

课题编号：  
密 级：

# 笔记&备忘课题 详细设计

(版本号 PD01 V1.00)

笔记&备忘课题组

2016 年 5 月 10 日

## 审批记录

[illegible]

## 变更记录

变更 编号	版本号	日期	章节/段落/行 或图/表号	变更 状态	变更简单描述	审核人	批准人
1	PD01 V1.00	2016.5 .24		A	初稿		
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							

变更状态: A – 增加    M – 修改    D – 删除

## 目录

课题编号: .....	I
密    级: .....	I
笔记&备忘课题组.....	I
1 引言.....	1
1.1 编写目的.....	1
1.2 课题背景.....	1
1.3 术语.....	1
1.4 参考资料.....	2
2 模块总体设计.....	2
2.1 模块汇总表.....	2
2.2 模块关系图.....	3
2.3 模块命名规则.....	5
3 模块详细设计.....	12
3.1 子系统 A 的模块设计.....	12
3.1.1 模块 A1 设计.....	12
3.1.2 模块 A2 设计.....	15
3.1.3 模块 A3 设计.....	16
3.1.4 模块 A4 设计.....	19
3.1.5 模块 A5 设计.....	20
3.1.6 模块 A6 设计.....	22
3.1.7 模块 A7 设计.....	23
3.1.8 模块 A8 设计.....	24
3.2 子系统 B 的模块设计.....	25
3.2.1 模块 B1 设计.....	25
3.2.2 模块 B2 设计.....	27
3.2.3 模块 B3 设计.....	28
3.3 子系统 C 的模块设计.....	28
3.3.1 模块 C1 设计.....	28
3.3.2 模块 C2 设计.....	31
3.3.3 模块 C3 设计.....	31
3.3.4 模块 C4 设计.....	32
3.3.5 模块 C5 设计.....	33
3.4 子系统 D 的模块设计.....	35
3.4.1 模块 D1 设计.....	35
3.4.2 模块 D2 设计.....	37
3.4.3 模块 D3 设计.....	39
3.5 子系统 E 的模块设计.....	39
3.5.1 模块 E1 设计.....	39
3.6 子系统 F 的模块设计.....	40
3.6.1 模块 F1 设计.....	40
3.7 子系统 G 的模块设计.....	41

3.7.1 模块 G1 设计 .....	41
3.8 子系统 H 的模块设计 .....	43
3.8.1 模块 H1 设计 .....	43
3.9 子系统 I 的模块设计 .....	44
3.9.1 模块 I1 设计 .....	44
3.10 子系统 J 的模块设计 .....	45
3.10.1 模块 J1 设计 .....	45
3.11 子系统 K 的模块设计 .....	46
3.11.1 模块 K1 设计 .....	46
4 数据库设计 .....	47
4.1 数据库环境说明 .....	47
4.2 数据库的命名规则 .....	47
4.3 逻辑设计 .....	47
4.3.1 逻辑设计的 E-R 图 .....	47
数据库逻辑结构设计 .....	47
4.4 物理设计 .....	48
4.4.1 数据库表一览表 .....	50
4.4.2 数据库表定义 .....	50
4.5 安全性设计 .....	51
4.5.1 防止用户直接操作数据库的方法 .....	51
4.5.2 用户帐号密码的加密方法 .....	51
4.5.3 角色与权限 .....	51
4.6 数据库管理与维护说明 .....	52
5 界面设计 .....	52
5.1 安卓版界面 .....	52
5.1.1 登录界面 .....	52
5.1.2 主界面 .....	53
5.1.3 笔记编辑界面 .....	54
5.2 ios 版界面 .....	55
5.2.1 登录界面 .....	55
5.2.2 主界面 .....	56
5.2.3 笔记编辑界面 .....	57
5.3 web 版界面 .....	58
5.3.1 登录界面 .....	58
5.3.2 主界面 .....	59
5.3.3 笔记编辑界面 .....	60
5.4 美学设计 .....	60
5.5 界面资源设计 .....	61
5.5.1 图标资源 .....	61
5.5.2 图像资源 .....	74
5.5.3 界面组件 .....	75
6 开发任务分配 .....	76

# 1 引言

## 1.1 编写目的

本阶段为笔记&备忘的详细设计说明书。本阶段完成系统的详细设计并明确系统的数据结构与软件结构。在软件设计阶段主要是把一个软件需求转化为软件表示的过程，这种表示只是描绘出软件的总的概貌。本详细设计说明书的目的就是进一步细化软件设计阶段得出的软件总体概貌，把它加工成在程序细节上非常接近于源程序的软件表示。

本阶段将详细介绍笔记&备忘软件的详细内容，包括模块总体设计、模块详细设计、数据库详细设计、界面详细设计几个部分。完成概要设计的设计细节和要求。

软件开发小组的每一位参与开发成员应该阅读本说明，以清楚产品在技术方面的要求和实现策略，本手册将进行技术评审和技术的可行性检查。

## 1.2 课题背景

课题名称：笔记&备忘软件

委托单位：课程要求

开发单位：项目组成员

开发方式：windows ide、Mac OS xCode、Linux 平台下开发

背景：为了能高效地工作、学习，市场上出现了各种笔记备忘类软件，这些软件替代了传统的纸质笔记本，以其方便、便携、共享性受到用户青睐。一些常用系统，比如 iOS、MIUI、Mac OS 也将笔记和自己的日历、短信功能结合起来，大大提升了使用体验和工作效率。市场上现有不少笔记备忘软件：Evernote、Wunderlist、MIUI 的笔记和便签、iOS 的备忘录提醒事项，这些软件的功能、体验很丰富，同时笔记类应用正变的越来越“重”：功能上，越来越复杂、交互上越来越难，用户体验则显得过于繁琐。

目标：本软件致力于最精致地实现笔记和备忘的功能，专注于专一、纯粹。

## 1.3 术语

缩写、术语	解 释

## 1.4 参考资料

[1]郑人杰,殷人昆,陶永雷。《实用软件工程》(第二版)。1997,北京:清华大学出版社。

[2]王立福,麻志毅。《软件工程》(第二版)。2001,北京:北京大学出版社。

[3]Sharl Lawrence Pfleeger,Joanne M. Atlee.Software Engineering:Theory and Practice. Fourth Edition.

