用户身份验证与管理:
通过 Firebase Authentication 和 Google Sign-In 来实现用户注册、登录及退出登录功能。

- `auth_methods.dart` 中使用 FirebaseAuth 和 GoogleSignIn 来完成 Google 登录，并在 Firestore 中为新用户创建对应的用户数据文档。
- 如果用户已经登陆过，则在应用启动时会根据 `StreamBuilder` 监听用户状态，自动显示已登录用户的主界面（HomeScreen）。

会议记录存储:
使用 Cloud Firestore 来存储用户的会议历史记录。

- `firestore_methods.dart` 中的 `addToMeetingHistory` 会在用户创建或加入会议后，将会议名称和创建时间记录到 Firestore 对应用户的集合中。
- `HistoryMeetingScreen` 通过 `StreamBuilder` 监听 Firestore 中的会议记录变化，并在界面上显示会议历史列表。

视频会议功能:

使用**flutter_zoom_videosdk插件**实现，我还搜索了另外一种插件Zoom Meeting SDK，实现起来更简单，但感觉不适用于我们的项目。

区别点与适用场景：

1. **底层 SDK 不同**:
    - **flutter_zoom_sdk** 插件基于 Zoom **Meeting SDK**。它是 Zoom 官方为在线会议场景提供的一套标准化解决方案，通常包含类似于用户熟悉的 Zoom 会议界面和流程，如使用会议号（Meeting ID）和密码加入会议。
    - **flutter_zoom_videosdk** 插件基于 Zoom **Video SDK**。Video SDK 是 Zoom 为开发者提供的更灵活的音视频会议基础能力，不再局限于传统 Zoom 会议ID/密码的方式，而是通过 Session、Token 等方式加入自定义的音视频互动场景。它不强制使用 Zoom 原生 UI，适合想要深度定制界面和交互的开发者。
2. **UI 定制与灵活性**:
    - **Meeting SDK (flutter_zoom_sdk)**：偏向于让你快速集成类似官方 Zoom 客户端的会议体验。UI 和功能较为固定，适合想要快速集成标准 Zoom 会议功能的场景，用户可快速上手，有较多官方的 UI 元素可用。
    - **Video SDK (flutter_zoom_videosdk)**：提供底层的音视频能力，UI、交互和流程几乎完全由你定制。虽然开发上更灵活，但也意味着需要花更多精力打造自己的 UI 界面和会议逻辑。
3. **使用方式与鉴权方式**:
    - **Meeting SDK**：通常使用 Meeting ID 与密码，或 ZAK Token 等方式加入特定 Zoom 会议室。更贴近用户日常使用 Zoom 的方式。
    - **Video SDK**：通过 Session Name 和 Session Token (JWT) 等方法加入一个自定义 Session。更适合创造自己的虚拟会议场景、互动房间或直播教室等自定义场景。

如果你想要快速接入 Zoom 的在线会议功能，并且界面和交互接近官方 Zoom 会议客户端，无需太多自定义，那么 **flutter_zoom_sdk (Meeting SDK)** 会更好用，因为集成流程相对简单，用户也更容易上手。

如果你的应用场景需要高度定制的互动音视频功能（不一定是传统意义上的「会议」），并希望自由地设计会议UI、业务逻辑等，那么 **flutter_zoom_videosdk** 更合适。它提供更灵活的接口，让你可以构建出与传统 Zoom 会议形态差异很大的自定义化视频互动产品，但这需要更多开发工作量和UI设计。

在这种情况下，我会更推荐使用 **Zoom Video SDK（flutter_zoom_videosdk）** 而非传统的 Zoom Meeting SDK（如 flutter_zoom_sdk），理由如下：

