStoryClick(story: Story)`: 이야기 항목 클릭 이벤트를 처리합니다.

위의 메소드들은 StoryFragment 클래스의 기능을 구현하기 위해 사용됩니다.

1. convertToNumbers(inventoryListE: List<String>): MutableList<Int>
   - 주어진 문자열 목록을 숫자 목록으로 변환하는 함수입니다.
   - 문자열과 해당하는 숫자의 매핑을 사용하여 문자열을 숫자로 변환합니다.
   - 변환된 숫자 목록을 반환합니다.
2. showConditionDialog()
   - 조건 다이얼로그를 표시하는 함수입니다.
   - 커스텀 다이얼로그 레이아웃을 설정하고 내부의 뷰들을 초기화합니다.
   - 데이터를 설정하고 다이얼로그를 표시합니다.

- closeButton 을 클릭하면 다이얼로그를 닫습니다.

3. onViewCreated(view: View, savedInstanceState: Bundle?)
   - 프래그먼트가 생성되었을 때 호출되는 함수입니다.
   - 데이터 파일이 존재하는지 확인하고, 파일이 존재하면 데이터를 읽어와서 처리합니다.
   - 데이터 파일이 존재하지 않으면 startGame() 함수를 호출하여 게임을 시작합니다.
4. isDataFileExists(): Boolean
   - 데이터 파일이 존재하는지 확인하는 함수입니다.
   - 파일 경로를 확인하여 파일이 존재하는지 여부를 반환합니다.
5. readDataFromFile(): String
   - 파일에서 데이터를 읽어오는 함수입니다.
   - 파일 경로에서 데이터를 읽어와서 문자열로 반환합니다.
6. parseData(data: String): Pair<String, MutableList<String>>
   - 데이터를 파싱하여 필요한 정보를 추출하는 함수입니다.
   - 데이터를 구분자(delimiter)로 나누고 필요한 정보를 추출하여 반환합니다.
7. onDestroy()
   - 프래그먼트가 소멸될 때 호출되는 함수입니다.
   - 실행 중인 Coroutine 을 취소합니다.
8. startGame()
   - 게임을 시작하는 함수입니다.
   - Coroutine 을 사용하여 게임을 실행하고 결과를 처리합니다.
9. setListener(listener: StoryFragmentListener)
   - StoryFragmentListener 인터페이스의 인스턴스를 설정하는 함수입니다.
   - 외부에서 StoryFragment 와 상호작용하기 위해 리스너를 설정합니다.
10. choiceNext()
    - 다음 선택지로 이동하는 함수입니다.
    - Coroutine 을 사용하여 선택지를 처리하고 결과를 업데이트합니다.
11. handleResult(result: String)
    - 게임 결과를 처리하는 함수입니다.
    - JSON 형식의 결과 데이터를 파싱하여 필요한 정보를 추출하고 UI 를 업데이트합니다.

. fun startorstop(action: String, number: Int, context: Context)

이 함수는 "start" 또는 "stop" 액션을 수신하고, 해당 액션에 따라 작업을 시작하거나 중지합니다. 함수에는 다음 매개변수가 있습니다:

- action: 시작 또는 중지 동작을 나타내는 문자열입니다.
- number: 숫자 매개변수입니다.
- context: 앱의 컨텍스트를 나타내는 객체입니다.

2. fun showToast(number: Int, context: Context)

이 함수는 토스트 메시지를 표시합니다. 함수에는 다음 매개변수가 있습니다:

- number: 숫자 매개변수입니다. 1 이면 "누군가 살아남길 시도했지만... 실패한 이야기"라는 메시지를 표시합니다.
- context: 앱의 컨텍스트를 나타내는 객체입니다.

3. fun parseJson(jsonString: String): Map<String, Any>

이 함수는 JSON 문자열을 구문 분석하여 맵 형태로 반환합니다. 함수에는 다음 매개변수가 있습니다:

- jsonString: JSON 형식의 문자열입니다.

4. private fun animateText(script: String)

이 함수는 텍스트를 애니메이션 효과와 함께 표시합니다. 함수에는 다음 매개변수가 있습니다:

- script: 표시할 스크립트 문자열입니다.

5. suspend fun getImageFromDescription(description: String): Bitmap?

이 함수는 설명(description)을 사용하여 이미지를 가져옵니다. 함수에는 다음 매개변수가 있습니다:

- description: 설명 문자열입니다.
- 반환값: 비트맵 형식의 이미지입니다.

6. fun setData(data: String)

이 함수는 데이터를 설정합니다. 함수에는 다음 매개변수가 있습니다:

- data: 설정할 데이터 문자열입니다.

7. private fun animateTextD(script: String)

이 함수는 텍스트를 애니메이션 효과와 함께 표시합니다. 함수에는 다음 매개변수가 있습니다:

- script: 표시할 스크립트 문자열입니다.

8. private fun animateTextWield(script: String)

이 함수는 텍스트를 애니메이션 효과와 함께 표시합니다. 함수에는 다음 매개변수가 있습니다:

- script: 표시할 스크립트 문자열입니다.

9. private fun animateTextWear(script: String)

이 함수는 텍스트를 애니메이션 효과와 함께 표시합니다. 함수에는 다음 매개변수가 있습니다:

- script: 표시할 스크립트 문자열입니다.

10. fun reset()

이 함수는 데이터와 뷰를 재설정합니다.

StoryFragment 개발자 가이드

StoryFragment 는 사용자에게 이야기를 표시하고 관련 작업을 수행하는 역할을 합니다. 아래의 단계에 따라 개발을 진행해주세요.

단계 1: StoryFragment 클래스 생성

1. `StoryFragment` 클래스를 생성합니다.
2. `Fragment` 클래스를 상속받습니다.
3. 필요한 import 문을 추가합니다.

```
import android.os.Bundle import android.view.LayoutInflater import
android.view.View import android.view.ViewGroup class StoryFragment :
Fragment() { // TODO: 필요한 변수 및 함수 선언 }
```

단계 2: onCreateView() 메서드 오버라이딩

1. `onCreateView()` 메서드를 오버라이딩합니다.
2. `LayoutInflater` 를 사용하여 fragment_story.xml 파일을 inflate 합니다.
3. inflate 한 뷰를 반환합니다.

```
override fun onCreateView( inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle? ): View? { return
inflater.inflate(R.layout.fragment_story, container, false) }
```

단계 3: onViewCreated() 메서드 오버라이딩

1. `onViewCreated()` 메서드를 오버라이딩합니다.
2. UI 요소를 초기화하고 이벤트 처리 등 필요한 작업을 수행합니다.

```
override fun onViewCreated( view: View, savedInstanceState: Bundle? )
{ super.onViewCreated(view, savedInstanceState) // TODO: 필요한 UI 요소
초기화 및 이벤트 처리 등의 작업 수행 }
```

단계 4: startOrStop() 함수 구현

1. `startOrStop()` 함수를 구현합니다.
2. 액션에 따라 작업을 시작하거나 중지합니다.
3. 필요한 매개변수와 컨텍스트 객체를 받아와 사용합니다.

```
private fun startOrStop(action: String, number: Int, context: Context) { if (action == "start") { // TODO: 작업 시작 } else if (action == "stop") { // TODO: 작업 중지 } }
```

단계 5: showToast() 함수 구현

1. `showToast()` 함수를 구현합니다.
2. 숫자에 따라 다른 토스트 메시지를 표시합니다.
3. 필요한 매개변수와 컨텍스트 객체를 받아와 사용합니다.

```
private fun showToast(number: Int, context: Context) { val message = when (number) { 1 -> "누군가 살아남길 시도했지만... 실패한 이야기" // TODO: 필요한 숫자에 따른 메시지 추가 else -> "" } Toast.makeText(context, message, Toast.LENGTH_SHORT).show() }
```

단계 6: parseJson() 함수 구현

1. `parseJson()` 함수를 구현합니다.
2. JSON 데이터를 파싱하여 필요한 정보를 추출합니다.
3. 필요한 매개변수와 컨텍스트 객체를 받아와 사용합니다.

```
private fun parseJson(json: String, context: Context) { // TODO: JSON 파싱 및 정보 추출 작업 수행 }
```

이제 위의 가이드를 참고하여 StoryFragment 를 개발할 수 있습니다. 필요에 따라 각 단계의 코드를 수정하거나 추가적인 작업을 진행할 수 있습니다. 세부 구현 사항은 개발 환경과 요구사항에 따라 다를 수 있습니다.