## 2 模块总体设计

### 2.1 模块汇总表

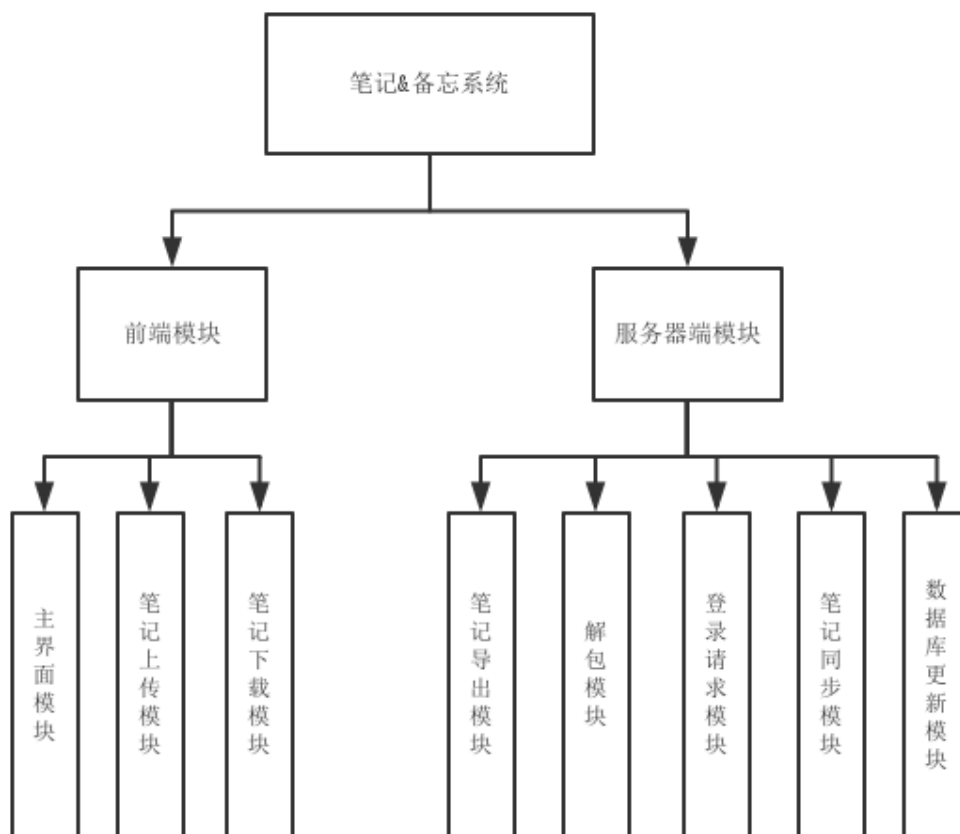
子系统 A 主界面	
模块名称	功能简述
主界面模块	完成客户端主界面的布局,实现和子功能结构的接口调用。
编辑笔记模块	实现客户端笔记编辑的界面布局,实现基本的文字输入功能,实现和子功能结构的接口,记录笔记创建和更新时间。
删除模块	完成删除笔记或备忘的功能
查询模块	实现查询关键字分词,识别功能。
备忘属性模块	设置对笔记添加备忘功能。
已完成界面模块	对已完成笔记的界面布局和逻辑实现。
登录模块	实现用户注册、登录
导出模块	触发导出操作。
子系统 B 备忘属性	
模块名称	功能简述
备忘属性模块	设置对笔记添加备忘功能。
提醒时间模块	设置对笔记提醒的时间。
计时模块	后台对备忘计时并提醒。
子系统 C 编辑笔记	
模块名称	功能简述
编辑笔记模块	实现客户端笔记编辑的界面布局,实现基本的文字输入功能,实现和子功能结构的接口,记录笔记创建和更新时间。
添加图片模块	实现从系统添加图片到笔记的功能。
添加画图模块	完成画图界面的逻辑实现和保存图片功能。
添加标签模块	实现本地全局标签的更新、同步和自动分类功能。
本地笔记保存模块	实现对本地笔记的保存。
子系统 D 查询	

模块名称	功能简述
查询模块	实现查询关键字分词，识别功能。
分类模块	实现查询，并对查询结果进行分类功能。
查询展示模块	实现对查询结果的显示界面的布局。
子系统 E 笔记上传	
模块名称	功能简述
笔记上传模块	实现对笔记的上传。
子系统 F 笔记下载	
模块名称	功能简述
笔记下载模块	实现对笔记的下载同步。
子系统 G 登录请求	
模块名称	功能简述
登录请求模块	处理用户注册、登录。并响应。
子系统 H 笔记同步	
模块名称	功能简述
笔记同步模块	处理用户多平台同步、笔记更新同步逻辑实现。
子系统 I 笔记导出	
模块名称	功能简述
笔记导出模块	处理导出请求
子系统 J 数据库更新	
模块名称	功能简述
数据库更新模块	更新数据库。
子系统 K 解包	
模块名称	功能简述
解包模块	处理客户端发送的 socket 包

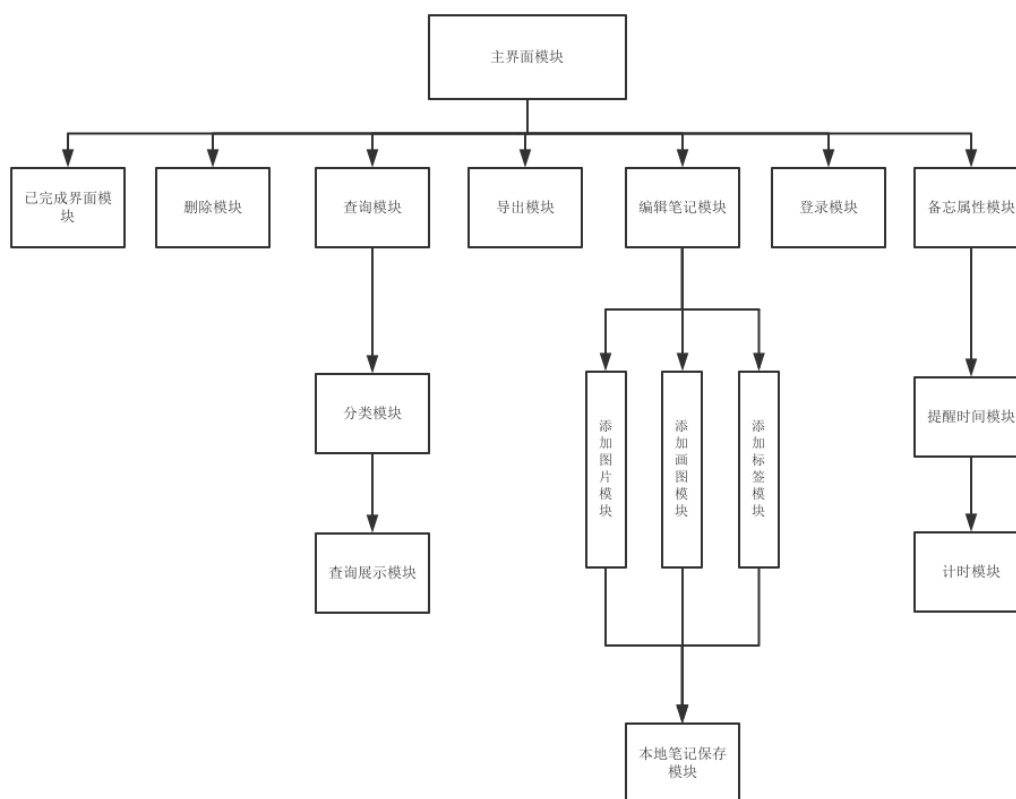
## 2.2 模块关系图

模块总体关系：





模块细节关系：



## 2.3 模块命名规则

本软件以 Java 语言编写 Android 方向的 App，以 Swift 语言编写 iOS 方向的 App，以 Python 语言编写 Web 方向的工作，本编码规范以 Java 为例，其余两种语言规范除了语言特性部分外，规范参照本 Java 规范。

### 一般规范

#### 规则 1

所有包，类，接口，方法，属性，变量，参数均使用英文单词进行命名，具体细节请参见命名规范一章

#### 规则 2

修改源代码时，应尽量保持与所修改系统的编码风格保持一致。

#### 规则 3

所有包名使用必须使用 `com.[company]` 前缀，所有项目使用 `com.[company].projects.[project name]`, `company` 是公司简称，`project name` 是项目的缩写。

### 格式规范

#### 规则 4

包的导入应该按照相关性进行分组

#### 规则 5

类和接口中元素的布局顺序。

1. 类和接口的文档描述
2. 类和接口的声明
3. 类的静态变量，按照 `public`，`protected`，`package`，`private` 的顺序。
4. 实例变量，按照 `public`，`protected`，`package`，`private` 的顺序。
5. 类的方法，无固定顺序。

