目 录

题目：**基于Android的生活记事本APP**

[1 概述 1](#_Toc484449667)

[1.1 研究背景 1](#_Toc484449668)

[1.2 研究意义 1](#_Toc484449669)

[2 系统设计 2](#_Toc484449671)

[2.1 系统功能模块图 2](#_Toc484449672)

[2.1.1 工程结构图 2](#_Toc484449675)

[2.1.2 功能模块图 3](#_Toc484449675)

[2.2 系统模块功能说明 4](#_Toc484449673)

[2.2.1 登录模块 4](#_Toc484449675)

[2.2.2 日记清单及适配器模块 4](#_Toc484449675)

[2.2.3 发布模块 4](#_Toc484449675)

[2.2.4 用户模块 4](#_Toc484449675)

[2.2.5 编辑模块 4](#_Toc484449675)

[2.2.6 Ui及数据库模块 4](#_Toc484449675)

2.3 数据库设计 4

[2.3.2 数据库表图 4](#_Toc484449675)

[2.3.3 数据库结构概述](#_Toc484449675) 4

[3 系统实现 5](#_Toc484449671)

[3.1 登录模块 5](#_Toc484449672)

[3.1.1 登录功能](#_Toc484449675) 5

[3.1.2 注册功能](#_Toc484449675) 5

[3.1.3 忘记密码功能](#_Toc484449675) 5

[3.1.4 注销账户功能](#_Toc484449675) 5

[3.1.5 流程图](#_Toc484449675) 6

[3.2 日记清单及适配器模块 6](#_Toc484449672)

[3.2.1 功能设计](#_Toc484449675) 6

[3.2.2 流程图](#_Toc484449675) 6

[3.3 发布模块 7](#_Toc484449672)

[3.3.1 功能设计](#_Toc484449675) 7

[3.3.2 流程图](#_Toc484449675) 7

[3.4 用户模块 7](#_Toc484449672)

[3.4.1 功能设计](#_Toc484449675) 7

[3.4.2 流程图](#_Toc484449675) 8

[3.5 Ui和数据库模块 8](#_Toc484449672)

[3.5.1 功能设计](#_Toc484449675) 8

[3.5.2 流程图](#_Toc484449675) 8

[4 系统测试 9](#_Toc484449671)

[4.1 登录模块测试 9](#_Toc484449672)

[4.2 日记清单模块测试 1](#_Toc484449672)1

[4.3 发布模块测试 1](#_Toc484449672)2

[4.4 用户模块测试 1](#_Toc484449672)2

[4.5 编辑模块测试 1](#_Toc484449672)3

[5 设计总结](#_Toc484449671) 13

[5.1 Bug集锦 1](#_Toc484449672)3

[5.1.1 数据库版本更新](#_Toc484449675) 13

[5.1.2 数据库类无法得到Dao](#_Toc484449675) 13

[5.1.3 添加到图片无法删除](#_Toc484449675) 13

[5.2 总结体会 1](#_Toc484449672)4

[5.2.1 对Fragment的使用](#_Toc484449675) 14

[5.2.2 对数据库的使用](#_Toc484449675) 14

[5.2.3 界面Ui的设计](#_Toc484449675) 14

[5.2.4 选择图片添加进EditText组件](#_Toc484449675) 14

**题目：基于Android的生活记事本APP**

1. 概述
   1. 研究背景

生活记事本是一款基于 Android 平台的应用程序，旨在帮助用户记录和组织个人的日常事件、想法和待办事项。它提供了一个便捷的方式，让用户随时随地记录重要的事件、感受和任务，以便将来回顾和参考。

背景研究表明，随着现代生活节奏的加快，人们越来越需要一个简单、易用的工具来记录和管理他们的日常事务。传统的纸质笔记本已经逐渐被数字化工具所取代，因为它们能够提供更多的功能和灵活性。