다음은 해당 코드에서 사용되는 모든 xml 을 작성해보겠습니다.

Story_fragment.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:background="#FFFFFF"
    android:layout_height="match_parent">
```

```
-->
    <!-- Condition Button -->
    <ImageButton
        android:id="@+id/conditionButton"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:scaleType="fitXY"
        android:layout_alignParentEnd="true"
        android:layout_alignParentTop="true"
        android:layout_marginEnd="16dp"
        android:layout_marginTop="16dp"
        android:src="@drawable/condition_icon"
        android:background="@android:color/transparent"
        android:contentDescription="Condition" />


    <!-- Description -->
    <TextView
        android:id="@+id/descriptionTextView"
        android:layout_width="match_parent"
        android:fontFamily="@font/scdream9"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:layout_below="@id/conditionButton"
        android:padding="16dp"
        android:text="" />

    <!-- Story Text -->
    <TextView
        android:id="@+id/storyTextView"
        android:layout_width="match_parent"
        android:fontFamily="@font/scdream9"
        android:textColor="#000000"
        android:layout_height="wrap_content"
        android:layout_below="@id/descriptionTextView"
        android:padding="16dp"
        android:text="" />

    <TextView
        android:id="@+id/wearingText"
        android:layout_width="match_parent"
        android:fontFamily="@font/scdream9"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:layout_below="@id/storyTextView"
        android:padding="16dp"
        android:text="" />

    <TextView
        android:id="@+id/wieldingText"
        android:layout_width="match_parent"
        android:fontFamily="@font/scdream9"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:layout_below="@id/wearingText"
```

```
        android:padding="16dp"
        android:text="" />

    <!-- Inventory Button -->
    <ImageButton
        android:id="@+id/inventoryButton"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:scaleType="fitXY"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true"
        android:layout_marginStart="16dp"
        android:layout_marginBottom="16dp"
        android:src="@drawable/inventory"
        android:background="@android:color/transparent"
        android:contentDescription="Inventory" />

    <!-- Talk Button -->
    <ImageButton
        android:id="@+id/talkButton"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:scaleType="fitXY"
        android:layout_alignParentBottom="true"
        android:layout_alignParentEnd="true"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        android:src="@drawable/talk"
        android:background="@android:color/transparent"
        android:contentDescription="Talk" />

    <!-- Progress Bar -->
    <ProgressBar
        android:id="@+id/progressBar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_above="@id/inventoryButton"
        android:visibility="gone" />

</RelativeLayout>
```

그 다음으로 버튼 상호작용 해서 condition.xml 작성해보겠습니다.

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="290dp"
    android:layout_height="wrap_content"
    android:background="@drawable/condition_dialog_bg"
    android:padding="16dp">

    <ImageButton
        android:id="@+id/closeButton"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:layout_alignParentEnd="true"
```

```
        android:layout_marginTop="8dp"
        android:src="@drawable/ic_close"
        android:contentDescription="Close" />

    <TextView
        android:layout_alignParentStart="true"
        android:id="@+id/conditionTitleTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:text="상태창"
        android:fontFamily="@font/scdream9"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_marginTop="8dp" />

    <TextView
        android:layout_alignParentStart="true"
        android:id="@+id/conditionDetailsTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:text="여러 정보를 포함하고 있습니다."
        android:fontFamily="@font/scdream7"
        android:textSize="16sp"
        android:layout_below="@id/conditionTitleTextView"
        android:layout_marginTop="8dp"
        android:layout_toStartOf="@id/closeButton" />

    <LinearLayout
        android:id="@+id/headerLayout"
        android:layout_width="95dp"
        android:layout_height="392dp"
        android:orientation="vertical"
        android:layout_below="@id/conditionDetailsTextView"
        android:layout_marginTop="16dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textStyle="bold"
            android:fontFamily="@font/scdream9"
            android:textColor="@color/black"
            android:text="차례" />

        <RelativeLayout
            android:layout_width="100dp"
            android:layout_height="30dp"
            android:layout_marginTop="10dp">

            <ImageView
                android:id="@+id/turnIconImageView"
                android:layout_width="30dp"
                android:layout_height="30dp"
                android:layout_centerVertical="true"
```

```
            android:contentDescription="Turn Icon"
            android:src="@drawable/turn_icon" />

        <TextView
            android:id="@+id/turnNumberTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_toEndOf="@id/turnIconImageView"
            android:textColor="@color/black"
            android:layout_marginStart="4dp"
            android:textStyle="bold"
            android:fontFamily="@font/scdream7"
            android:text="Turn: 1" />

    </RelativeLayout>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:textStyle="bold"
        android:fontFamily="@font/scdream9"
        android:layout_marginTop="10dp"
        android:textColor="@color/black"
        android:text="난이도" />

    <TextView
        android:layout_marginTop="8dp"
        android:id="@+id/difficultyTextView"
        android:layout_width="wrap_content"
        android:textColor="@color/black"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:fontFamily="@font/scdream7"
        android:text="Difficulty: 5" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:textStyle="bold"
        android:fontFamily="@font/scdream9"
        android:textColor="@color/black"
        android:layout_marginTop="10dp"
        android:text="시간" />

    <RelativeLayout
        android:layout_width="100dp"
        android:layout_height="30dp"
        android:layout_marginTop="10dp">

        <ImageView
            android:id="@+id/timeIconImageView"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_centerVertical="true"
```

```
            android:contentDescription="Time Icon"
            android:src="@drawable/time_icon" />

        <TextView
            android:id="@+id/timePeriodTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_toEndOf="@id/timeIconImageView"
            android:textStyle="bold"
            android:fontFamily="@font/scdream7"
            android:textColor="@color/black"
            android:textSize="10dp"
            android:layout_marginStart="4dp"
            android:text="Afternoon" />

    </RelativeLayout>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:fontFamily="@font/scdream9"
        android:layout_marginTop="10dp"
        android:textColor="@color/black"
        android:text="날짜" />

    <RelativeLayout
        android:layout_width="130dp"
        android:layout_height="30dp"
        android:layout_marginTop="10dp">

        <ImageView
            android:id="@+id/dayIconImageView"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_centerVertical="true"
            android:src="@drawable/day_icon"
            android:contentDescription="Day Icon" />

        <TextView
            android:id="@+id/dayNumberTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_toEndOf="@id/dayIconImageView"
            android:layout_centerVertical="true"
            android:textColor="@color/black"
            android:textStyle="bold"
            android:fontFamily="@font/scdream7"
            android:layout_marginStart="4dp"
            android:text="Day: 1" />

    </RelativeLayout>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
            android:textStyle="bold"
            android:fontFamily="@font/scdream9"
            android:layout_marginTop="10dp"
            android:textColor="@color/black"
            android:text="날씨" />

        <RelativeLayout
            android:layout_width="130dp"
            android:layout_height="50dp"
            android:layout_marginTop="0dp">

            <ImageView
                android:id="@+id/weatherIconImageView"
                android:layout_width="30dp"
                android:layout_height="30dp"
                android:layout_centerVertical="true"
                android:src="@drawable/weather_icon"
                android:contentDescription="Weather Icon" />

            <TextView
                android:id="@+id/weatherTextView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_toEndOf="@id/weatherIconImageView"
                android:layout_centerVertical="true"
                android:textStyle="bold"
                android:textSize="10sp"
                android:textColor="@color/black"
                android:fontFamily="@font/scdream7"
                android:layout_marginStart="4dp"
                android:text="Cloudy" />

        </RelativeLayout>

        <TextView
            android:id="@+id/weatherTextView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:fontFamily="@font/scdream7"
            android:textSize="10sp"
            android:textColor="@color/black"
            android:text="" />
        <!-- Add any additional views or information you want to display
in the header -->

    </LinearLayout>

    <LinearLayout
        android:id="@+id/statsLayout"
        android:layout_width="160dp"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_below="@id/conditionDetailsTextView"
        android:layout_alignParentEnd="true"
        android:layout_marginTop="16dp">
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:fontFamily="@font/scdream9"
        android:textColor="@color/black"
        android:text="Level" />

    <RelativeLayout
        android:layout_width="100dp"
        android:layout_height="30dp"
        android:layout_marginTop="10dp">

        <ImageView
            android:id="@+id/levelIconImageView"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_centerVertical="true"
            android:contentDescription="Level Icon"
            android:src="@drawable/level_icon" />

        <TextView
            android:id="@+id/levelTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_toEndOf="@id/levelIconImageView"
            android:textStyle="bold"
            android:fontFamily="@font/scdream7"
            android:textColor="@color/black"
            android:layout_marginStart="16dp"
            android:text="Level: 1" />

    </RelativeLayout>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:fontFamily="@font/scdream9"
        android:layout_marginTop="10dp"
        android:textColor="@color/black"
        android:text="경험치 (XP)" />

    <ProgressBar
        android:id="@+id/xpProgressBar"
        style="@android:style/Widget.ProgressBar.Horizontal"
        android:layout_width="match_parent"
        android:layout_height="20dp"
        android:layout_marginTop="8dp"
        android:max="100"
        android:progress="0"
        android:progressTint="#00FF00" />

    <TextView
        android:fontFamily="@font/scdream9"
        android:layout_marginTop="10dp"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/black"
            android:text="HP" />

        <RelativeLayout
            android:layout_width="150dp"
            android:layout_height="30dp">

            <ImageView
                android:id="@+id/healthIconImageView"
                android:layout_width="30dp"
                android:layout_height="30dp"
                android:layout_centerVertical="true"
                android:src="@drawable/heart_icon"

                android:contentDescription="Heart Icon" />

            <TextView
                android:id="@+id/healthTextView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_toEndOf="@id/healthIconImageView"
                android:layout_centerVertical="true"
                android:layout_marginStart="10dp"
                android:fontFamily="@font/scdream7"
                android:textColor="@color/black"
                android:text="Health: 20/20" />

        </RelativeLayout>

        <!-- Add any additional views or information you want to display
in the stats section -->

    </LinearLayout>

    <LinearLayout
        android:id="@+id/abilitiesLayout"
        android:layout_width="160dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/statsLayout"
        android:layout_alignParentEnd="true"
        android:layout_marginTop="7dp"
        android:orientation="vertical">

        <TextView
            android:id="@+id/abilitiesTitleTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="8dp"
            android:textColor="@color/black"
            android:text="Abilities"
            android:textSize="18sp"
            android:textStyle="bold" />

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
```

```
        android:orientation="horizontal"
        android:gravity="center_vertical">

        <ImageView
            android:id="@+id/persuasionIconImageView"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_centerVertical="true"
            android:contentDescription="Persuasion Icon"
            android:src="@drawable/persuasion_icon" />

        <TextView
            android:id="@+id/persuasionTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_toEndOf="@id/persuasionIconImageView"
            android:layout_marginStart="8dp"
            android:textColor="@color/black"
            android:fontFamily="@font/scdream7"
            android:text="Persuasion: 12" />

    </RelativeLayout>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="7dp"
        android:orientation="horizontal"
        android:gravity="center_vertical">

        <ImageView
            android:id="@+id/strengthIconImageView"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_centerVertical="true"
            android:contentDescription="Strength Icon"
            android:src="@drawable/strength_icon" />

        <TextView
            android:id="@+id/strengthTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:fontFamily="@font/scdream7"
            android:textColor="@color/black"
            android:layout_centerVertical="true"
            android:layout_toEndOf="@id/strengthIconImageView"
            android:text="Strength: 8" />

    </RelativeLayout>

    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="7dp"
        android:orientation="horizontal"
        android:gravity="center_vertical">
```

```
        <ImageView
            android:id="@+id/intelligenceIconImageView"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_centerVertical="true"
            android:contentDescription="Intelligence Icon"
            android:src="@drawable/intelligence_icon" />

        <TextView
            android:id="@+id/intelligenceTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:textColor="@color/black"
            android:fontFamily="@font/scdream7"
            android:layout_centerVertical="true"
            android:layout_toEndOf="@id/intelligenceIconImageView"
            android:text="Intelligence: 14" />

    </RelativeLayout>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="7dp"
        android:orientation="horizontal"
        android:gravity="center_vertical">

        <ImageView
            android:id="@+id/dexterityIconImageView"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_centerVertical="true"
            android:contentDescription="Dexterity Icon"
            android:src="@drawable/dexterity_icon" />

        <TextView
            android:id="@+id/dexterityTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:textColor="@color/black"
            android:fontFamily="@font/scdream7"
            android:layout_centerVertical="true"
            android:layout_toEndOf="@id/dexterityIconImageView"
            android:text="Dexterity: 10" />

    </RelativeLayout>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="7dp"
        android:orientation="horizontal"
        android:gravity="center_vertical">

        <ImageView
```

```
            android:id="@+id/luckIconImageView"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_centerVertical="true"
            android:contentDescription="Luck Icon"
            android:src="@drawable/luck_icon" />

        <TextView
            android:id="@+id/luckTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:textColor="@color/black"
            android:layout_toEndOf="@id/luckIconImageView"
            android:layout_marginStart="10dp"
            android:fontFamily="@font/scdream7"
            android:text="Luck: 16" />

    </RelativeLayout>

</LinearLayout>


</RelativeLayout>
```