#### 规则 6

类的声明，基类和实现的接口应该独立成行，保证可读性。

#### 规则 7

变量声明，采用 Camel 表示法不要在一行声明多个变量。

//推荐

```
int level;  
int size;
```

//避免

```
int level, size;
```

#### 规则 8

代码缩进，应该使用 4 个空格为一个单位进行缩进。

#### 规则 9

条件语句的主要形式，即使单条语句，也要使用括号括起来。

#### 规则 10

空格的使用

1. 运算符两边应该各有一个空格。
2. Java 保留字后面应跟随一个空格。
3. 逗号后面应跟随一个空格。
4. 冒号后面应各有一个空格。
5. 分号后面应跟随一个空格。

#### 规则 11

空行的使用

1. 文件头部注释、`package` 语句和 `import` 语句之间。
2. `class` 之间
3. 方法之间
4. 方法中，变量的申明和具体代码之间。
5. 逻辑上相关的语句段之间
6. 块注释和行注释的前面

### 命名规范

#### 规则 12

包名应该用小写字母，不要出现下划线等符号，名词用有意义的缩写或者英文单词。

规则 13

所有类命名使用 **Pascal** 表示方式，使用名词组合。

规则 14

接口命名使用字母 “I” 加上 **Pascal** 形式的表示方式。

规则 15

使用名词组合或形容词去命名一个接口，接口声明了一个对象能提供的服务，也描述了一个对象的能力。一般以 “able” 和 “ible” 作为后缀，代表了一种能力。

规则 16

变量名和参数名使用 **Camel** 表示方式。

规则 17

对于常量名，使用大写字母，并使用下划线做间隔。

**MAX\_TIMES, DEFAULT\_NAME**

程序中应该使用常量代替 “25”， “100” 等实际的数字。

规则 18

方法名应该使用动词开头，使用 **Camel** 表示方式,一般由动词+名词组成。

规则 19

缩写字母也应该保持首字母大写

规则 20

变量的名字应该和类型名称一致

规则 20

根据变量的作用范围，作用范围大的应该使用长名称，作用范围大，表明变量的生命周期比较长，为了有助于理解，应尽量用长名称以表达变量的真实意图。反之，对于作用范围小，可以使用一些简化的名称，比如 i, j, k 等，提高编程效率。

规则 21

使用 **get/set** 对类属性进行访问，这是 **Java** 社区的核心编码规范。

规则 22

使用 **is** 前缀表示一个布尔变量和方法。

规则 23

在查询方法中应使用 *find* 作为前缀

规则 24

使用 **initialize** 做为对象初始化的方法前缀

规则 25

对于对象集合，变量名称应使用复数。

## 规则 26

对于抽象类，应该使用 `Abstract` 前缀。

## 规则 27

对于表示编号的变量，应加 `No` 后缀。

## 规则 28

常在一起使用的对称词汇，这些词汇一起使用，方法的表达意图自然可以互相推测和演绎。 `get/set`, `add/remove`, `create/destroy`, `start/stop`, `insert/delete`, `increment/decrement`, `begin/end`, `first/last`, `up/down`, `min/max`, `next/previous`, `old/new`, `open/close`, `show/hide`, `suspend/resume` etc

## 规则 29

禁止使用否定布尔变量

## 规则 30

异常类应该使用 `Exception` 做为后缀。

## 规则 31

缺省接口实现应该使用 `Default` 前缀

## 规则 32

对于单例类（`Singleton`），应该使用 `getInstance` 方法得到单例。

## 规则 33

对于工厂类，进行创建对象的方法，应该使用 `new` 前缀

## 注释规范

## 规则 34

注释尽量简洁，尺度没有准确的定义，大部分人能明白即可，可以将自己的代码给同事看看。太简单的方法就不要注释。

## 规则 35

注释内容

- ① 代码的版权信息。
- ② 类描述信息，描述类的主要职责和用处。
- ③ 方法描述信息，描述方法是做什么的，如何调用，最好给出调用代码示例。
- ④ `JavaDoc` tags ,用来生成 `Html` 形式的 `API` 文档
- ⑤ 内部实现注释，用于描述复杂的算法，长方法，从为什么要这么做角度去 描述

/\*

\* Copyright (c) 2002-2006 by OpenSymphony

\* All rights reserved.

```

*/
package com.opensymphony.xwork2;
import com.opensymphony.xwork2.interceptor.PreResultListener;
import com.opensymphony.xwork2.util.ValueStack;
import java.io.Serializable;
/**
 * 类职责简要描述
 *   ...
 *
 * @author Jason Carreira
 * @see com.opensymphony.xwork2.ActionProxy
 */
public class ActionInvocation implements Serializable {
/**
 * 方法简要描述
 *
 * 方法详细描述
 *   ...
 *
 * JavaDoc tags,比如
 * @author
 * @version
 * @see ...
 * @return a Result instance
 */
Result getResult() throws Exception{
//内部实现注释
//多行内部实现注释
String name = this.getName();
}
/**
 * Get the Action associated with this ActionInvocation
 */
Object getAction();
/**
 * @return whether this ActionInvocation has executed before.
 * executed.
 */

```

```

boolean isExecuted();
/**
 * Invokes the next step in processing this ActionInvocation.
 * one. If Interceptors choose not to
 * they will call invoke() to allow the next Interceptor to execute
 * the Action is executed. If the ActionProxy getExecuteResult
 */
String invoke() throws Exception;
}

```

#### 规则 36

尽可能在类描述中加入代码调用示例，使用<pre></pre>标记，提示  
 JavaDoc 工具不要改变格式。

#### 规则 37

使用行末注释对深层嵌套代码进行注释

### 实践规范

#### 规则 38

对于静态方法，应该使用类名去使用，不应该用实例去引用，主要是为了体现更多的语义。

#### 规则 39

对一些基本数据类型和不太可能通过继承进行扩展的类，应声明为  
**final**，提高效率。

#### 规则 40

类和方法的粒度保持适中，保持类的规模尽量短小，职责单一。小类有很多好处，易于设计，易于测试，易于理解。同样方法也要尽量的小，每个方法尽量不要超出 25 行。

#### 规则 41

里氏代换原则：假设有两个类，一个是基类 **Base**，一个是派生类 **Derived**，如果一个方法可以接受基类对象 **b** 的话：`method1(Base b)`，同样，这个方法也应该接受派生类 **Derived** 的对象 **d**，而不影响方法的行 为。里氏代换原则是继承复用的基石

#### 规则 42

抽象依赖原则（稳定依赖原则）。应该依赖于抽象而不依赖与具体类，抽象的类和接口是稳定的，而具体类是易变的，如果依赖于具体类，代码就会非常脆弱，失去了灵活性。

#### 规则 43

接口隔离原则，一个类对另外一个类的依赖应该建立在最小的接口之上的。

#### 规则 44

单一职责原则，如果一个类有多于一种的职责，当需求变化时，类的职责就要发生变化，而因此就会引起引用该类的代码发生改变，职责越多，这个类就容易跟更多的类产生耦合关系，而且改变一个职责，可能会影响到另外一个职责的履行。

## 设计模式

### 规则 45

#### 简单工厂

```
abstract class Fruit{ }
class Apple extends Fruit{ }
class Orange extends Fruit{ }
class FruitFactory{
public static Fruit getFruit(String fruitType){
if ("apple" == fruitType){
return new Apple(); } else if ("orange" == fruitType) {
return new Orange(); } } }
Client: Apple apple = FruitFactory.getFruit( "apple" ); ...
```