* 1. 研究意义

基于上述背景，开发基于 Android 的生活记事本应用具有很多优势和潜在的用户需求。一些主要的研究背景因素包括：

1.移动性：Android 平台的普及使得人们的智能手机成为随身携带的重要工具。生活记事本应用可以在任何时间、任何地点记录重要的事项，方便用户及时保存和管理。

2.多媒体支持：Android 平台提供了强大的多媒体功能，例如照片和声音录制，这使得用户可以更加丰富地记录日常事件。生活记事本应用可以通过集成这些功能，让用户以更多元的方式记录和回顾重要时刻。

3.数据同步和备份：随着人们在多个设备上使用应用程序的需求增加，数据同步和备份的重要性也日益凸显。生活记事本应用可以通过与云服务集成，实现数据的自动同步和备份，使用户可以在不同设备之间无缝切换和访问他们的记事。

4.数据隐私和安全：在数字化时代，数据隐私和安全成为用户最关心的问题之一。在设计生活记事本应用时，应考虑用户数据的保护和隐私设置，确保用户感到安全并愿意信任该应用。

通过对用户需求和现代生活方式的深入了解，可以设计出功能完善、用户友好的生活记事本应用程序，以满足用户的日常记录和组织需求。

以上是基于 Android 的生活记事本应用的研究背景。在实际开发过程中，还需考虑市场竞争、用户反馈和不断变化的技术趋势，以确保应用能够成功满足用户的期望和需求。