그 다음으로 fragment_inventory.xml 을 작성해보도록 하겠습니다.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <!-- Title -->

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:text="Inventory"
            android:textSize="20sp"
            android:textStyle="bold"
            android:padding="16dp" />

        <ImageButton
            android:id="@+id/closeButton"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_alignParentEnd="true"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
```

```
        android:contentDescription="Close"
        android:src="@drawable/ic_close" />

</RelativeLayout>

<!-- Gold -->

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="center_vertical"
    android:padding="16dp">

    <ImageView
        android:id="@+id/goldimage"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:src="@drawable/ic_gold" />

    <TextView
        android:id="@+id/gold"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:fontFamily="@font/scdream7"
        android:textSize="25sp"
        android:layout_marginStart="8dp" />

</LinearLayout>

<!-- Inventory List -->

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal">

    <!-- Left Side -->
    <ListView
        android:id="@+id/inventoryLeftListView"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"/>

    <!-- Right Side -->
    <ListView
        android:id="@+id/inventoryRightListView"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"/>

    <TextView
        android:id="@+id/emptyTextView"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
```

```
            android:text = "안녕하세요"
            android:visibility="gone" />

    </LinearLayout>

    <!-- Buttons -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <!-- Use Button -->
        <View
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_weight="1" />

        <Button
            android:id="@+id/useButton"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:fontFamily="@font/scdream7"
            android:text="사용"
            android:textSize="14sp"
            android:layout_weight="1"
            android:layout_marginTop="8dp"
            android:layout_marginStart="2dp"
            android:layout_marginEnd="2dp"
            android:padding="8dp" />

        <View
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_weight="1" />

        <Button
            android:id="@+id/discardButton"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="취소"
            android:fontFamily="@font/scdream7"
            android:textSize="14sp"
            android:layout_weight="1"
            android:layout_marginTop="8dp"
            android:layout_marginStart="2dp"
            android:layout_marginEnd="2dp"
            android:padding="8dp" />

        <View
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_weight="1" />

    </LinearLayout>

</LinearLayout>
```

다음으로 inventory 에 있ㄴ느

Dialog 화면 2 개를 작성했습니다.

Dialog_gold.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="얼마의 골드를 사용하시겠습니까"
        android:textSize="18sp"
        android:textStyle="bold"
        android:gravity="center"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:gravity="center_vertical">

        <Space
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_weight="1" />

        <ImageButton
            android:id="@+id/buttonDecrease"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:src="@drawable/ic_arrow_down" />



        <EditText
            android:id="@+id/textViewQuantity"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:inputType="number"
            android:text="0"
            android:textSize="24sp"
            android:textStyle="bold"
            android:layout_marginStart="16dp"
            android:layout_marginEnd="16dp"/>

        <ImageButton
            android:id="@+id/buttonIncrease"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:src="@drawable/ic_arrow_up" />
```

```
        <Space
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_weight="1" />

    </LinearLayout>

    <EditText
        android:id="@+id/editTextGold"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="그리고 어떻게 사용하시겠습니까"
        android:layout_marginEnd="8dp"
        android:gravity="center"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:gravity="center">



        <Button
            android:id="@+id/buttonConfirm"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Confirm"
            android:layout_marginEnd="8dp"/>

        <Button
            android:id="@+id/buttonCancel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Cancel"
            android:layout_marginStart="8dp"/>

    </LinearLayout>

</LinearLayout>
```