### 规则 46

#### 工厂方法

```
interface IFruitFactory{
public Fruit getFruit(); }
class AppleFactory implements IFruitFactory{
public Fruit getFruit(){
//生产苹果
return new Apple(); } }
Client: IFruitFactory factory = new AppleFactory(); Fruit fruit = new
factory.getFruit();
```

### 规则 47

#### 单例模式

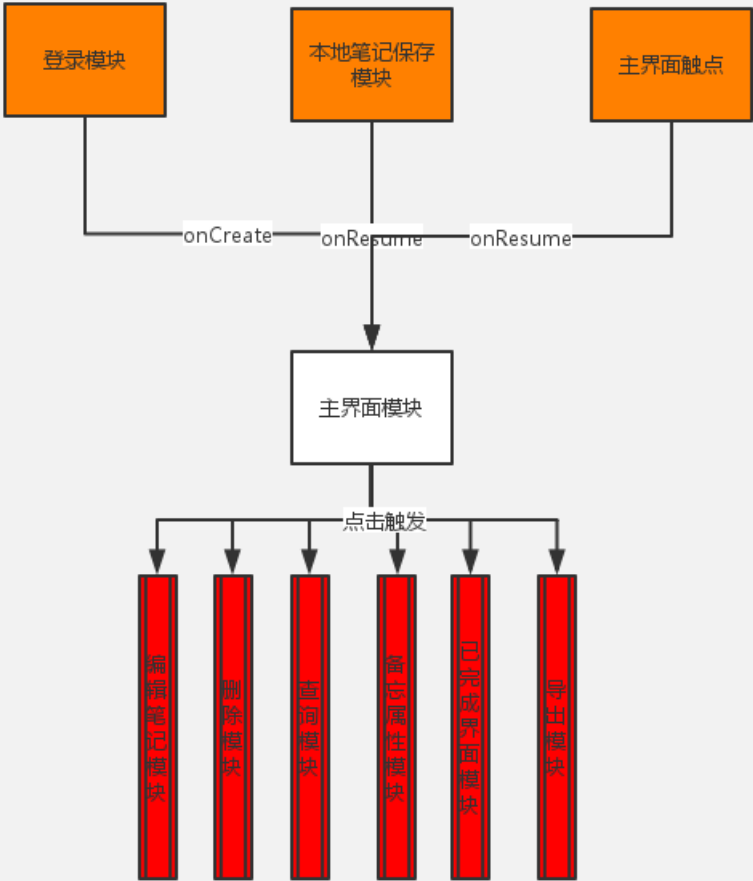
```
class Singleton{
private static Singleton singleton = null;
public static Singleton getInstance(){
if(null == singleton){ singleton = new Singleton(); }
return singleton; }
public String otherOperation(){
//方法实现 } }
Client: String str = Singleton.getInstance().otherOperation();
```



## 3 模块详细设计

### 3.1 子系统 A 的模块设计

#### 3.1.1 模块 A1 设计

模块名称	主界面模块
功能描述	完成客户端主界面的布局，实现和子功能结构的接口调用。
性能描述	对于平铺于屏幕的所有触点都有相应的动作定义，对于不支持的动作定义无输出反馈，不会因为用户的自由操作或者过度操作导致系统触发异常而崩溃。
程序逻辑	<div><pre>graph TD; Login[登录模块] -- onCreate --&gt; Main[主界面模块]; Local[本地笔记保存模块] -- onResume --&gt; Main; Touchpoint[主界面触点] -- onResume --&gt; Main; Main -- 点击触发 --&gt; Edit[编辑笔记模块]; Main -- 点击触发 --&gt; Delete[删除模块]; Main -- 点击触发 --&gt; Query[查询模块]; Main -- 点击触发 --&gt; Reminder[备忘属性模块]; Main -- 点击触发 --&gt; Done[已完成界面模块]; Main -- 点击触发 --&gt; Export[导出模块];</pre></div> <p>主界面的进入由 onCreate 方法和 onResume 方法调用触发，分别是由登录模块、本地笔记保存模块、主界面触点触发。</p> <p>主界面的输出是调用其它模块，编辑笔记模块、删除模块、查询模块、备忘属性模块、已完成界面模块、导出模块。</p>
接口	被调用接口： 1、界面由登录界面的 onClick 回调函数触发，触发的输入为服务器传

来的登录允许许可。

```
Class message{
String userid;          // 用户的 id
Int state;              // 登录的状态，成功或失败
String information;     // 成功或失败的信息字符串描述
}
```

2、由本地笔记保存模块退出触发的主界面 onResume 方法进入主界面模块，不需要传入参数。

3、由点击本地笔记操作的 onClick 方法触发主界面 onResume 方法进入主界面模块，不需要传入参数。

调用接口：

1、编辑笔记

由 onClick 方法调用编辑笔记的界面，参数创建时间确定的唯一的笔记 ID；

String Date;

2、删除

由 onClick 方法调用进入删除模块，传入的参数为笔记的 ID；

String ID;

3、查询

由 onClick 回调查询模块，传入的参数为查询关键字。

String keyword;

4、备忘属性

由 onClick 方法调用进入添加备忘属性模块，传入的参数为笔记的 ID；

String ID;

5、已完成界面

由 onClick 回调一个 Android Fragment，不需要传入参数。

6、导出

由 onClick 方法调用一个后台方法，触发导出请求，不需要传入参数。

局部数据结构：

```
Class mainModule{
Private: noteName [] notes;
        Int notesNum;

        String userid;
        Remind [] remind;
}
Class Remind{
        noteName note;
        String time;
        Bool isFinished;
```

	<pre> } </pre> <p>主界面的主功能结构，保存笔记的数目、笔记拥有者、笔记的数组结构和备忘的数组结构。</p> <p>备忘是由笔记、提醒时间组成。</p>
数据结构与算法	<p>采用经典的数据结构和无特别算法的逻辑处理。</p> <p>为主界面创建本地数据 <code>mainModule</code></p> <p>为每个输出接口创建 <code>onClick</code> 调用方法。</p>
输入	<ol style="list-style-type: none"> <li>1、界面由登录界面的 <code>onClick</code> 回调函数触发，触发的输入为服务器传来的登录允许许可。</li> </ol> <pre> Class message{ String userid;           // 用户的 id Int state;               // 登录的状态，成功或失败 String information;      // 成功或失败的信息字符串描述 } </pre> <p>来源：登录模块</p> <ol style="list-style-type: none"> <li>2、由本地笔记保存模块退出触发的主界面 <code>onResume</code> 方法进入主界面模块，不需要传入参数。</li> </ol> <p>来源：本地笔记保存模块</p> <ol style="list-style-type: none"> <li>3、由点击本地笔记操作的 <code>onClick</code> 方法触发主界面 <code>onResume</code> 方法进入主界面模块，不需要传入参数。</li> </ol> <p>来源：点击本地笔记的操作</p>
输出	<ol style="list-style-type: none"> <li>1、编辑笔记 由 <code>onClick</code> 方法调用编辑笔记的界面，参数创建时间确定的唯一的笔记 ID； <code>String Date;</code></li> <li>2、删除 由 <code>onClick</code> 方法调用进入删除模块，传入的参数为笔记的 ID； <code>String ID;</code></li> <li>3、查询 由 <code>onClick</code> 回调查询模块，传入的参数为查询关键字。 <code>String keyword;</code></li> <li>4、备忘属性 由 <code>onClick</code> 方法调用进入添加备忘属性模块，传入的参数为笔记的 ID； <code>String ID;</code></li> <li>5、已完成界面 由 <code>onClick</code> 回调一个 <code>Android Fragment</code>，不需要传入参数。</li> <li>6、导出 由 <code>onClick</code> 方法调用一个后台方法，触发导出请求，不需要传入参数。</li> </ol>
存储分配	本地存储为 <code>Class mainModule</code> 分配空间，用于保存笔记的核心统计数据。