1. 系统设计
   1. 系统功能模块图
      1. 工程结构图

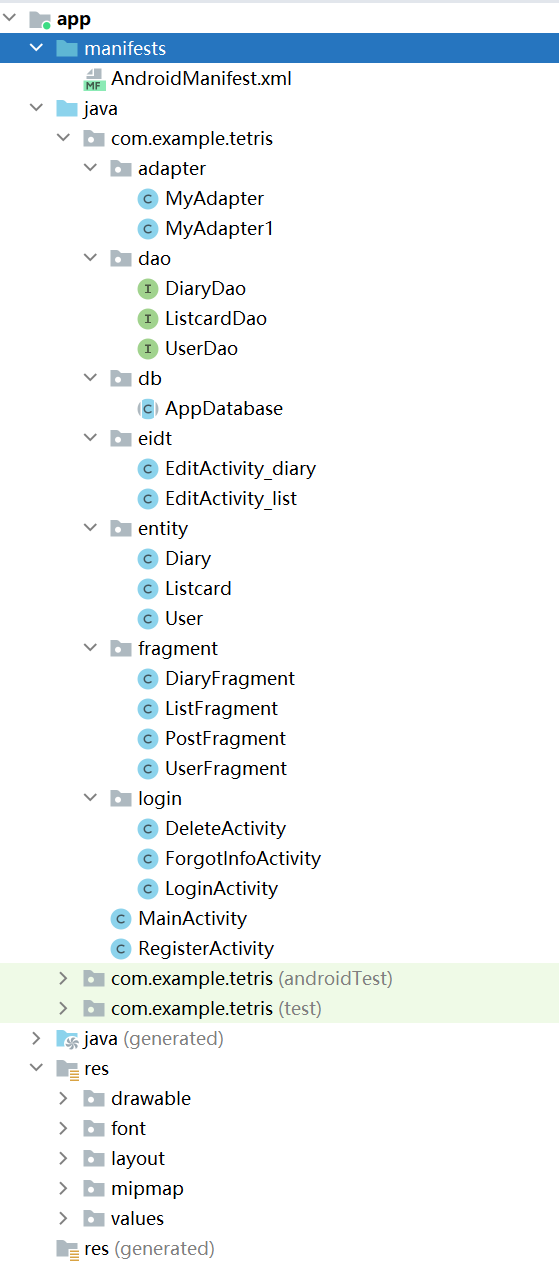


图2.1.1 工程结构

* + 1. 功能模块图

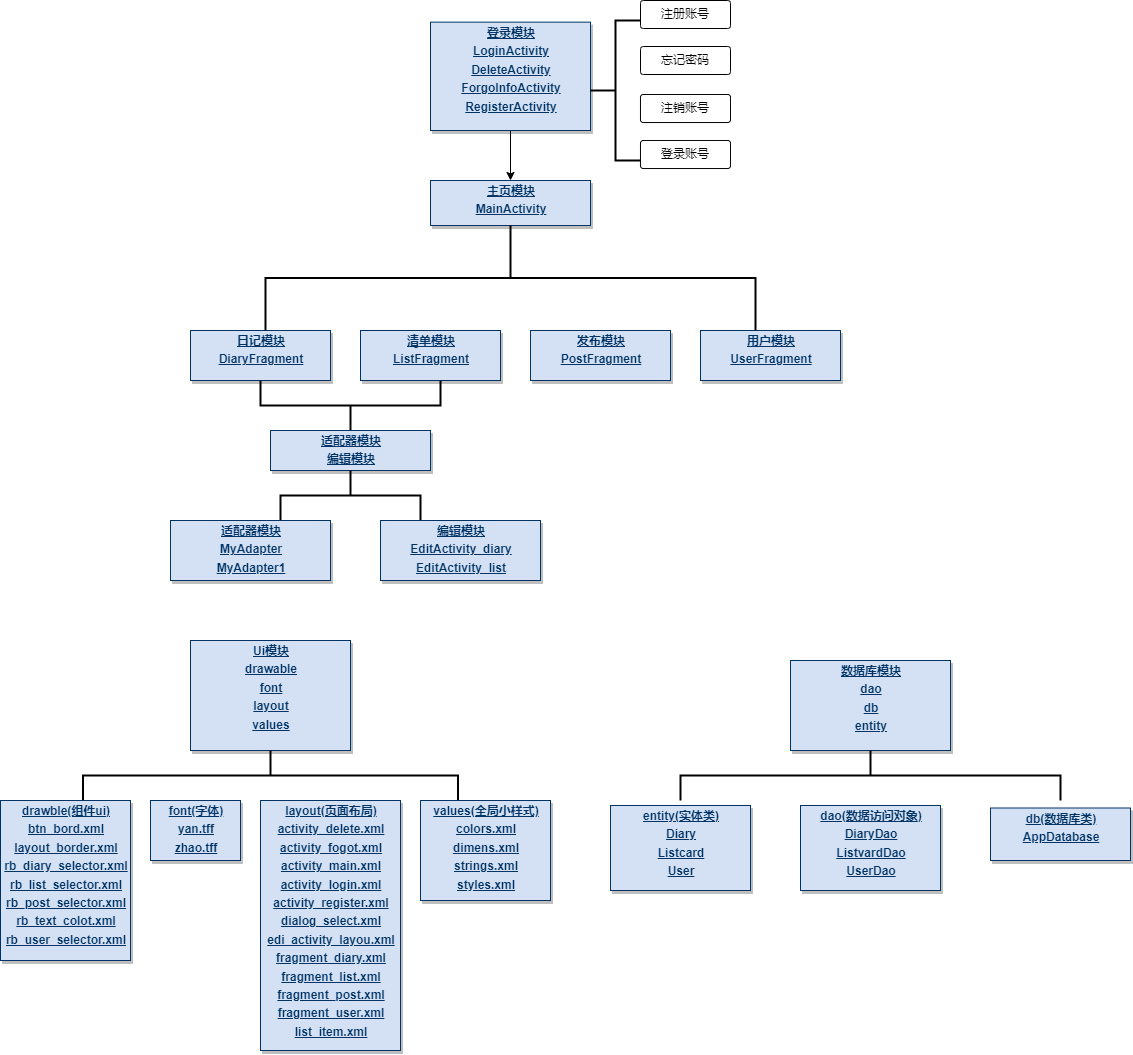


图2.1.2 功能模块图

* 1. 系统模块功能说明
     1. 登录模块

给用户提供登录账户，忘记密码，注销账户及注册账户的功能，与数据库结合，在用户数据比对中提供了以上功能的校验环节，确保对数据库中数据操作不会出错。

* + 1. 日记清单及适配器模块

显示用户保存的日记和清单，包含标题，内容的头部简介，以及删除该条清单的功能。适配器将用户数据库中的日记和清单数据填充到日记和清单模块，以及编辑和删除的事件处理。

* + 1. 用户模块

显示用户的账号和名称，提供登出账号的功能和关于我的的查看。

* + 1. 发布模块

在该模块编辑你的日记或者清单，写好题目和内容后，选择保存为日记或者清单，以及添加图片功能。

* + 1. 编辑模块

提供对用户已经存储在数据库的清单和日记进行修改的功能。

* + 1. Ui和数据库模块

UI给app的所有模块的界面一个相对美观的展示。数据库存储用户的账号密码，账号名字，保存的日记和清单。

* 1. 数据库设计
     1. 数据库表图

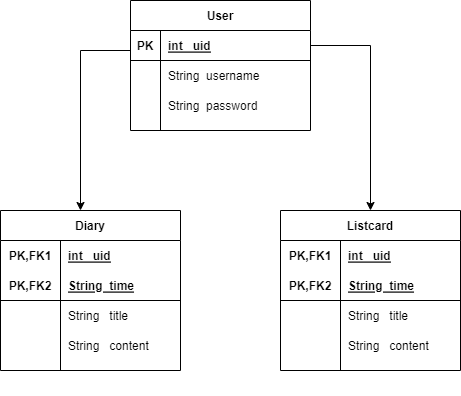


图2.3.1 数据库表图

* + 1. 数据库结构概述

该app采用Room框架来访问和对SQLite数据库完成增删改查等操作，首先设立了三个entity实体类，分别为user，diary和listcard，用来关联数据库中的表，再设立dao包，创建三个实体类的dao接口，用于增删改查等操作，最后为Database类，将关联的实体类全部填入，并声明各个实体类dao的抽象方法。

1. 系统实现
   1. 登录模块

3.1.1 登录功能