그 다음으로

dialog_item_quantity.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/condition_dialog_bg"
    android:orientation="vertical"
    android:padding="16dp">
```

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="30dp"
    android:layout_weight="1"
    android:gravity="center"
    android:textColor="@color/black"
    android:text="아이템 개수 선택"
    android:textSize="18sp"
    android:textStyle="bold" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_weight="7"
    android:orientation="horizontal"
    android:padding="16dp">

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="3"
        android:gravity="center"
        android:orientation="vertical">

        <EditText
            android:id="@+id/editQuantity"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textColor="@color/black"
            android:background="@null"
            android:gravity="center"
            android:inputType="number"
            android:text="1"
            android:textSize="24sp" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="55dp"
        android:layout_weight="1"
        android:gravity="center"
        android:orientation="vertical">

        <ImageButton
            android:id="@+id/buttonIncrease"
            android:layout_width="match_parent"
            android:layout_height="20dp"
            android:layout_marginBottom="5dp"
            android:background="@android:color/transparent"
            android:src="@drawable/ic_arrow_up" />


        <ImageButton
            android:id="@+id/buttonDecrease"
            android:layout_width="match_parent"
```

```
            android:layout_height="20dp"
            android:layout_marginTop="5dp"
            android:background="@android:color/transparent"
            android:src="@drawable/ic_arrow_down" />

    </LinearLayout>

</LinearLayout>


<TextView
    android:id="@+id/textViewQuantity"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:gravity="center"
    android:textColor="@color/black"
    android:text="최대 개수: 3"
    android:textSize="16sp" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <Button
        android:id="@+id/checkdButton"
        android:layout_width="wrap_content"
        android:layout_height="50dp"
        android:text="확인"
        android:fontFamily="@font/scdream7"
        android:textSize="14sp"
        android:layout_weight="1"
        android:layout_marginTop="8dp"
        android:layout_marginStart="2dp"
        android:layout_marginEnd="2dp"
        android:padding="8dp" />

    <Button
        android:id="@+id/cancelButton"
        android:layout_width="wrap_content"
        android:layout_height="50dp"
        android:text="취소"
        android:fontFamily="@font/scdream7"
        android:textSize="14sp"
        android:layout_weight="1"
        android:layout_marginTop="8dp"
        android:layout_marginStart="2dp"
        android:layout_marginEnd="2dp"
        android:padding="8dp" />

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
```

```
        android:layout_weight="1"
        android:gravity="center"
        android:orientation="horizontal">

        <Button
            android:id="@+id/buttonCancel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="?android:attr/selectableItemBackground"
            android:text="취소"
            android:textSize="16sp" />

        <Space
            android:layout_width="4dp"
            android:layout_height="match_parent" />

        <Button
            android:id="@+id/buttonConfirm"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="?android:attr/selectableItemBackground"
            android:text="확인"
            android:textSize="16sp" />

    </LinearLayout>

</LinearLayout>
```

이제 StoryFragment 의 마지막 chat 화면입니다.

Activity_chat.xml

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

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="8dp">

        <EditText
            android:id="@+id/messageEditText"
            android:layout_width="0dp"
```

```
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="Enter message"/>

        <Button
            android:id="@+id/sendButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Send"/>

    </LinearLayout>

</LinearLayout>
```

Profile_info.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="350dp"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <ImageButton
        android:id="@+id/closeButton"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:src="@drawable/ic_close"
        android:contentDescription="Close" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <ImageView
            android:id="@+id/characterFaceImageView"
            android:layout_width="200dp"
            android:layout_height="200dp"
            android:scaleType="centerCrop" />

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_marginStart="16dp">

            <LinearLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:orientation="horizontal"
                android:layout_marginTop="7dp"
```

```
        android:gravity="center_vertical">

        <TextView
            android:id="@+id/nameTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:fontFamily="@font/scdream7"
            android:textSize="20sp"
            android:textColor="@color/black"
            android:textStyle="bold"
            android:text="이름" />

        <TextView
            android:id="@+id/nameTextView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:textColor="@color/black"
            android:textSize="16sp"
            android:fontFamily="@font/scdream7"
            android:text="이름 2" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="8dp">

        <TextView
            android:id="@+id/genderTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:fontFamily="@font/scdream7"
            android:textColor="@color/black"
            android:textSize="20sp"
            android:text="성별" />

        <ImageView
            android:id="@+id/genderIconImageView1"
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:src="@drawable/male"
            android:layout_marginStart="8dp"
            android:visibility="gone" />

        <ImageView
            android:id="@+id/genderIconImageView2"
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:src="@drawable/female"
            android:layout_marginStart="8dp"
            android:visibility="gone" />

    </LinearLayout>
```

```
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:layout_marginTop="7dp"
            android:gravity="center_vertical">


            <TextView
                android:id="@+id/ageTextView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textColor="@color/black"
                android:fontFamily="@font/scdream9"
                android:textSize="20sp"
                android:text="나이" />

            <TextView
                android:id="@+id/ageTextView2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textColor="@color/black"
                android:layout_marginStart="8dp"
                android:textSize="16sp"
                android:fontFamily="@font/scdream7"
                android:text="나이 2" />

        </LinearLayout>

    </LinearLayout>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="7dp"
    android:gravity="center_vertical">

    <TextView
        android:id="@+id/personalityTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:fontFamily="@font/scdream9"
        android:textColor="@color/black"
        android:layout_marginTop="8dp"
        android:text="성격" />

    <TextView
        android:id="@+id/personalityTextView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
```

```
        android:textColor="@color/black"
        android:textSize="16sp"
        android:layout_marginTop="8dp"
        android:fontFamily="@font/scdream7"
        android:text="역할 2" />

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="7dp"
    android:gravity="center_vertical">

    <TextView
        android:id="@+id/roleTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:textColor="@color/black"
        android:fontFamily="@font/scdream9"
        android:layout_marginTop="8dp"
        android:text="역할" />

    <TextView
        android:id="@+id/roleTextView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:textColor="@color/black"
        android:textSize="16sp"
        android:layout_marginTop="8dp"
        android:fontFamily="@font/scdream7"
        android:text="역할 2" />

</LinearLayout>


<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_marginTop="7dp"
    android:gravity="center_vertical">

    <TextView
        android:id="@+id/hobbyTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="@font/scdream9"
        android:textSize="20sp"
        android:layout_marginTop="8dp"
        android:textColor="@color/black"
        android:text="취미" />
```

```
        <TextView
            android:id="@+id/hobbyTextView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:textSize="16sp"
            android:layout_marginTop="8dp"
            android:fontFamily="@font/scdream7"
            android:textColor="@color/black"
            android:text="취미 2" />


    </LinearLayout>


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="7dp"
        android:gravity="center_vertical">

        <TextView
            android:id="@+id/locationTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:layout_marginTop="8dp"
            android:fontFamily="@font/scdream9"
            android:textColor="@color/black"
            android:text="거주지역" />

        <TextView
            android:id="@+id/locationTextView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:textSize="16sp"
            android:layout_marginTop="8dp"
            android:textColor="@color/black"
            android:fontFamily="@font/scdream7"
            android:text="사는 곳 2" />

    </LinearLayout>


</LinearLayout>
```

Chat_rooms_dialog.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:id="@+id/parentLayout2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">



    <LinearLayout
        android:id="@+id/parentLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/profilePhotoImageView"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:src="@drawable/profile_photo" />

        <ImageView
            android:id="@+id/addChatRoomButton"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:src="@drawable/plus_icon" />

    </LinearLayout>

    <LinearLayout
        android:id="@+id/chatLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:orientation="vertical">

        <LinearLayout
            android:id="@+id/normallayout"
            android:layout_width="match_parent"
            android:layout_height="50dp"
            android:layout_weight="1"
            android:orientation="vertical">

            <TextView
                android:id="@+id/chatRoomNameTextView"
                android:layout_width="match_parent"
                android:layout_height="30dp"
                android:text="채팅방 상대 이름"
                android:fontFamily="@font/scdream9"
                android:textSize="20sp" />

            <TextView
                android:id="@+id/recentMessageTextView"
                android:layout_width="match_parent"
                android:layout_height="70dp"
                android:text="가장 최근에 이야기한 내용"
                android:fontFamily="@font/scdream7"
                android:textSize="15sp" />
```