测试要点	测试所有触点、触发方式的动作有效性，不会因为动作未定义或者数值越界导致异常而崩溃。
限制条件	无
补充说明	主界面模块为笔记&备忘软件的核心模块、主模块，必须保证所有触点的动作定义的有效性。

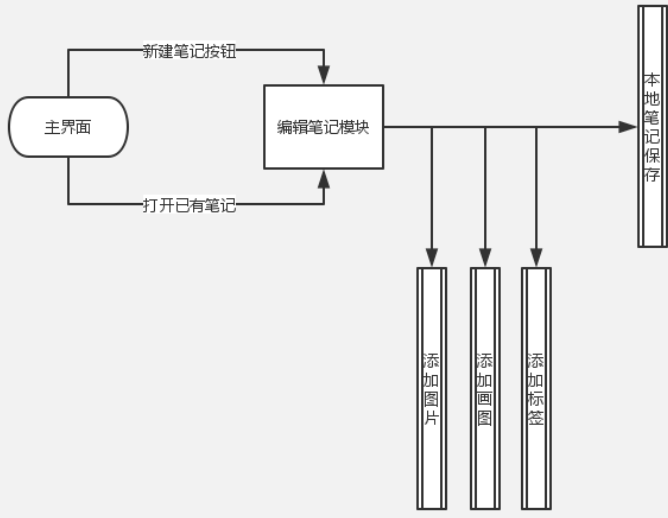
### 3.1.2 模块 A2 设计

模块名称	登录模块
功能描述	实现用户注册、登录
性能描述	用户输入用户名是自动检测是否为邮箱，密码提醒安全程度，对输入数据进行防拖库处理，严格按照邮箱、密码检查输入精度。并且如果用户在非恶意的情况下 输入了错误的数据类型参数，系统将自动提示用户再次输入正确的参数。
程序逻辑	<pre> graph TD     Start([登录页面]) --&gt; D1{是否去注册}     D1 -- 是 --&gt; I1[输入用户邮箱、密码]     I1 --&gt; D2{验证输入}     D2 -- 失败 --&gt; I1     D2 -- 成功 --&gt; I2[发送验证邮件]     I2 --&gt; D3{服务器验证}     D3 -- 失败 --&gt; I2     D3 -- 成功 --&gt; E1([登录成功])     D1 -- 否 --&gt; I3[输入用户邮箱及密码]     I3 --&gt; D4{验证输入}     D4 -- 失败 --&gt; I3     D4 -- 成功 --&gt; D5{服务器验证}     D5 -- 失败 --&gt; D4     D5 -- 成功 --&gt; E2([登录成功])   </pre> <p>App 启动进入此界面，对于注册和登录采取不同的服务器交互处理</p>
接口	<p>被调用接口：</p> <p>程序运行时调用 onCreate 方法</p> <p>无参数</p> <p>调用接口：</p> <p>调用接口向服务器传递消息，并将登录请求或者注册请求发送给服务器。</p> <pre> struct loginCommand{     commandID id;     String userid;     String password; }  enum commandID{     login=1, backup=2, update=3; }   </pre> <p>解析结果为登录请求，请求字段为用户邮箱和用密钥加密的密码。请求 ID 分别为登录请求、同步请求、导出请求。</p> <p>局部数据结构：</p>

	<pre> Class login{ String userid; Stirng password; } </pre>
数据结构与算法	数据结构采取经典的数据结构，算法采用非特殊的判定算法。
输入	无
输出	<pre> struct loginCommand{     commandID id;     String userid;     String password; } enum commandID{     login=1, backup=2, update=3; } </pre> <p>解析结果为登录请求，请求字段为用户邮箱和用密钥加密的密码。请求 ID 分别为登录请求、同步请求、导出请求。</p>
存储分配	<pre> Class login{ String userid; Stirng password; } </pre> <p>存储本模块输入的用户名和密码</p>
测试要点	<p>对于恶意的非正常的输入有一定的抵抗和识别能力。</p> <p>对于与服务器消息传递的时延、成功率、有效应答做出保证。</p>
限制条件	无
补充说明	无

### 3.1.3 模块 A3 设计

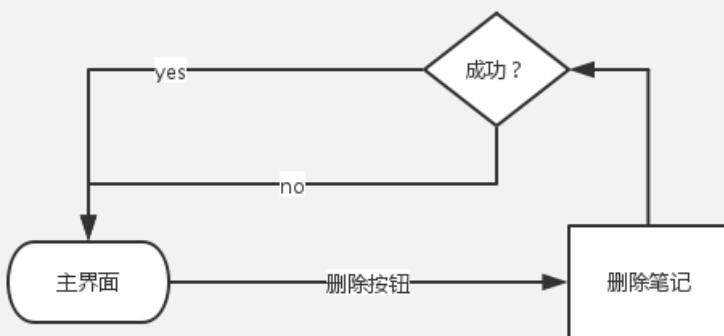
模块名称	编辑笔记模块
功能描述	实现客户端笔记编辑的界面布局，实现基本的文字输入功能，实现和子功能结构的接口，记录笔记创建和更新时间。
性能描述	<p>日期存储严格按照 Mon-Day-Year 的形式和精度。</p> <p>对于平铺于屏幕的所有触点都有相应的动作定义，对于不支持的动作定义无输出反馈，不会因为用户的自由操作或者过度操作导致系统触发异常而崩溃。</p>

程序逻辑	 <p>由编辑笔记触点和打开笔记触点触发进入编辑界面，在编辑界面由触点触发其他接口调用其它模块。</p>
接口	<p>被调用接口：</p> <ol style="list-style-type: none"> <li>1、主界面的编辑笔记触点 由主界面编辑笔记按钮触发新的 Activity，传入参数是笔记唯一标识的 ID，由主界面传入。 String ID;</li> <li>2、主界面的打开笔记触点 由主界面某一笔记点击触发打开笔记的 Activity，传入参数是笔记唯一标识的 ID，由主界面传入。 String ID;</li> </ol> <p>调用接口：</p> <ol style="list-style-type: none"> <li>1、添加图片 由 onClick 方法触发调用系统相册接口，传入参数为笔记 ID，调用模块结束后记录图片 ID 的图片备份。 String ID;</li> <li>2、添加画图 由 onClick 方法触发新的 Activity，传入参数为笔记 ID，调用模块结束后记录图片 ID 和图片备份。 String ID;</li> <li>3、添加标签 由在特定位置输入触发添加标签模块。输入为笔记 ID; String ID;</li> <li>4、本地笔记保存 由按下保存笔记触发，返回主界面。输入参数为笔记 ID; String ID;</li> </ol> <p>局部数据结构：</p> <pre> Class note{ String ID; </pre>