a. 用户打开登录页面，输入其已注册的用户名和密码。

b. 系统验证用户输入的用户名和密码是否匹配数据库中存储的数据。

c. 如果匹配成功，系统将用户标识为已登录状态，并跳转到应用主页页面。

d. 如果匹配失败，系统给出相应的错误提示，要求用户重新输入有效的用户名和密码。

3.1.2 注册功能

a. 用户点击注册按钮跳转注册界面，填写要注册的账户的用户名和密码并重复。

b. 系统验证用户提供的信息的有效性，例如检查用户名是否已被占用，密码是否符合要求等。

c. 如果验证通过，系统将用户提供的信息存储到数据库中，并分配一个唯一的用户ID。

d. 注册成功后，用户便可会返回登录界面进行登录。

3.1.3 忘记密码功能

a. 用户点击忘记密码按钮，系统提示用户提供其注册时所填写的账号。

b. 系统从数据库中存储的数据验证用户提供的账户的有效性。

c. 用户输入新的密码并确认。

d. 系统验证新密码的有效性，并将其更新到对应的用户账户中，完成密码重置过程。

3.1.4 注销账户功能

a. 用户点击注销账户按钮进入注销账户界面。

b. 用户填写注销账户信息，账号及密码。

d. 用户确认注销操作后，系统先根据数据库检验账户有效性，有效则执行注销操作，将用户账户从数据库中永久删除账户的相关数据。

3.1.5 流程图

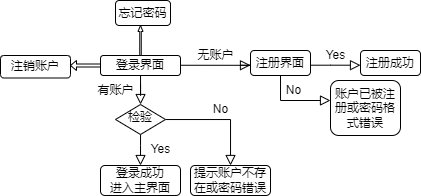


图3.1.5 登录流程图

* 1. 日记清单及适配器编辑模块

3.2.1 功能设计

主要有显示，编辑和删除功能，登录进app后的首要显示的fragment即为用户的日记和清单记录显示，由登录模块登入时通过Intent传入主键Uid，通过uid，系统从数据库中读取该账户的日记和清单数据，利用自定义的适配器将数据填充到ListView组件显示出来，同时自定义的listitem的Ui组件含有编辑和删除功能的按钮，删除会直接从该账户中删除该条日记或清单，编辑则会跳转编辑页面，编辑成功后会将更新后的数据同步到数据库中的该条日记或清单数据中。