```
        </LinearLayout>

    </LinearLayout>

</LinearLayout>
```

이렇게 작성하시면 해당 코드의 작동에 유용하게 사용하실 수 있습니다.

Com.exmaple.chatbot 패키지 설명

해당 패키지는 dialogflow2를 이용하여 만든 챗봇 패키지로 조금 더 복잡하지만 내용은 효과적으로 하게끔 작성되었습니다. 해당 코드의 최적화 부분을 처리한다면, 원활히 진행하실 수 있습니다.

챗봇 애플리케이션 개발자 가이드

이 개발자 가이드는 챗봇 애플리케이션을 개발하기 위한 주요 구성 요소들에 대한 설명과 사용 방법을 안내합니다. 아래에서는 `ChatActivity`, `ChatFragment`, `MessageAdapter`, `Message`, `SendRequestTask` 클래스들을 다루고 있습니다.

1. ChatActivity 클래스

`ChatActivity` 클래스는 채팅 화면을 관리하고 사용자와의 상호작용을 처리하는 역할을 합니다. 다음은 `ChatActivity` 클래스의 주요 기능입니다:

- Dialogflow 와의 세션 관리 및 통신
- 사용자의 메시지 입력 처리
- 응답 메시지 표시 및 갱신

사용 방법

1. `ChatActivity` 클래스를 생성합니다.
2. `onCreate` 메서드에서 `recyclerView`, `sendButton`, `messageEditText` 등의 필요한 UI 요소들을 초기화합니다.

3. Dialogflow 세션 클라이언트를 생성하는 `createSessionsClient` 메서드를 구현합니다.
4. 메시지 전송 버튼 클릭 시 실행되는 `onClick` 메서드에서 다음 동작들을 수행합니다:
    - 사용자가 입력한 메시지를 가져옵니다.
    - 메시지 리스트에 새로운 메시지를 추가합니다.
    - 얻은 메시지를 Dialogflow 로 전송하는 `SendRequestTask` 클래스를 실행합니다.

2. ChatFragment 클래스

`ChatFragment` 클래스는 채팅 화면을 Fragment 형태로 관리하는 역할을 합니다. 다음은 `ChatFragment` 클래스의 주요 기능입니다:

- Dialogflow 와의 통신을 위한 API 키 관리
- 메시지 목록을 표시하고 갱신하는 RecyclerView 관리
- 사용자의 메시지 입력 처리

사용 방법

1. `ChatFragment` 클래스를 생성합니다.
2. `onCreateView` 메서드에서 적절한 레이아웃 파일을 inflate 하고 필요한 UI 요소들을 초기화합니다.
3. `onViewCreated` 메서드에서 다음 동작들을 수행합니다:
    - 전송 버튼 클릭 리스너를 설정하여 사용자의 메시지를 처리합니다.
    - 사용자가 입력한 메시지를 가져옵니다.
    - 메시지 리스트에 새로운 메시지를 추가합니다.
    - 얻은 메시지를 Dialogflow 로 전송하는 `SendRequestTask` 클래스를 실행합니다.

3. MessageAdapter 클래스

`MessageAdapter` 클래스는 채팅 화면에서 메시지 목록을 표시하기 위한 RecyclerView 의 어댑터 역할을 합니다. 다음은 `MessageAdapter` 클래스의 주요 기능입니다:

- 메시지 목록을 RecyclerView 에 바인딩하여 표시
- 메시지의 송신자에 따라 다른 뷰 홀더 및 배경 설정

사용 방법

1. `MessageAdapter` 클래스를 생성합니다.
2. `onCreateViewHolder` 메서드에서 메시지 아이템을 위한 뷰 홀더를 생성합니다.
3. `onBindViewHolder` 메서드에서 메시지 데이터를 뷰 홀더에 바인딩하고 화면에 표시합니다.
4. `getItemCount` 메서드에서 전체 메시지 개수를 반환합니다.

4. Message 클래스

`Message` 클래스는 채팅 메시지의 데이터 모델을 나타냅니다. 다음은 `Message` 클래스의 주요 기능입니다:

- 메시지 텍스트와 송신자 정보를 저장

사용 방법

1. `Message` 클래스를 생성합니다.
2. `getMessageText` 메서드를 사용하여 메시지 텍스트를 가져옵니다.
3. `isSentByUser` 메서드를 사용하여 송신자가 사용자인지 확인합니다.

5. SendRequestTask 클래스

`SendRequestTask` 클래스는 Dialogflow 와의 통신을 위해 HTTP POST 요청을 보내는 작업을 수행합니다. 다음은 `SendRequestTask` 클래스의 주요 기능입니다:

- Dialogflow 에 메시지를 전송하고 응답을 처리

사용 방법

1. `SendRequestTask` 클래스를 생성합니다.
2. `run` 메서드에서 HTTP POST 요청을 보내고 응답을 처리합니다.

ChatActivity 개발자 가이드

`ChatActivity`는 대화형 챗봇 기능을 제공하는 화면입니다. 아래의 단계에 따라 개발을 진행해주세요.

단계 1: ChatActivity 클래스 생성

1. `ChatActivity` 클래스를 생성합니다.
2. `AppCompatActivity` 클래스를 상속받습니다.
3. 필요한 import 문을 추가합니다.

```
import android.os.AsyncTask;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;



import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;



import com.google.cloud.dialogflow.v2.DetectIntentRequest;

import com.google.cloud.dialogflow.v2.DetectIntentResponse;

import com.google.cloud.dialogflow.v2.QueryInput;

import com.google.cloud.dialogflow.v2.QueryParameters;

import com.google.cloud.dialogflow.v2.SessionName;
```

```
import com.google.cloud.dialogflow.v2.SessionsClient;

import com.google.cloud.dialogflow.v2.SessionsSettings;

import com.google.cloud.dialogflow.v2.TextInput;



import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.TimeZone;

import java.util.UUID;



public class ChatActivity extends AppCompatActivity {

    // TODO: 필요한 변수 및 객체 선언

}
```

단계 2: onCreate() 메서드 오버라이딩

1. `onCreate()` 메서드를 오버라이딩합니다.
2. 필요한 레이아웃과 UI 요소를 초기화합니다.
3. 세션 클라이언트를 생성합니다.
4. 리사이클러뷰와 어댑터를 초기화합니다.

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

    recyclerView = findViewById(R.id.recyclerView);
    sendButton = findViewById(R.id.sendButton);
    messageEditText = findViewById(R.id.messageEditText);

    sessionsClient = createSessionsClient();

    layoutManager = new LinearLayoutManager(this);
    recyclerView.setLayoutManager(layoutManager);
```

```
        mAdapter = new MessageAdapter(messageList, "chatbot");
        recyclerView.setAdapter(mAdapter);

        // TODO: 전송 버튼 클릭 이벤트 처리
}
```

단계 3: createSessionsClient() 메서드 구현

1. `createSessionsClient()` 메서드를 구현합니다.
2. 세션 클라이언트를 생성하고 세션 설정을 초기화합니다.
3. 세션 클라이언트를 반환합니다.

```
private SessionsClient createSessionsClient() {

        try {

                SessionsSettings.Builder settingsBuilder =
SessionsSettings.newBuilder();

                SessionsSettings sessionsSettings = settingsBuilder.build();

                return SessionsClient.create(sessionsSettings);

        } catch (Exception e) {

                Toast.makeText(this, "Error creating sessions client",
Toast.LENGTH_SHORT).show();

                return null;

        }

} 단계 4: getSessionId() 메서드 구현
```

1. `getSessionId()` 메서드를 구현합니다.
2. 현재 역할에 대한 세션 ID 가 이미 있는지 확인합니다.
3. 이미 있는 경우 해당 세션 ID 를 반환합니다.
4. 없는 경우 새로운 세션 ID 를 생성하고 현재 역할에 대한 세션 맵에 저장한 후 반환합니다.

```
private String getSessionId() {
```

```
    // TODO: 현재 역할에 대한 세션 ID 확인 및 생성 로직 구현
}
```

단계 5: getRoleName() 메서드 구현

1. `getRoleName()` 메서드를 구현합니다.
2. 현재 사용자의 역할 이름을 가져오는 로직을 구현합니다.
3. 예를 들어 Firebase 인증을 사용하여 사용자의 역할을 가져올 수 있습니다.
4. 이 예제에서는 하드코딩된 역할 이름을 반환합니다.

```
private String getRoleName() {
    // TODO: 현재 사용자의 역할 이름 가져오기 로직 구현
    return "user";
}
```

