Campus Connect项目说明文档

项目简介

Campus Connect是一个校园信息发布平台，有分享身边趣事、交流校园动态和发布二手交易信息等多种多样的功能。

前后端部署说明

详见`forum/README.md`中的**Getting Started**。

操作指南

详见`forum/README.md`中的 **Campus Connect操作指南**

设计与实现

前后端实现

该项目基于**Flutter**开发，并使用**Firebase**作为后端数据库和存储解决方案，同时利用Firebase提供的认证服务和云函数来处理后端逻辑。

- **Flutter**：跨平台的移动应用开发框架，用于构建美观且高性能的移动应用界面。
- **Firebase**：全面的移动和Web应用开发平台，提供多种云端服务和工具，简化开发和管理过程。
    - **Firebase数据库**：用于存储和同步应用数据的云端NoSQL数据库。
    - **Firebase身份验证**：提供安全的用户认证和身份管理功能，确保应用数据的访问权限。
    - **Firebase存储**：用于在云端安全地存储用户上传的图片等文件。
    - **Firebase云函数**：无服务器的后端代码托管服务，用于执行自定义逻辑和处理请求。

用戶相关功能

功能点	设计与实现
注册与登录	1. **注册和密码设置**：使用`Firebase`和`FirebaseUIAuth`库进行身份验证，利用`SignInScreen`组件中实现注册与登录页面。 2. **修改用户名、头像、简介和密码**：在修改页面中，保存按钮被点击后，将更新用户的信息，并将其保存到Firebase的Cloud Firestore数据库中，以供后续使用。 3. **关注和取消关注、关注列表、信息界面显示已关注**：点击关注按钮时，根据当前的关注状态，更新Cloud Firestore数据库中用户的关注列表，在关注页面中以`ListView`呈现，在信息列表或信息详情界面显示关注。 4. **实现用户屏蔽功能**：点击屏蔽按钮时，更新Cloud Firestore数据库中用户的屏蔽列表，在信息查看界面进行分类浏览时排除已屏蔽用户的内容。
私信	1. 设计数据模型：设计用于存储聊天消息的数据模型。使用 Firebase 实时数据库来存储这些消息。在数据模型中，定义每个消息的属性，如消息 ID、发送者 ID、接收者 ID、消息时间戳和消息文本。 2. 实现聊天 UI：使用 Flutter 构建聊天 UI。使用 Flutter 的 Widgets 来构建聊天消息列表和输入框等 UI 组件。在聊天 UI 中，需要使用实时数据库 API 来获取存储在 Firebase 中的消息，并将其显示在屏幕上。还需要使用实时数据库 API 将新的消息保存到 Firebase 中。 3. 实现消息发送：使用 Flutter 和 Firebase 实时数据库代码来发送和接收消息。在 Flutter 应用程序中，使用实时数据库 API 来发送和接收消息。在发送消息时，将消息数据保存到 Firebase 中，并在 Firebase 数据库中设置适当的监听器以接收新消息的通知。
用户个人页面	1. **关注和取消关注、关注列表、信息界面显示已关注**：点击关注按钮时，根据当前的关注状态，更新Cloud Firestore数据库中用户的关注列表，在关注页面中以`ListView`呈现，在信息列表或信息详情界面显示关注。 2. **实现用户屏蔽功能**：点击屏蔽按钮时，更新Cloud Firestore数据库中用户的屏蔽列表，在信息查看界面进行分类浏览时排除已屏蔽用户的内容。

信息发布

功能点	设计与实现
发布图片+文字	通过调用`ImagePicker`库选择图片文件，并将其上传到Firebase Storage中保存。在Cloud Firestore数据库中创建新的帖子文档，包含图片的下载URL和帖子内容等信息。
发布视频+文字	通过调用`ImagePicker`库选择视频文件，并将其上传到Firebase Storage中保存。在Cloud Firestore数据库中创建新的帖子文档，包含视频的下载URL和帖子内容等信息。
添加位置信息	使用`Geolocator`库获取当前设备的位置信息，并将经纬度保存在Cloud Firestore数据库中的帖子文档中。
添加信息类型	通过在界面中选择标签（"校园资讯"、"二手交易"），将选定的标签保存在Cloud Firestore数据库中的帖子文档中。
支持markdown格式	使用`TextFormField`组件在界面中输入Markdown格式的文本，并将其保存在Cloud Firestore数据库中的帖子文档中。支持多级标题、列表、文字加粗
修改字体大小、颜色、粗细	通过使用`Slider`组件调整字体大小，使用`BlockPicker`选择字体颜色。将选定的字体大小、颜色等信息保存在Cloud Firestore数据库中的帖子文档中。
调用相机拍照、录影	使用`ImagePicker`库调用相机功能，通过`ImageSource.camera`参数指定使用相机进行拍照或录影。 获取拍照或录影的结果，使用`FirebaseStorage`库将图片或视频上传到Firestore Storage，获取相应的下载链接。