3.2.2 流程图

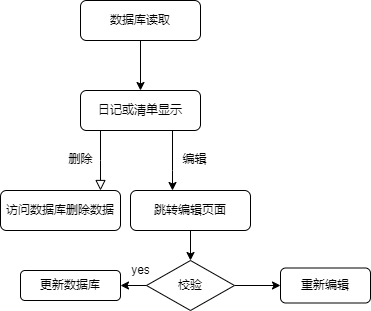


图3.2.2 日记清单流程图

* 1. 用户模块

3.3.1 功能设计

用户在该fragment内可看到自己的账户名称，关于我的及登出账户的功能，账户名称由登录模块登入时通过Intent传入主键Uid，通过uid，系统读出该用户的相关数据并显示，关于我的则设计了一个点击事件，点击后出现弹窗显示“这是我的课设”，登出账户设计的点击事件为结束当前activity，返回登录界面。

3.3.2 流程图



图3.3.2 用户流程图

* 1. 发布模块
     1. 功能设计

用户在该界面编辑自己想要发布的日记或清单，总体界面布局为当前时间显示，标题编辑和内容编辑，标题和内容可以为空，

用户编辑完日记或清单后，点击保存为的按钮，触发点击事件，弹出选择保存为日记还是清单的选择框，选择确定以后，系统将用户所填写的数据保存进数据库中，显示在日记和清单模块上。

添加图片功能：给定一个onActivityResult()方法的实现，用于处理从图库中选择图片后的返回结果。在方法中，首先检查请求码是否与预期的图片选择请求码匹配，并检查结果码是否为成功(RESULT\_OK)。如果条件满足，获取选中图片的URI。然后，将URI转换为Bitmap对象， 接下来，创建一个可编辑的SpannableStringBuilder对象，用于插入图片。在处理图片之前，对选中的图片进行压缩操作。然后，设置所需的图片尺寸，并使用Bitmap.createScaledBitmap()方法将压缩后的Bitmap对象调整为指定的宽度和高度。将调整大小后的Bitmap对象转换为Drawable对象，通过使用BitmapDrawable类，并传入Bitmap对象和资源获取器(getResources()）。

创建ImageSpan对象 在可编辑的SpannableStringBuilder对象中插入ImageSpan，通过使用append()方法将一个空格添加到builder，然后使用setSpan()方法将ImageSpan插入到空格位置。

设置Drawable的边界，通过调用setBounds()方法，传入左坐标、上坐标、右坐标和下坐标。获取EditText的当前光标位置(selectionStart)。在当前光标位置插入图片，通过获取EditText的Editable对象(editable)，然后调用insert()方法，将builder插入到指定位置。

* + 1. 流程图

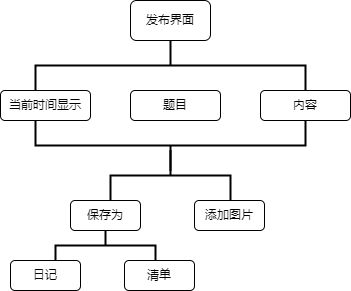


图3.4.2 发布流程图

* 1. Ui和数据库模块
     1. 功能设计

Ui模块给各个界面一个较为美观的可视化界面，主要调整各个组件的位置和样式，对按钮选中和未选中做出两种不同样式，每个主键设定一个唯一的id，使其能在activity中被操作，数据库模块为所有数据提供存储，该app使用room框架的database类来使用建立了一个mydb的SQLite数据库，其中有三张表用于存储即三个entity实体，分别是user表用于存储用户的名称，密码，主键为系统自动分配且自增的uid，diary表和list表用于存储所有用户的日记和清单数据，主键为uid和time，通过用户的uid来提取该用户的日记和清单数据，对数据库的操作全部通过dao层的代码实现。

