设计文档

白润声 王皓雯

关键功能

笔记选择页面的呈现

文件显示

为了能够合理的方便用户检索已有的笔记文件，为了选择页面的多样化布局，我们需要在数据列表中记录多种文件的信息，并用`RecyclerView`来实现页面的多样化布局。首先，我们创建了`File`类以记录文件数据，`FileType`以记录文件种类。文件种类包括：文件夹、笔记、占位符和添加按钮。这些信息储存在`Adapter`的`file_list`之中，对文件增删改查的操作都依据于该数组。其后，根据记录下来的信息，我们在`RecyclerView`中为每个`item`绑定了不同的点击响应函数、`layout`文件和数据，从而实现了多样化布局。

页面跳转

如何将树状的文件结构在数据库中储存，并且相对方便检索呢？实现中，我们为每个文件（笔记或文件夹）记录了其所在路径，路径包括其所在位置和标题，作用相当于一个文件的唯一标识。

因此在前端中，我们首先在选择页面对应的`Fragment`中用一成员变量记录当前的路径。对于文件夹或返回上级按钮时，在点击时更新该路径；其后，根据更新到的路径，在后端寻找与该路径匹配的文件夹（如果路径`为root`则记为`None`），找到后继续查找以该文件夹为父亲（或父亲为空）的文件，遍历该结果并返回；最后根据后端返回的信息，在`Response中`创建`file_list`，并将当前`Adapter`中`file_list`替换为新获取的list即可。

而对于笔记相对简单，只需要获取到当前路径，并传递该数据并用`intent`打开笔记`Activity`即可。需要额外强调一点的是，由于关闭笔记`Activity`之后，前端页面并不会重新访问后端，也不会刷新页面需要更新的数据，因此需要使用`StartActiviyForResult`，在笔记`Activity`结束时传回更改的内容（如标题）并手动更新页面。

文件搜索

关于文件搜索，主要包含两方面的工作，一是前端的显示和数据绑定，二是后端的搜索匹配方式。

在后端中，拿到用户的关键词序列，我们首先将关键词通过空格`split`为多个关键词。其次，对每个关键词，我们利用`fuzzywuzzy`库中最佳子串匹配接口，获取每个关键词和该用户笔记的内容的匹配度，并记录不同每个笔记对不同关键词匹配度的最大值和平均值，以二者的加权和作为最终的匹配得分。该方法相当于主要以某个关键词的匹配度为准，辅以与整体的关系来判断最终匹配度。最终，获取到所有笔记中得分高于某个`threshold`（当前为60分，满分100）的部分，并返回他们的信息到前端。需要额外强调，此次返回不需要返回文件类型，因为均为笔记；而需要额外返回文件路径，因为搜索结果是全局的，不能根据当前路径加标题的方式获取笔记并

而前端获取到该队列后，并不能和文件跳转做同样的操作。在替换`file_list`之外，我们还需要做如下操作：1. 更新主页面状态，将`in_search`置为真，并将返回上级按钮替换为退出搜索按钮；2. 文件跳转时，若处于搜索状态，则直接获取`path_list`中对应位置的路径，从而打开文件。

分工：白润声

笔记详情页面的呈现

为了让笔记可以实时呈现文字、图像与音频，并且用户可以随时进行编辑、添加、删除等功能的交互，我们选择使用`RecyclerView`来显示笔记的内容，其中每一类内容对应不同类型的`NoteItem`。`NoteItem`的属性包括类型`type`与内容`content`。`content`分别用来记录文字的内容、图像的`url`，和音频文件的`url`。不同类型的`NoteItem`有不同的`ViewHolder`来呈现，分别是自定义类`TextViewHolder`，`ImageViewHolder`，和`AudioViewHolder`。每一类`ViewHolder`中有对应的组件来呈现不同类型的内容。

使用了`RecyclerView`之后，笔记内容的添加与删除就非常方便了，只需对`List<NoteItem>`类对象`noteList`进行`add`和`remove`操作等。