1. **高度灵活的业务逻辑与流程定制**：
   Zoom Video SDK 将音视频能力抽象为底层能力，而不是限制在标准化的 Zoom 会议流程中。这意味着你可以根据自己的业务场景自由设计会议加入流程、身份鉴权方式（Session Token）、界面元素、权限管理、以及加密策略的动态调整，而不会被传统会议模式（Meeting ID、会议密码、固定UI等）所限制。
2. **深度集成高级安全与加密策略**：
   在 Video SDK 模型下，你可以更容易地拦截、处理和定制数据传输过程。尽管需要自己实现部分加密和安全策略，但这也给予了你更大的自由度去整合你所列出的高级安全功能（如动态加密算法切换、基于AI的流量监测和风险识别、敏感信息检测与加密存储等）。Meeting SDK 尝试保持与原生 Zoom 客户端类似的体验，其数据交互过程和安全策略相对固定，不利于如此深度和定制化的安全扩展。
3. **定制UI与AI辅助功能整合更为顺畅**：
   你提到的智能笔记、敏感信息检测、文档审阅、行为分析、态势感知可视化等功能，并不局限于"在线会议"传统场景。这些功能需要将AI、NLP、安全检测与可视化分析无缝嵌入会议应用中。Video SDK 的核心定位是提供底层音视频通讯能力，你在此基础上构建自己的完整逻辑和UI，这与高自由度的AI分析和可视化场景天然契合。
4. **适用于创新与研发场景**：
   Meeting SDK 更像是快速集成标准化视频会议的现成方案，适用于想迅速拥有"像Zoom"一样的会议体验的应用。而你所描述的功能需求远远超出传统会议场景，更多是想要构建一个高度定制化、安全化、智能化的交互系统。Video SDK 的定位正是为这种创新型场景而设计，更易融入自定义AI模型、深度加密逻辑和自定义权限、角色管理、文档审查模块。

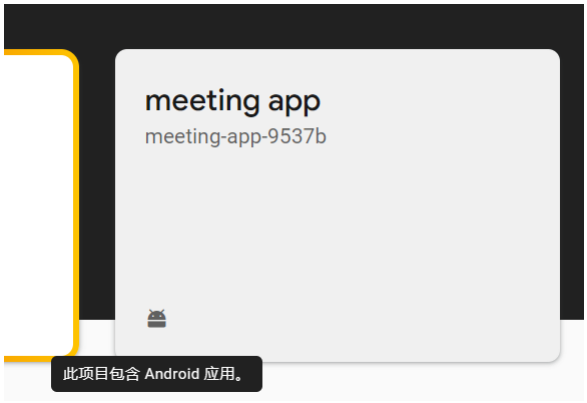
结论：
选择 **flutter_zoom_videosdk** 更有利于实施你计划中的那些高级安全与智能功能，因为它提供了更灵活的架构、底层API和可定制空间，使你能够深度融合高级加密策略、AI分析、智能笔记、文档审阅以及各种安全监控和可视化功能，从而构建出一套真正高度可定制、智能、安全的会议与协作平