* + 1. 流程图

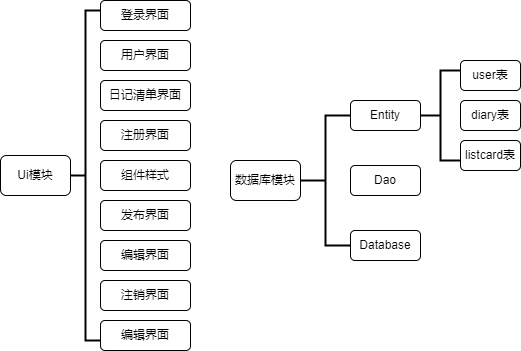


图3.5.2 Ui及数据库流程图

1. 系统测试
   1. 登录模块



图4.1.1 登录账号



图4.1.2 注册账号

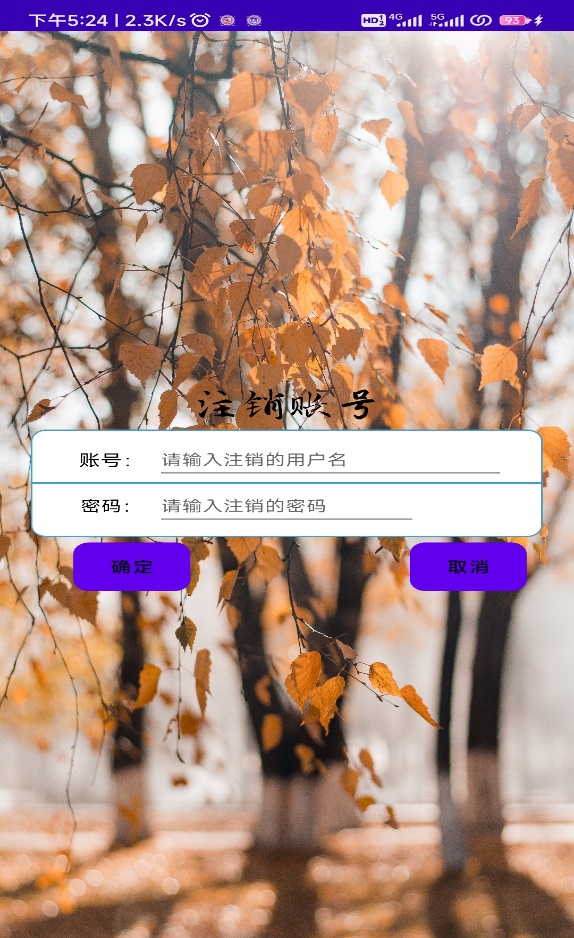


图4.1.3 注销账号



图4.1.4 忘记密码

* 1. 日记清单模块

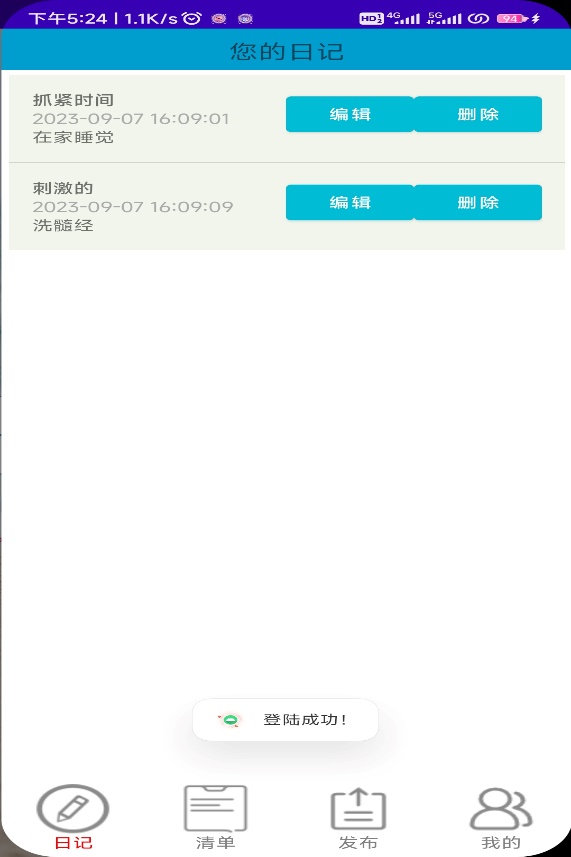


图4.2.1 日记界面



图4.2.2 清单界面

* 1. 发布模块

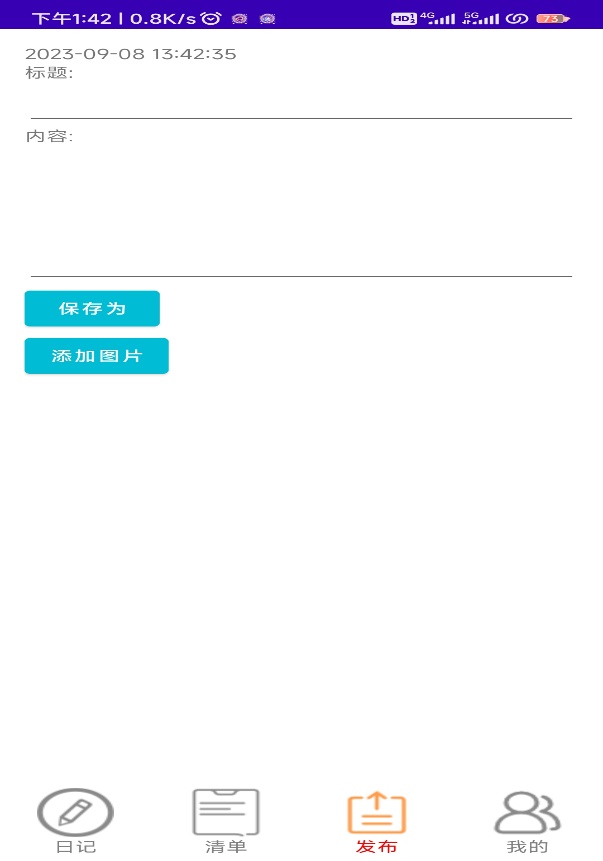


图4.3 发布界面

* 1. 用户模块



图4.4 用户界面

* 1. 编辑模块

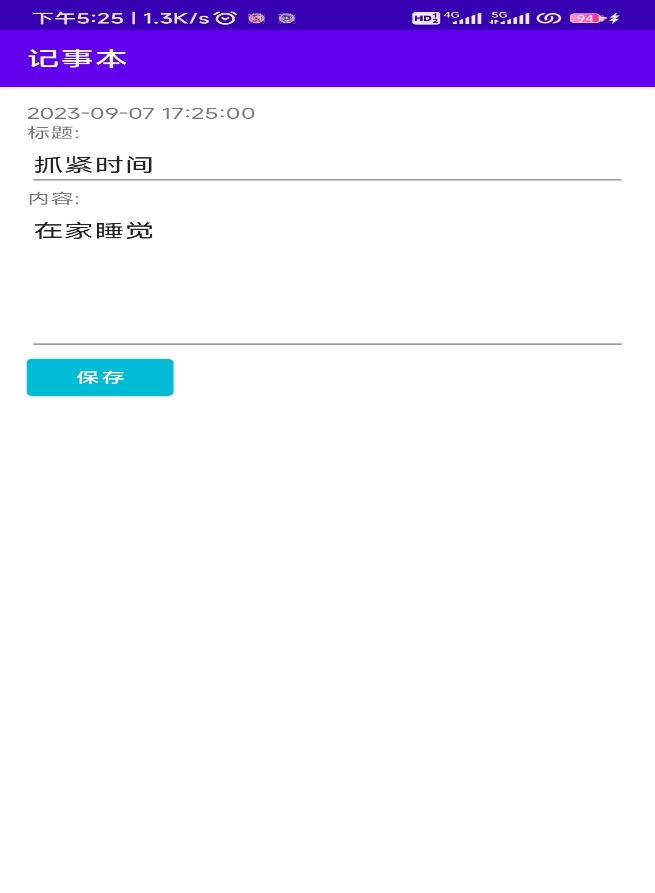


图4.5 编辑界面

1. 设计总结
   1. Bug集锦

5.1.1 数据库版本更新

在使用Room框架时，对entity成员属性设置和dao层的方法未考虑到位，更新属性类型和方法后发现数据库无法访问。

解决方法：更新database数据库的version版本号。

5.1.2 数据库类无法得到dao

一开始只设置了一个dao层，database类代码为一层dao抽象方法的示例代码，添加新的dao后在activity中发现无法通过database的实例获得新的dao。

解决方法：在database类中添加新的dao的抽象方法，系统会根据这个自动实例化dao。

5.1.3 添加的图片无法删除