단계 6: SendRequestTask 클래스 구현

1. `SendRequestTask` 클래스를 구현합니다.
2. `AsyncTask`를 상속받습니다.
3. `doInBackground()` 메서드를 오버라이딩하여 백그라운드에서 작업을 수행합니다.
4. 전달된 매개변수를 사용하여 Dialogflow 에 대화 요청을 보냅니다.
5. `onPostExecute()` 메서드를 오버라이딩하여 요청 결과를 처리합니다.

```
private class SendRequestTask extends AsyncTask<String, Void,
DetectIntentResponse> {
    @Override
    protected DetectIntentResponse doInBackground(String... strings) {
        // TODO: Dialogflow 에 대화 요청 보내기 및 결과 받아오기 로직
구현
    }

    @Override
```

```
    protected void onPostExecute(DetectIntentResponse response) {

        // TODO: 요청 결과 처리 로직 구현

    }

}
```

단계 7: 전송 버튼 클릭 이벤트 처리

1. 전송 버튼 클릭 이벤트를 처리하는 코드를 추가합니다.
2. 사용자가 입력한 메시지를 가져옵니다.
3. 메시지를 메시지 목록에 추가하고 어댑터에 변경 사항을 알립니다.
4. 입력 필드를 초기화합니다.
5. 세션 ID와 역할 이름을 가져옵니다.
6. `SendRequestTask`를 실행하여 Dialogflow에 대화 요청을 보냅니다.

```
sendButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // TODO: 전송 버튼 클릭 이벤트 처리 로직 구현
    }
});
```

단계 8: 전체 코드 검토 및 수정

1. 위의 단계를 참고하여 `ChatActivity` 클래스를 완성합니다.
2. 필요에 따라 코드를 검토하고 수정하세요.
3. 필요한 경우 Dialogflow의 언어 코드, 시간대 설정 등을 변경하세요.

ChatFragment 개발자 가이드

`ChatFragment`는 사용자와 챗봇 간의 대화를 표시하고 사용자 입력을 처리하는 데 사용되는 Fragment입니다. 아래 가이드에서는 `ChatFragment` 클래스의 코드를 설명하고 사용 방법을 안내합니다.

단계 1: ChatFragment 클래스 생성

1. `ChatFragment` 클래스를 생성합니다.
2. `Fragment`를 상속받습니다.

```
public class ChatFragment extends Fragment {

    // 코드 작성

}
```

단계 2: onCreateView() 메서드 오버라이딩

1. `onCreateView()` 메서드를 오버라이딩합니다.
2. `fragment_chat` 레이아웃을 inflate 하여 View 를 생성합니다.
3. `RecyclerView` 및 기타 필요한 구성 요소를 초기화합니다.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_chat, container, false);

    // RecyclerView 및 기타 구성 요소 초기화

    return view;
```

단계 3: onViewCreated() 메서드 오버라이딩

1. `onViewCreated()` 메서드를 오버라이딩합니다.
2. View 가 생성된 후 호출되는 메서드입니다.
3. 전송 버튼 클릭 이벤트 처리 코드를 추가합니다.
4. 사용자 입력을 처리하고 대화를 업데이트하는 `SendRequestTask`를 실행합니다.

```
@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    // 전송 버튼 클릭 이벤트 처리
    view.findViewById(R.id.sendButton).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // 사용자 입력 처리 및 대화 업데이트
        }
    });
}
```

단계 4: SendRequestTask 클래스 구현

1. SendRequestTask 클래스를 구현합니다.
2. 사용자 입력을 Dialogflow 로 보내고 응답을 처리하는 작업을 수행합니다.
3. Runnable 인터페이스를 구현하여 백그라운드 스레드에서 작업을 실행합니다.

```
private class SendRequestTask implements Runnable {

    private String dialogflowKey;

    private String messageText;

    private List<Message> messageList;

    private MessageAdapter mAdapter;



    public SendRequestTask(String dialogflowKey, String messageText,
List<Message> messageList, MessageAdapter mAdapter) {

        this.dialogflowKey = dialogflowKey;

        this.messageText = messageText;

        this.messageList = messageList;

        this.mAdapter = mAdapter;

    }



    @Override

    public void run() {

        // 사용자 입력을 Dialogflow 로 보내고 응답 처리

    }

}
```

단계 5: 전송 버튼 클릭 이벤트 처리

1. 전송 버튼 클릭 이벤트를 처리하는 코드를 추가합니다.
2. 사용자가 입력한 메시지를 가져옵니다.
3. 메시지를 메시지 목록에 추가하고 어댑터에 변경 사항을 알립니다.
4. 사용자 입력을 Dialogflow 로 보내고 응답을 처리하는 `SendRequestTask` 를 실행합니다.

```
view.findViewById(R.id.sendButton).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String messageText = ((EditText)
view.findViewById(R.id.messageEditText)).getText().toString();

        messageList.add(new Message(messageText, true));
        mAdapter.notifyDataSetChanged();

        ((EditText) view.findViewById(R.id.messageEditText)).setText("");

        new SendRequestTask(dialogflowKey, messageText, messageList, mAdapter).run();
    }
});
```

이제 `ChatFragment` 클래스를 사용하여 채팅 화면을 구성하고 사용자와 대화할 수 있습니다.

MessageAdapter 개발자 가이드

`MessageAdapter` 는 채팅 화면에서 메시지를 표시하는 데 사용되는 RecyclerView 의 어댑터입니다. 아래 가이드에서는 `MessageAdapter` 클래스의 코드를 설명하고 사용 방법을 안내합니다.

단계 1: MessageAdapter 클래스 생성

1. `MessageAdapter` 클래스를 생성합니다.
2. `RecyclerView.Adapter` 를 상속받습니다.

```
public class MessageAdapter extends
RecyclerView.Adapter<MessageAdapter.MessageViewHolder> {

    // 코드 작성

}
```

단계 2: MessageViewHolder 클래스 생성

1. `MessageViewHolder` 클래스를 생성합니다.
2. `RecyclerView.ViewHolder` 를 상속받습니다.
3. 메시지를 표시할 뷰 요소들을 선언합니다.

```
public static class MessageViewHolder extends RecyclerView.ViewHolder {

    public TextView messageTextView;

    public LinearLayout messageLinearLayout;



    public MessageViewHolder(View v) {

        super(v);

        messageTextView = v.findViewById(R.id.messageTextView);

        messageLinearLayout = v.findViewById(R.id.messageLinearLayout);

    }

}
```

단계 3: onCreateViewHolder() 메서드 오버라이딩

1. `onCreateViewHolder()` 메서드를 오버라이딩합니다.
2. `item_message` 레이아웃을 inflate 하여 `MessageViewHolder` 를 생성하고 반환합니다.

```
@NonNull

@Override

public MessageViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {

    View v = LayoutInflater.from(parent.getContext())

            .inflate(R.layout.item_message, parent, false);

    MessageViewHolder vh = new MessageViewHolder(v);
```

```
    return vh;
}
```

단계 4: onBindViewHolder() 메서드 오버라이딩

1. `onBindViewHolder()` 메서드를 오버라이딩합니다.
2. 특정 위치에 있는 메시지를 가져와서 뷰 홀더에 표시합니다.
3. 메시지의 송신자에 따라 메시지 배경과 여백을 설정합니다.

```
@Override
public void onBindViewHolder(@NonNull MessageViewHolder holder, int position) {
    Message message = messageList.get(position);

    holder.messageTextView.setText(message.getMessageText());

    LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);

    if (message.isSentByUser()) {
        params.setMargins(100, 10, 10, 10);
holder.messageTextView.setBackgroundResource(R.drawable.user_message_background);
    } else {
        params.setMargins(10, 10, 100, 10);
holder.messageTextView.setBackgroundResource(R.drawable.bot_message_background);
    }
```

```
        holder.messageLinearLayout.setLayoutParams(params);
    }
```

단계 5: getItemCount() 메서드 오버라이딩

1. `getItemCount()` 메서드를 오버라이딩하여 메시지 목록의 크기를 반환합니다.

```
@Override
public int getItemCount() {
    return messageList.size();
}
```

이제 `MessageAdapter` 클래스를 사용하여 RecyclerView 에 메시지를 표시할 수 있습니다. 필요한 경우 코드를 확장하여 추가 기능을 구현할 수 있습니다.

Message 클래스 개발자 가이드

`Message` 클래스는 채팅 메시지의 정보를 담는 모델 클래스입니다. 아래 가이드에서는 `Message` 클래스의 코드를 설명하고 사용 방법을 안내합니다.

단계 1: Message 클래스 생성

1. `Message` 클래스를 생성합니다.
2. `messageText` 와 `sentByUser` 라는 두 개의 멤버 변수를 선언합니다.