	<pre>String tags[]; String picture[]; String context[]; String editTime; }</pre> <p>保存笔记的唯一标识 ID、标签列表、图片列表、正文、编辑时间。</p>
数据结构与算法	数据结构采用经典的数据结构，算法采用非特殊的条件判别算法。
输入	<p>1、主界面的编辑笔记触点 由主界面编辑笔记按钮触发新的 Activity，传入参数是笔记唯一标识的 ID，由主界面传入。 String ID;</p> <p>2、主界面的打开笔记触点 由主界面某一笔记点击触发打开笔记的 Activity，传入参数是笔记唯一标识的 ID，由主界面传入。 String ID;</p>
输出	<p>1、添加图片 由 onClick 方法触发调用系统相册接口，传入参数为笔记 ID，调用模块结束后记录图片 ID 的图片备份。 String ID;</p> <p>2、添加画图 由 onClick 方法触发新的 Activity，传入参数为笔记 ID，调用模块结束后记录图片 ID 和图片备份。 String ID;</p> <p>3、添加标签 由在特定位置输入触发添加标签模块。输入为笔记 ID; String ID;</p> <p>4、本地笔记保存 由按下保存笔记触发，返回主界面。输入参数为笔记 ID; String ID;</p>
存储分配	<pre>Class note{ String ID; String tags[]; String picture[]; String context[]; String editTime; }</pre> <p>保存笔记的唯一标识 ID、标签列表、图片列表、正文、编辑时间。</p>
测试要点	<p>保证存储空间的充足。</p> <p>测试所有触点、触发方式的动作有效性，不会因为动作未定义或者数值越</p>

	界导致异常而崩溃。
限制条件	无
补充说明	无

### 3.1.4 模块 A4 设计

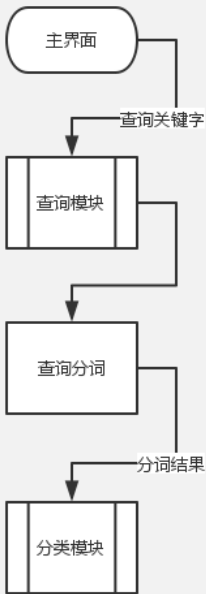
模块名称	删除模块
功能描述	完成删除笔记或备忘的功能
性能描述	用户在执行此模块时会严格执行删除操作，不会误判区域。对于删除失败的情况也有对应的处理。
程序逻辑	 <pre> graph LR     A([主界面]) -- 删除按钮 --&gt; B[删除笔记]     B --&gt; C{成功?}     C -- yes --&gt; A     C -- no --&gt; A </pre> <p>接收主界面的调用，完成删除功能。</p>
接口	<p>被调用接口：</p> <p>主界面删除按钮，接受的参数为笔记的 ID</p> <p>String ID;</p> <p>调用接口：</p> <p>笔记删除成功与否的信息</p> <pre> Class message{ String ID; Bool flag; String message; } </pre> <p>局部数据结构：</p> <p>笔记的 ID</p>
数据结构与算法	<p>笔记存储采用 trie 树的数据结构，采用 trie 树的搜索算法。</p> <p>数据结构：（就是字典树的结构，以下图为例）</p>



	<pre> graph TD     Root(( )) -- t --&gt; t((t))     Root -- A --&gt; A((A))     Root -- i --&gt; i((i))     t -- o --&gt; to((to))     t -- e --&gt; te((te))     to -- 7 --&gt; 7[7]     te -- a --&gt; tea((tea))     te -- d --&gt; ted((ted))     te -- n --&gt; ten((ten))     tea -- 3 --&gt; 3[3]     ted -- 4 --&gt; 4[4]     ten -- 12 --&gt; 12[12]     A -- 15 --&gt; 15[15]     i -- n --&gt; in((in))     in -- 5 --&gt; 5[5]     in -- n --&gt; inn((inn))     inn -- 9 --&gt; 9[9] </pre> <p>搜索算法:</p> <pre> def find(node, key):     for char in key:         if char in node.children:             node = node.children[char]         else:             return None </pre>
输入	String ID; 笔记的 ID
输出	<pre> Class message{     String ID;     Bool flag;     String message; } </pre> <p>记录删除笔记的 ID，并返回删除成功与否的信息。</p>
存储分配	无
测试要点	测试操作的鲁棒性。确定提示删除信息时确实是删除了笔记，提示删除失败时确实是未删除笔记。
限制条件	无
补充说明	无

### 3.1.5 模块 A5 设计

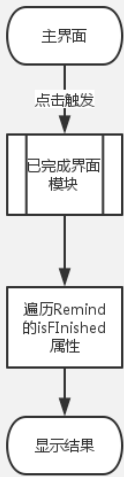
模块名称	查询模块
功能描述	实现查询关键字分词，识别功能。
性能描述	查询输入要求如果是日期的话严格按照日期格式的要求。Month-Day-Year。否则难以识别。

程序逻辑	 <pre> graph TD     A([主界面]) -- 查询关键字 --&gt; B[查询模块]     B --&gt; C[查询分词]     C -- 分词结果 --&gt; D[分类模块] </pre> <p>接受主界面的输入请求，对输入进行分析传递给下一个模块。</p>
接口	<p>被调用接口：</p> <p>主界面调用，传入参数为一个输入字符串。</p> <p><b>String searchWord;</b></p> <p>调用接口：</p> <p>传递给分类模块，参数为分词结果。</p> <p><b>String keys[];</b></p> <p>局部数据结构：</p> <p>保存本地搜索结果的 note 列表；</p> <pre> Class results{ String contents []; String dates[]; String tags[]; } </pre> <p>分别用于保存搜索结果的 notes id，依次是内容结果、日期结果、标签结果。</p>
数据结构与算法	<p>数据结构采用一般的数据结构。</p> <p>分词算法采用中科院的 ICTCLAS（最新版本改名为 NLPIR）中文分词系统。</p> <pre> package com.nlpir;  import kevin.zhang.NLPIR;  public class TestUTF8 {      public static void main(String[] args) {          try {              testUTF8();          } catch (Exception e) { </pre>

	<pre>         e.printStackTrace();     } }  static void testUTF8() throws Exception {      // 创建接口实例     NLPir nlpir = new NLPir();      // NLPir_Init     if (!NLPir.NLPir_Init("./file/".getBytes("utf-8"), 1)) {          System.out.println("NLPir 初始化失败...");          return;     }      String temp = ;      // 要统一编码，否则分词结果会产生乱码     byte [] resBytes = nlpir.NLPir_ParagraphProcess(temp.getBytes("UTF-8"), 1);      nlpir.NLPir_FileProcess(utf8File.getBytes("utf-8"), utf8FileResult.getBytes("utf-8"), 1);      // 退出，释放资源     NLPir.NLPir_Exit();  } } </pre>
输入	主界面调用，传入参数为一个输入字符串。 <b>String searchWord;</b>
输出	传递给分类模块，参数为分词结果。 <b>String keys[];</b>
存储分配	保存本地搜索结果的 <b>note</b> 列表； <b>Class results{</b> <b>String contents [];</b> <b>String dates[];</b> <b>String tags[];</b> <b>}</b> 分别用于保存搜索结果的 <b>notes id</b> ，依次是内容结果、日期结果、标签结果。
测试要点	无
限制条件	无
补充说明	无

### 3.1.6 模块 A6 设计

模块名称	已完成界面模块
功能描述	对已完成笔记的界面布局和逻辑实现。
性能描述	对于已完成触点有确定的动作定义，对于不支持的动作定义无输出反馈，不会因为用户的自由操作或者过度操作导致系统触发异常而崩溃。
程序逻辑	