将图片通过onActivityResult传回来放置在EditText上后发现无法通过删除键删除。

解决方法：设置了一个对edittext的OnKeyListener来监听删除键的点击事件。当检测到删除键被按下时，我们首先获取EditText的文本内容，并使用 getSpans() 方法获取文本中的所有 ImageSpan 对象。然后，我们根据当前光标位置判断光标所处的位置是否在某个 ImageSpan 范围内，如果是，则将其删除，并更新 EditText 的内容。

* 1. 总结体会

5.2.1 对Fragment的使用

使用Fragment可以将应用程序的用户界面模块化，使得界面的组织更加灵活和可扩展。通过将不同的功能模块拆分为独立的Fragment，可以方便地管理复杂的用户界面，并实现模块的复用。Fragment之间可以通过Activity作为中介来进行通信和交互。通过定义接口在Fragment之间传递数据、事件等通信，可以实现不同Fragment之间的信息共享和交互操作。每个fragment的界面可以单独设计，比反复切换activity来的快且节省内存，通过良好的用户界面设计和流畅的交互操作，提高用户的使用体验，并使得App更加易于理解和使用。

5.2.2 对数据库的使用

Room框架下的数据库的操作依靠对Dao层的编写，实体Entity类即为数据库中的表，需要合理设置主码及相关的约束，在对主键的设置需要在类头设置，不能使用多个@Primkey，然后是创建数据库的Database类，在对Entity和Dao层更改后要改变其版本。

5.2.3 界面Ui的设计