```
public class Message {
    private String messageText;
    private boolean sentByUser;
```

```
    public Message(String messageText, boolean sentByUser) {
        this.messageText = messageText;
        this.sentByUser = sentByUser;
    }

    // Getter 와 Setter 메서드 작성
}
```

단계 2: 멤버 변수와 Getter/Setter 메서드

1. `messageText` 와 `sentByUser` 멤버 변수의 Getter 와 Setter 메서드를 작성합니다.

```
public String getMessageText() {
    return messageText;
}

public void setMessageText(String messageText) {
    this.messageText = messageText;
}

public boolean isSentByUser() {
    return sentByUser;
}

public void setSentByUser(boolean sentByUser) {
    this.sentByUser = sentByUser;
```

```
}
```

이제 `Message` 클래스를 사용하여 채팅 메시지의 정보를 저장하고 조작할 수 있습니다. 필요한 경우 코드를 확장하여 추가 기능을 구현할 수 있습니다.

SendRequestTask 클래스 개발자 가이드

`SendRequestTask` 클래스는 Dialogflow 와의 통신을 위해 HTTP POST 요청을 보내는 작업을 수행하는 클래스입니다. 아래 가이드에서는 `SendRequestTask` 클래스의 코드를 설명하고 사용 방법을 안내합니다.

단계 1: SendRequestTask 클래스 생성

1. `SendRequestTask` 클래스를 생성합니다.
2. `dialogflowKey`, `messageText`, `messageList`, `mAdapter` 라는 네 개의 멤버 변수를 선언합니다.

```
public class SendRequestTask implements Runnable {

    private static final String TAG = "SendRequestTask";

    private String dialogflowKey;

    private String messageText;
```

```
    private List<Message> messageList;

    private MessageAdapter mAdapter;



    public SendRequestTask(String dialogflowKey, String messageText,
List<Message> messageList, MessageAdapter mAdapter) {

        this.dialogflowKey = dialogflowKey;

        this.messageText = messageText;

        this.messageList = messageList;

        this.mAdapter = mAdapter;

    }



    @Override

    public void run() {

        try {

            // Send HTTP POST request to Dialogflow

            // Handle the response

        } catch (Exception e) {

            Log.e(TAG, "Exception caught: ", e);

        }

    }

}
```

단계 2: run 메서드 작성

1. `run` 메서드 내에서 HTTP POST 요청을 보내고 응답을 처리합니다.

```
@Override
```

```
public void run() {
	try {
		URL url = new
URL("https://api.dialogflow.com/v1/query?v=20150910");
		HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
		conn.setRequestMethod("POST");
		conn.setRequestProperty("Authorization", "Bearer " + dialogflowKey);
		conn.setRequestProperty("Content-Type", "application/json;
charset=utf-8");
		conn.setDoOutput(true);

		String jsonRequest = "{ \"query\": \"" + messageText + "\", \"lang\":
\"en\", \"sessionId\": \"1234567890\" }";
		OutputStream os = conn.getOutputStream();
		os.write(jsonRequest.getBytes("UTF-8"));
		os.close();

		BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));
		StringBuilder sb = new StringBuilder();
		String line;
		while ((line = br.readLine()) != null) {
			sb.append(line + "\n");
		}
		conn.disconnect();

		// Handle the response on the main thread
		String response = sb.toString();
		messageList.add(new Message(response, false));
		mAdapter.notifyDataSetChanged();