	 <pre> graph TD     A([主界面]) -- 点击触发 --&gt; B[已完成界面模块]     B --&gt; C[遍历Remind的isFinished属性]     C --&gt; D([显示结果])   </pre> <p>由主界面调用点击触发进入已完成界面。已完成界面显示已完成的备忘。</p>
接口	<p>调用接口： 无</p> <p>被调用接口： 主界面的调用，传入参数为在主界面存储的 Remind；</p> <p>Class Remind；</p> <p>局部数据结构： 本地使用结构 String noteid 来遍历 Remind；</p>
数据结构与算法	<p>数据结构采用自建的结构。</p> <pre> Class Remind{     noteName note;     String time;     Bool isFinished; }   </pre> <p>算法采用遍历的算法。</p>
输入	Class Remind;传入主界面 mainModule 里的 Remind 参数。
输出	显示已完成备忘的 Fragment
存储分配	分配一个本地的数据存储，用于遍历 Remind。
测试要点	无
限制条件	无
补充说明	无

### 3.1.7 模块 A7 设计

模块名称	导出模块
功能描述	触发导出操作。
性能描述	对于导出触点有确定的动作定义，对于不支持的动作定义无输出反馈，不会因为用户的自由操作或者过度操作导致系统触发异常而崩溃。

程序逻辑	<pre> graph TD     Start([主界面]) -- 点击触发 --&gt; ExportModule[导出模块]     ExportModule --&gt; SendRequest[发出导出请求]     SendRequest --&gt; ServerRequest[(服务器请求)]     ServerRequest --&gt; Decision{导出成功?}     Decision -- yes --&gt; ExportModule     Decision -- no --&gt; NextStep[ ]   </pre> <p>对于导出请求的完成，包括和服务器的交互。</p>
接口	<p>调用接口：</p> <p>将导出 message 发送给服务器，用户 ID 保存在结构中。</p> <pre> struct backupCommand{     commandID id;     String userid; } </pre> <p>被调用接口：</p> <p>主界面调用，输入数据为用户 ID；</p> <pre> String ID; </pre> <p>局部数据结构：</p> <p>局部开辟空间用于保存与客户端的交互信息结果。</p>
数据结构与算法	<p>采用普通的数据结构。</p> <p>采用简便的 socket 编程算法。</p>
输入	<p>主界面调用，输入数据为用户 ID；</p> <pre> String ID; </pre>
输出	<pre> struct backupCommand{     commandID id;     String userid; } </pre> <p>将导出 message 发送给服务器，用户 ID 保存在结构中</p>
存储分配	<p>局部开辟空间用于保存与客户端的交互信息结果。</p>
测试要点	<p>保证客户端和服务端交互时结果的一致性，保证服务器消息传递的正确性和与本地的同步性。处理好无响应和无网络的特殊情况。</p>
限制条件	无
补充说明	无

### 3.1.8 模块 A8 设计

模块名称	备忘属性模块
功能描述	设置对笔记添加备忘功能。
性能描述	对于添加备忘属性触点有确定的动作定义，对于不支持的动作定义无输出反馈，不会因为用户的自由操作或者过度操作导致系统触发异常

	而崩溃。
程序逻辑	<pre> graph TD     A([主界面]) -- 点击触发 --&gt; B[备忘属性模块]     B --&gt; C[选择备忘文件]     C --&gt; D[添加备忘时间]     D --&gt; E[确认备忘时间]     E --&gt; F[添加线程提醒备忘]     F --&gt; G{提示成功?}     G -- yes --&gt; A     G -- no --&gt; B </pre> <p>模块接收主界面调用，并完成添加备忘属性的功能。</p>
接口	<p>调用接口： 将笔记的 ID 传递给下一个模块。 <b>String ID;</b></p> <p>被调用接口： 主界面调用，传递要备忘的笔记 ID <b>String ID;</b></p> <p>局部数据结构： 无</p>
数据结构与算法	<p>采用朴素的数据结构。</p> <p>采用分支编程的基础算法。</p>
输入	<b>String ID;</b> 将笔记的 ID 传递给下一个模块。
输出	<b>String ID;</b> 主界面调用，传递要备忘的笔记 ID
存储分配	无
测试要点	保证备忘设置的稳定性。确保备忘提醒线程在后台运行良好。
限制条件	无
补充说明	无

## 3.2 子系统 B 的模块设计

### 3.2.1 模块 B1 设计

模块名称	备忘属性模块
------	--------

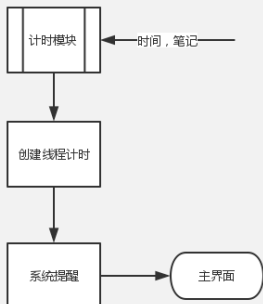
功能描述	设置对笔记添加备忘功能。
性能描述	对于添加备忘属性触点有确定的动作定义, 对于不支持的动作定义无输出反馈, 不会因为用户的自由操作或者过度操作导致系统触发异常而崩溃。
程序逻辑	<pre> graph TD     A([主界面]) -- 点击触发 --&gt; B[备忘属性模块]     B --&gt; C[选择备忘文件]     C --&gt; D[添加备忘时间]     D --&gt; E[确认备忘时间]     E --&gt; F[添加线程提醒备忘]     F --&gt; G{提示成功?}     G -- yes --&gt; A     G -- no --&gt; F </pre> <p>模块接收主界面调用, 并完成添加备忘属性的功能。</p>
接口	<p>调用接口:</p> <p>将笔记的 ID 传递给下一个模块。</p> <p><b>String ID;</b></p> <p>被调用接口:</p> <p>主界面调用, 传递要备忘的笔记 ID</p> <p><b>String ID;</b></p> <p>局部数据结构:</p> <p>无</p>
数据结构与算法	<p>采用朴素的数据结构。</p> <p>采用分支编程的基础算法。</p>
输入	<b>String ID;</b> 将笔记的 ID 传递给下一个模块。
输出	<b>String ID;</b> 主界面调用, 传递要备忘的笔记 ID
存储分配	无
测试要点	保证备忘设置的稳定性。
限制条件	无
补充说明	无

### 3.2.2 模块 B2 设计

模块名称	提醒时间模块
功能描述	设置对笔记提醒的时间。
性能描述	用户设置的时间格式必须和系统相匹配，从而好做比较。
程序逻辑	<pre> graph TD     A[备忘文件] --&gt; B[添加备忘时间]     B --&gt; C[调出添加备忘时间界面]     C --&gt; D[选择备忘时间]     D -- "时间, 笔记" --&gt; E[计时模块] </pre> <p>接受备忘模块参数，调出添加时间界面。</p>
接口	<p>调用接口：</p> <p>调用计时模块，对备忘进行计时。参数是笔记和备忘时间。</p> <pre> Class alert{     String ID;     String Date; } </pre> <p>被调用接口：</p> <p>由备忘属性模块调用，传入参数为笔记 ID</p> <p>局部数据结构：</p> <p>本地生成数据结构 Class alert.</p>
数据结构与算法	<p>采用数据结构</p> <pre> Class alert{     String ID;     String Date; } </pre> <p>采用 Android 编程的模块编程，调出设置时间模块。</p>
输入	String ID; 传入参数为笔记 ID
输出	Class alert; 参数是笔记和备忘时间。
存储分配	本地合成元素 Class alert，用于传递给下一个模块。
测试要点	测试时注意调用 Android 时间模块时得到的时间参数，用合适的数据结构转化。
限制条件	无
补充说明	无