信息查看与操作

功能点	设计与实现
按属性分类浏览	根据Cloud Firestore数据库中已关注的发布者、点赞评论量、信息类型的变量过滤显示的帖子
按发布时间、点赞量、评论量排序	根据Cloud Firestore数据库中发布时间、点赞量、评论量的变量排序显示的帖子
点赞或取消点赞，收藏或取消收藏	根据Cloud Firestore数据库，更新帖子的点赞数、收藏状态和点赞状态，收藏的帖子在Favorites页面中以ListView显示。
评论	Firebase Firestore监听帖子的评论数据流。 使用`StreamBuilder`构建评论列表，并根据评论数据的数量动态生成评论项，包括评论者的姓名、评论内容和评论时间。
分享	通过使用`share`库中的`Share.share()`方法实现，接受要分享的文本内容和可选的主题作为参数，然后触发分享操作。
信息详情页面	点击帖子后进入到信息详情页面，信息详情页面用户可以查看帖子的详细信息，并在评论框中输入评论内容，然后点击发送按钮将评论保存到数据库中，并更新评论。

通知

功能点	设计与实现
收到私信、信息被点赞、回复、关注用户发布新信息时接收通知	1. 每次进行相应的操作时, 通过向https://fcm.googleapis.com/fcm/send发送一个http.post的请求, firebase成功收到后,会发送一个FCM(Firebase cloud message), 到对应的设备上. 2. fcmToken是在注册时记录下来的,一个设备有唯一的fcmToken 3. 接收到信息后, ChatProvider.dart会对收到的`RemoteMessage message`进行处理,然后调用`showNotification`函数进行展示 4. 最后使用`flutterLocalNotificationsPlugin`库进行消息的显示
个人中心界面进入通知消息列表，查看通知详情、回复私信	1. 判断当前是否有用户已登录，如果没有则显示一个“未登录”页面。 2. 如果用户已登录，则在Firestore数据库中查询当前用户的消息记录。这里使用了Firestore的实时查询功能，即使用`StreamBuilder`来监听查询结果的变化。 3. 如果查询出错，则在页面中显示错误信息。 4. 如果正在等待查询结果，则显示一个进度指示器。 5. 如果查询结果不为空，则遍历每条消息记录，并将其展示在一个`ListView`中。 6. 对于每条消息记录，如果其标题为“您的动态收到了一条点赞”、“您收到了一条评论”或“您的动态收到了一条回复”，则不需要跳转到具体的页面，并在控制台中输出一条日志信息。否则，解析消息记录中的UID，并使用Navigator跳转到私信页面。

搜索

功能点	设计与实现
搜索控件	1. 获取用户在搜索框中输入的关键词。 2. 对每个关键词，分别在“users”集合和“posts”集合中进行搜索。 3. 在“users”集合中搜索时，查询集合中的所有文档，对于每个文档，如果其“name”字段包含该关键词，则将该用户信息添加到结果集。 4. 在“posts”集合中搜索时，查询集合中的所有文档，对于每个文档，如果其“markdownText”字段或“tag”字段包含该关键词，或该文档的作者在前面步骤中已经被搜索到，那么将该文档的UID添加到结果集。 5. 将结果集中的重复项去除，然后将结果分别展示在“Users”和“Posts”两个子列表中。
多关键词联合搜索	1. 在搜索框中输入关键词后，需要将关键词按空格分割为多个单独的关键词，并对每个关键词分别进行搜索。 2. 由于搜索是异步的操作，需要在搜索完成后调用`setState`方法来更新搜索结果的显示。

UI 设计与交互

实现了以下页面：信息浏览界面、信息详情界面、评论界面、信息发布界面、个人中心(设置)界面、收藏信息列表界面、个人主页及其他用户主页(可按时间线展示用户发布信息)、私信界面、关注列表界面

在内容加载或界面切换时，我们使用了动画效果，提升了用户体验。

分工与贡献情况

- 陈敬文：主要负责Post相关功能的实现
- 江灿：主要负责私信、通知、搜索等功能的实现
- 田晋恺：主要负责用户相关功能的实现