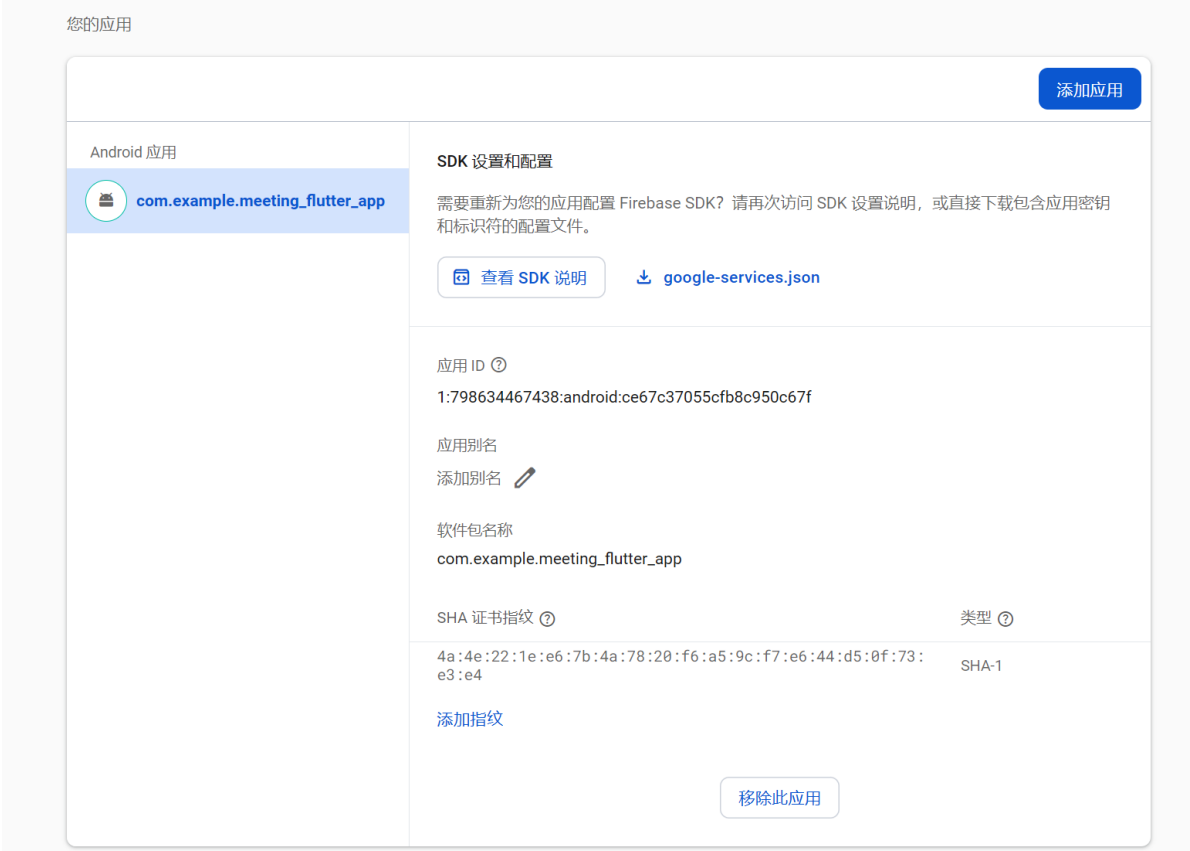
lib目录下的native是flutter_zoom_videosdk插件提供的代码接口示例，有需要的功能可以去里面看看

目前的问题是插件和安卓版本或者jdk版本兼容问题（在这之前我已经调了很多不兼容了）

代码里面我已经连到了我的zoom账号，firebase也连到了我这边，不知道你们能不能用

还有一个问题就是，我的flutter找不到firebase项目，但是在firebase终端可以显示创建的项目，所以我只把firebase连到了android上，也就是android/app/google-services.json文件，里面的信息就是我的firebase。





连安卓模拟机的话，这样子弄，把系统防火墙关了，然后新建一个flutter项目，开/关梯子都试试，修改以下文件：

android/bulid.gradle:

```
buildscript {
    ext.kotlin_version = '1.7.10'
    repositories {
        maven { url 'https://maven.aliyun.com/repository/public/' }
        maven { url 'https://maven.aliyun.com/repository/spring/'}
        maven { url 'https://maven.aliyun.com/repository/google/'}
        maven { url 'https://maven.aliyun.com/repository/gradle-plugin/'}
        maven { url 'https://maven.aliyun.com/repository/spring-plugin/'}
        maven { url 'https://maven.aliyun.com/repository/grails-core/'}
        maven { url 'https://maven.aliyun.com/repository/apache-snapshots/'}
        google()
        mavenCentral()
    }

    dependencies {
        classpath 'com.android.tools.build:gradle:8.0.2'
        classpath 'com.google.gms:google-services:4.4.2'
```

```
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    }
}




allprojects {
    repositories {
        maven { url 'https://maven.aliyun.com/repository/public/' }
        maven { url 'https://maven.aliyun.com/repository/spring/'}
        maven { url 'https://maven.aliyun.com/repository/google/'}
        maven { url 'https://maven.aliyun.com/repository/gradle-plugin/'}
        maven { url 'https://maven.aliyun.com/repository/spring-plugin/'}
        maven { url 'https://maven.aliyun.com/repository/grails-core/'}
        maven { url 'https://maven.aliyun.com/repository/apache-snapshots/'}
        google()

    }
}
rootProject.buildDir = "../build"

subprojects {
    project.buildDir = "${rootProject.buildDir}/${project.name}"
}
subprojects {
    project.evaluationDependsOn(":app")
}

tasks.register("clean", Delete) {
    delete rootProject.buildDir
}
```

android\gradle\wrapper\gradle-wrapper.properties:

加上这一行:

```
distributionUrl=https\://services.gradle.org/distributions/gradle-8.3-all.zip
```