想要一个相对美观的Ui，往往需要自定义各个组件的样式，设置不同情况下的各个组件和界面布局的状态，然后细心排版，还要收集合适美观的图标代替系统自带的古板样式，才能获得一个相对整洁美观的Ui。

5.2.4 选择图片添加进EditText组件

给定一个onActivityResult()方法的实现，用于处理从图库中选择图片后的返回结果。在方法中，首先检查请求码是否与预期的图片选择请求码匹配，并检查结果码是否为成功(RESULT\_OK)。如果条件满足，获取选中图片的URI。然后，将URI转换为Bitmap对象， 接下来，创建一个可编辑的SpannableStringBuilder对象，用于插入图片。在处理图片之前，对选中的图片进行压缩操作。然后，设置所需的图片尺寸，并使用Bitmap.createScaledBitmap()方法将压缩后的Bitmap对象调整为指定的宽度和高度。将调整大小后的Bitmap对象转换为Drawable对象，通过使用BitmapDrawable类，并传入Bitmap对象和资源获取器(getResources()）。

创建ImageSpan对象 在可编辑的SpannableStringBuilder对象中插入ImageSpan，通过使用append()方法将一个空格添加到builder，然后使用setSpan()方法将ImageSpan插入到空格位置。

设置Drawable的边界，通过调用setBounds()方法，传入左坐标、上坐标、右坐标和下坐标。获取EditText的当前光标位置(selectionStart)。在当前光标位置插入图片，通过获取EditText的Editable对象(editable)，然后调用insert()方法，将builder插入到指定位置。