	} catch (Exception e) {
		Log.e(TAG, "Exception caught: ", e);
	}
}
```

이제 `SendRequestTask` 클래스를 사용하여 Dialogflow 와의 통신을 수행할 수 있습니다. 필요한 경우 코드를 확장하여 추가 기능을 구현할 수 있습니다.

ImgGenFragment 클래스 개발자 가이드

이 개발자 가이드는 `ImgGenFragment` 클래스의 역할과 주요 기능을 설명하며, 해당 클래스를 사용하는 방법에 대해 안내합니다.

1. ImgGenFragment 클래스

`ImgGenFragment` 클래스는 이미지 생성 화면을 관리하는 Fragment 입니다. 주요 기능은 다음과 같습니다:

- 사용자의 텍스트 입력에 기반하여 이미지를 생성하고 표시
- 이미지 생성 상태에 따라 UI 업데이트

사용 방법

1. `ImgGenFragment` 클래스를 생성합니다.
2. `onCreateView` 메서드에서 레이아웃 파일을 inflate 하고 필요한 UI 요소들을 초기화합니다.
3. 필요한 변수 및 객체를 선언합니다.
4. `onViewCreated` 메서드에서 다음 동작들을 수행합니다:
    - 이미지 생성을 위해 필요한 텍스트 데이터를 가져옵니다.
    - 이미지 생성 상태에 따라 UI 를 업데이트하고 이미지를 표시합니다.
    - `refreshButton` 을 클릭할 때 실행되는 코드를 작성합니다:
        - 텍스트 데이터가 준비되지 않은 경우 적절한 토스트 메시지를 표시합니다.
        - 텍스트 데이터가 준비된 경우 이미지 생성을 위한 작업을 실행합니다.
        - 이미지 생성이 완료된 경우 UI 를 업데이트하고 이미지를 표시합니다.
5. 필요에 따라 `updateDescription` 메서드를 호출하여 텍스트 데이터와 이미지를 업데이트합니다.
6. 필요한 경우 `reset` 메서드를 호출하여 변수 및 상태를 초기화합니다.

2. getImageFromDescription 메서드

`getImageFromDescription` 메서드는 텍스트 데이터를 기반으로 이미지를 생성하는 비동기 작업을 수행합니다.

사용 방법

1. `getImageFromDescription` 메서드를 호출하고 텍스트 데이터를 전달합니다.
2. 내부에서 Python 인터프리터를 시작하고 `script` 모듈을 가져옵니다.
3. `set_openai_api_key` 메서드를 호출하여 OpenAI API 키를 설정합니다.
4. `main` 메서드를 호출하여 텍스트 데이터를 기반으로 이미지를 생성합니다.
5. 반환된 이미지 데이터를 디코딩하고 `Bitmap` 객체로 변환합니다.
6. 필요한 경우 추가적인 작업을 수행한 후, 생성된 `Bitmap` 객체를 반환합니다.

3. reset 메서드

`reset` 메서드는 변수 및 상태를 초기화하는 역할을 합니다.

사용 방법

1. `reset` 메서드를 호출하여 변수 및 상태를 초기화합니다.
2. 필요한 경우 UI 요소들을 초기화하거나 특정 동작을 수행합니다.

4. Auto Execution (자동 실행) `ImgGenFragment` 클래스에는 주기적으로 이미지 생성 작업을 자동으로 실행하는 기능인 Auto Execution 이 있습니다. 다음은 Auto Execution 의 사용 방법입니다:
- `startAutoExecution()` 메서드를 호출하여 Auto Execution 을 시작합니다. 이 메서드는 이미지 생성 작업을 일정 간격으로 주기적으로 실행합니다.
- Auto Execution 의 간격은 `interval` 변수를 통해 설정할 수 있으며, 기본값은 1 초(1000ms)입니다. 필요에 따라 원하는 간격으로 설정할 수 있습니다.
- Auto Execution 은 백그라운드 스레드에서 실행되며, 액티비티의 상태에 영향을 받지 않습니다. 그러나 액티비티가 일시 중지(pause)되면 Auto Execution 도 중지됩니다.
- Auto Execution 을 중지하려면 `stopAutoExecution()` 메서드를 호출합니다.

5. ViewModel 사용 `ImgGenFragment` 클래스는 `ImgGenViewModel` 을 사용하여 데이터를 관리할 수 있습니다. ViewModel 은 액티비티 또는 프래그먼트의 생명주기와 독립적으로 데이터를 보관하고 관리하는 데 도움이 됩니다. 다음은 ViewModel 의 사용 방법입니다:
- `ViewModelProvider` 를 사용하여 `ImgGenViewModel` 객체를 생성합니다.
- `ImgGenViewModel` 객체를 초기화하고 필요한 데이터를 저장하거나 가져옵니다.

- 필요한 경우 `ImgGenViewModel` 객체를 옵저버로 등록하여 데이터의 변경을 감지하고, 적절한 액션을 수행할 수 있습니다.

ImgGenViewModel 개발자 가이드

소개: ImgGenViewModel 은 이미지 생성 기능을 담당하는 ViewModel 입니다. 이 클래스는 다음과 같은 기능을 수행합니다:

- 사용자 입력을 기반으로 이미지 생성
- 생성된 이미지를 관리 및 제공

사용 방법: ImgGenViewModel 클래스를 사용하기 위해서는 다음과 같은 단계를 따르세요:

1. ImgGenViewModel.kt 파일을 앱의 소스 코드에 추가합니다.
2. 필요한 라이브러리와 종속성을 추가합니다.
3. ImgGenViewModel 객체를 생성합니다.
4. generateImage() 메서드를 호출하여 이미지 생성을 시작합니다.
5. imageBitmap LiveData 를 관찰하여 생성된 이미지를 받아옵니다.

필드: ImgGenViewModel 클래스에는 다음과 같은 필드들이 있을 수 있습니다:

- `oPENAI_API_KEY`: OpenAI API 키를 저장하는 문자열 필드입니다.
- `_imageBitmap`: 생성된 이미지를 관리하는 MutableLiveData<Bitmap> 객체입니다.
- `imageBitmap`: 생성된 이미지를 외부에 노출하기 위한 LiveData<Bitmap> 객체입니다.

메소드: ImgGenViewModel 클래스에는 다음과 같은 주요 메소드들이 있을 수 있습니다:

- `generateImage(description: String, context: Context)`: 사용자 입력을 기반으로 이미지 생성을 시작합니다.
- `executeChaquopyInBackground(description: String, context: Context)`: 백그라운드에서 Chaquopy 를 사용하여 이미지 생성 작업을 수행합니다.

단국대학교 DANKOOK UNIVERSITY

StoryAdapter 개발자 가이드

소개: StoryAdapter 는 ViewPager2 와 함께 사용되는 어댑터 클래스입니다. 이 클래스는 여러 프래그먼트를 표시하고 스와이프를 통해 전환하는 데 사용됩니다. StoryAdapter 는 다음과 같은 기능을 수행합니다:

- ViewPager2 와 함께 사용하여 여러 프래그먼트를 관리합니다.
- 프래그먼트의 생성과 관리를 담당합니다.

사용 방법: StoryAdapter 클래스를 사용하기 위해서는 다음과 같은 단계를 따르세요:

1. StoryAdapter.kt 파일을 앱의 소스 코드에 추가합니다.
2. 필요한 라이브러리와 종속성을 추가합니다.
3. StoryAdapter 객체를 생성합니다.
4. ViewPager2 와 StoryAdapter 를 연결합니다.

필드: StoryAdapter 클래스에는 다음과 같은 필드들이 있을 수 있습니다:

- `fragmentManager`: Fragment 를 관리하는 FragmentManager 객체입니다.
- `lifecycle`: ViewPager2 의 Lifecycle 을 나타내는 Lifecycle 객체입니다.

메소드: StoryAdapter 클래스에는 다음과 같은 주요 메소드들이 있을 수 있습니다:

- `getItemCount(): Int`: 전체 페이지 수를 반환하는 메소드입니다.
- `createFragment(position: Int): Fragment`: 현재 페이지의 위치(position)에 따라 알맞은 Fragment 를 생성하여 반환하는 메소드입니다.

단국대학교 DANKOOK UNIVERSITY

StoryActivity 개발자 가이드

소개: `StoryActivity`는 앱의 이야기 화면을 담당하는 액티비티입니다. 이 클래스는 ViewPager2 와 함께 사용되며, 다양한 프래그먼트를 표시하고 스와이프를 통해 전환할 수 있습니다. `StoryActivity`는 다음과 같은 기능을 수행합니다:

- ViewPager2 를 초기화하고 프래그먼트를 관리합니다.
- 이벤트 처리 및 화면 전환을 담당합니다.

사용 방법: `StoryActivity` 클래스를 사용하기 위해서는 다음과 같은 단계를 따르세요:

1. `StoryActivity.kt` 파일을 앱의 소스 코드에 추가합니다.
2. 필요한 라이브러리와 종속성을 추가합니다.
3. `StoryActivity` 클래스의 필드와 메소드를 사용하여 앱의 기능을 구현합니다.

필드: `StoryActivity` 클래스에는 다음과 같은 필드들이 있을 수 있습니다:

- `storyFragment`: 이야기 프래그먼트를 나타내는 StoryFragment 객체입니다.
- `viewPager`: ViewPager2 객체로, 프래그먼트를 표시하고 스와이프를 통해 전환하는 데 사용됩니다.
- `storyAdapter`: `StoryAdapter` 객체로, ViewPager2 와 프래그먼트 간의 연결을 담당합니다.
- `circleIndicator3`: CircleIndicator3 객체로, ViewPager2 의 현재 페이지를 표시하는 인디케이터입니다.

메소드: `StoryActivity` 클래스에는 다음과 같은 주요 메소드들이 있습니다:

- `onCreate(savedInstanceState: Bundle?)`: 액티비티가 생성될 때 호출되는 메소드로, 초기화 작업과 이벤트 처리를 수행합니다.
- `getViewPager(): ViewPager2`: ViewPager2 객체를 반환하는 메소드입니다.
- `receiveData(data: String)`: 데이터를 전달받아 StoryFragment 에 설정하는 메소드입니다.
- `getStoryAdapter(): StoryAdapter`: StoryAdapter 객체를 반환하는 메소드입니다.
- `initializeAndNavigateToStoryFragment()`: StoryFragment 를 초기화하고 해당 프래그먼트로 이동하는 메소드입니다.

단국대학교 DANKOOK UNIVERSITY

중첩 클래스 `StoryAdapter`: `StoryActivity` 클래스에는 중첩 클래스로 `StoryAdapter`가 포함되어 있습니다. 이 클래스는 ViewPager2 와 함께 사용되는 어댑터로, 프래그먼트의 생성과 관리를 담당합니다. `StoryAdapter` 클래스에는 다음과 같은 주요 메소드들이 있습니다:

- `addFragment(fragment: Fragment)`: 프래그먼트를 추가하는 메소드입니다.
- `getStoryFragment(): StoryFragment?`: StoryFragment 객체를 반환하는 메소드입니다.
- `getItemCount(): Int`: 전체 페이지 수를 반환하는 메소드입니다.
- `createFragment(position: Int): Fragment`: 현재 페이지의 위치(position)에 따라 알맞은 Fragment 를 생성하여 반환하는 메소드입니다.
- `getImgGenFragment(): ImgGenFragment?`: ImgGenFragment 객체를 반환하는 메소드입니다.
- `getChoiceFragment(): ChoiceFragment?`: ChoiceFragment 객체를 반환하는 메소드입니다.

ChoiceFragment 개발자 가이드

소개

`ChoiceFragment`는 사용자에게 선택지를 제공하고 선택된 항목을 처리하는 데 사용되는 안드로이드 프래그먼트입니다. 이 개발자 가이드에서는 `ChoiceFragment` 클래스의 주요 기능과 사용법에 대해 설명합니다.

기능

- 선택지 버튼을 화면에 표시합니다.
- 선택된 항목을 처리하기 위해 다른 프래그먼트로 화면을 전환합니다.
- 선택지를 번역하여 다른 언어로 표시할 수 있습니다.

사용법

1. `ChoiceFragment` 클래스를 `Fragment`를 상속하는 방식으로 생성합니다.
2. 필요한 변수와 뷰 요소들을 선언합니다. (`command1Button`, `command2Button`, `command3Button`, `choiceButton`, `refreshButton` 등)

3. `onCreateView` 메서드를 재정의하고 프래그먼트의 레이아웃을 설정합니다. 여기서 뷰 요소들을 초기화하고 클릭 리스너를 설정합니다.
4. 선택지 버튼을 클릭했을 때의 동작을 처리하는 `gostart` 메서드를 구현합니다. 선택된 항목을 다른 프래그먼트로 전달하고 화면을 전환합니다.
5. 선택지 버튼의 텍스트를 번역하는 `translateCommands` 메서드를 구현합니다. 번역에는 `Python` 모듈을 사용합니다.
6. 필요한 경우 선택지를 수정할 수 있는 다이얼로그를 표시하는 `showEditCommandsDialog` 메서드를 구현합니다.
7. 필요한 경우 선택지를 선택할 수 있는 다이얼로그를 표시하는 `showChoiceDialog` 메서드를 구현합니다.
8. `ChoiceListener` 인터페이스를 정의하여 선택된 항목을 처리하는 콜백 메서드를 제공합니다.
9. `ChoiceFragment`의 인스턴스를 생성하고 `FragmentTransaction`을 사용하여 화면에 추가합니다.

이 외의 추가적인 코드가 있다면, 알려주시면 해당 코드에 대한 정보를 이어서 추가하도록 하겠습니다.