选择本地相册功能

为了实现选择本地相册功能，我们创建了一个`Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);`来指定从相册选择图片。一旦用户选择了图片并且返回时，我们获取图片的`uri`后，添加到`noteList`来更新前端。此时前端展示的图片为本地路径下的图片。同时，前端调用`uploadImage`函数将图片传到后端，实现笔记的同步。

在`uploadImage`函数中，我们首先将图片内容转换为字节数组，创建一个`RequestBody`对象，将字节数组作为请求体，并指定其内容类型为`"image/jpeg"`，表示照片的类型为 JPEG。之后再创建一个`MultipartBody.Builder`对象，并设置其类型为`MultipartBody.FORM`，来使用表单形式提交数据，包括用户的`id`，当前笔记的`path`，和`requestBody`。接着调用`api`接口实现图片的上传。

后端获得图片文件后，会更新图片路径为后端的路径。当前端再次加载页面时，前端显示的图片将为后端路径下的图片，而非本地图片。为了避免图片上传时间过长，笔记页面无法正确加载，当前端得到选择的图片后，后端会先添加默认图片到笔记中，等图片上传成功后再将路径进行替换。这样最大程度地考虑到了用户的体验感和交互感。

相机拍摄功能

与添加本地相册的图片相比，实现相机拍摄要更加复杂，因为需要动态地获取权限。首先检查应用是否具有相机权限，如果有则打开相机，如果没有则请求相机权限。申请权限时，系统会显示权限请求对话框，让用户决定是否授予相机权限。获取到权限之后，调用`openCamera`函数来实现具体的拍摄功能。

`openCamera`函数中，通过创建`Intent(MediaStore.*ACTION_IMAGE_CAPTURE*);`来启动相机进行拍摄，并且根据设备的 Android 版本和相机应用的可用性，选择合适的方式创建照片保存的文件路径或内容 URI。后续操作与选择本地相册类似，包括前端的更新、与上传，后端的创建与替换。

添加录音得到的音频文件

录音功能包括，用户分别点击开始录音与结束录音按钮进行录音，前端播放音频文件、上传文件到后端进行同步。

与相机拍摄类似，录音也需要动态获取权限。具体的录音过程借助了`MediaRecorder`类来实现。而音频文件的上传与图片的上传类似，通过`RequestBody`和`MultipartBody`类提交表单数据。

图片与音频的删除

为了让用户可以便捷删除图片与音频文件，我们设计了长按删除的功能。进行删除时，前端调用`deleteItem`接口更新后端的笔记内容数据，同时前端删除`noteList`中图片和音频对应的`NoteItem`，并且将前后的文字部分进行合并，方便用户的编辑。

分工：王皓雯

AI模型交互

为了让用户可以更效率的撰写笔记，我们在笔记编辑页面添加了AI对话功能，可以实现与AI交互。实现中，我们创建了一个底部窗口来获取用户的问题、显示模型的回答，而其中主要技术点包括两方面：模型使用和状态控制。

在模型使用中，我们使用了Kimi的接口，调用chat接口来获取问题的回答。同时，为了使得AI能够基于笔记作出回答，我们对用户自己的提问脚本进行了简单的修改，具体的内容如下：“以下是笔记的内容：<笔记内容> 基于以上笔记内容，回答问题：<用户的问题>”。另外，当笔记内容为空时，脚本为：“请回答问题：<用户的问题>”

AI部分的另一个难点就是状态的控制，由于AI回答问题所需时间较长，且用户的操作不可预知，我们创建了DIALOG_STATE来记录当前状态，包括：初始、关闭、等待回答和已有回答四个状态。打开、关闭窗口无论何时会强制设置状态为初始和关闭；发送问题时窗口必须打开，且若正在等待回答，用户点击发送会得到请等待的提示，其他情况下成功发送并置状态为等待回答；而当收到回答后，仅在等待回答状态下会对答案进行处理和显示并更新状态，防止对已关闭窗口或重开窗口进行误操作；对话框同时支持复制操作，方便用户处理回答，而复制也仅在已有回答状态下可以操作，以确保用户复制到正确的内容。

分工：白润声