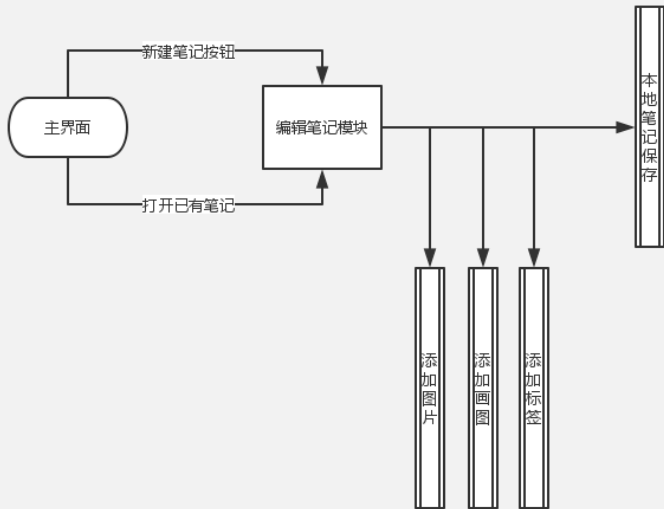
### 3.2.3 模块 B3 设计

模块名称	计时模块
功能描述	后台对备忘计时并提醒。
性能描述	传入数据结构的正确性
程序逻辑	 <p>对笔记进行计时。</p>
接口	<p>调用接口： Android 系统提醒接口。</p> <p>被调用接口： 提醒时间模块的时间和对应笔记。参数是 Class alert;</p> <p>局部数据结构： 创建线程。</p>
数据结构与算法	<p>采用普通的数据结构。</p> <p>采用线程级编程的编程算法。</p>
输入	Class alert; 提醒时间模块的时间和对应笔记
输出	返回主界面，并显示设置成功与否。
存储分配	本地线程资源分配。
测试要点	线程在后台运行的健壮性。在因为其他 App 异常而退出后依然能够重新启动。
限制条件	无
补充说明	无

## 3.3 子系统 C 的模块设计

### 3.3.1 模块 C1 设计

模块名称	编辑笔记模块
功能描述	实现客户端笔记编辑的界面布局，实现基本的文字输入功能，实现和子功能结构的接口，记录笔记创建和更新时间。
性能描述	<p>日期存储严格按照 Mon-Day-Year 的形式和精度。</p> <p>对于平铺于屏幕的所有触点都有相应的动作定义，对于不支持的动作定义无输出反馈，不会因为用户的自由操作或者过度操作导致系统触</p>

	发异常而崩溃。
程序逻辑	 <p>由编辑笔记触点和打开笔记触点触发进入编辑界面，在编辑界面由触点触发其他接口调用其它模块。</p>
接口	<p>被调用接口：</p> <ol style="list-style-type: none"> <li>3、主界面的编辑笔记触点 由主界面编辑笔记按钮触发新的 Activity，传入参数是笔记唯一标识的 ID，由主界面传入。 <b>String ID;</b></li> <li>4、主界面的打开笔记触点 由主界面某一笔记点击触发打开笔记的 Activity，传入参数是笔记唯一标识的 ID，由主界面传入。 <b>String ID;</b></li> </ol> <p>调用接口：</p> <ol style="list-style-type: none"> <li>5、添加图片 由 onClick 方法触发调用系统相册接口，传入参数为笔记 ID，调用模块结束后记录图片 ID 的图片备份。 <b>String ID;</b></li> <li>6、添加画图 由 onClick 方法触发新的 Activity，传入参数为笔记 ID，调用模块结束后记录图片 ID 和图片备份。 <b>String ID;</b></li> <li>7、添加标签 由在特定位置输入触发添加标签模块。输入为笔记 ID； <b>String ID;</b></li> <li>8、本地笔记保存 由按下保存笔记触发，返回主界面。输入参数为笔记 ID； <b>String ID;</b></li> </ol> <p>局部数据结构：</p>

	<pre> Class note{ String ID; String tags[]; String picture[]; String context; String editTime; } </pre> <p>保存笔记的唯一标识 ID、标签列表、图片列表、正文、编辑时间。</p>
数据结构与算法	数据结构采用经典的数据结构，算法采用非特殊的条件判别算法。
输入	<p>3、主界面的编辑笔记触点</p> <p>由主界面编辑笔记按钮触发新的 Activity，传入参数是笔记唯一标识的 ID，由主界面传入。</p> <pre>String ID;</pre> <p>4、主界面的打开笔记触点</p> <p>由主界面某一笔记点击触发打开笔记的 Activity，传入参数是笔记唯一标识的 ID，由主界面传入。</p> <pre>String ID;</pre>
输出	<p>5、添加图片</p> <p>由 onClick 方法触发调用系统相册接口，传入参数为笔记 ID，调用模块结束后记录图片 ID 的图片备份。</p> <pre>String ID;</pre> <p>6、添加画图</p> <p>由 onClick 方法触发新的 Activity，传入参数为笔记 ID，调用模块结束后记录图片 ID 和图片备份。</p> <pre>String ID;</pre> <p>7、添加标签</p> <p>由在特定位置输入触发添加标签模块。输入为笔记 ID；</p> <pre>String ID;</pre> <p>8、本地笔记保存</p> <p>由按下保存笔记触发，返回主界面。输入参数为笔记 ID；</p> <pre>String ID;</pre>
存储分配	<pre> Class note{ String ID; String tags[]; String picture[]; String context; String editTime; } </pre> <p>保存笔记的唯一标识 ID、标签列表、图片列表、正文、编辑时间。</p>

测试要点	保证存储空间的充足。 测试所有触点、触发方式的动作有效性，不会因为动作未定义或者数值越界导致异常而崩溃。
限制条件	无
补充说明	无

### 3.3.2 模块 C2 设计

模块名称	添加图片模块
功能描述	实现从系统添加图片到笔记的功能。
性能描述	调用系统图片接口，要求稳定性。
程序逻辑	<pre> graph TD     Start([ ]) -- 笔记 --&gt; AddImage[添加图片]     AddImage --&gt; CallInterface[调用系统图片接口，选择图片]     CallInterface --&gt; InsertNote[插入图片到笔记中]     InsertNote --&gt; Decision{插入成功?}     Decision -- no 提示失败 --&gt; EditInterface([编辑界面])     Decision -- yes 提示成功 --&gt; EditInterface   </pre> <p>由编辑界面传入，添加系统图片进入笔记。</p>
接口	调用接口： 系统相册接口 被调用接口： 编辑界面点击触发本模块，输入为笔记的 ID String ID; 局部数据结构：
数据结构与算法	采用普通的数据结构。 算法采用顺序结构的算法。
输入	String ID; 输入为笔记的 ID
输出	提示信息
存储分配	无
测试要点	对于本地相册权限访问控制的请求以及异常处理。
限制条件	无
补充说明	无

### 3.3.3 模块 C3 设计

模块名称	添加画图模块
功能描述	完成画图界面的逻辑实现和保存图片功能。

性能描述	无
程序逻辑	<pre> graph TD     Note[笔记] --&gt; AddImage[添加图片]     AddImage --&gt; Draw[进入绘图界面, 进行绘图。]     Draw --&gt; Save[保存绘图为图片, 插入图片到笔记中]     Save --&gt; Decision{插入成功?}     Decision -- "no 提示失败" --&gt; Edit[编辑界面]     Decision -- "yes 提示成功" --&gt; Edit   </pre> <p>基本和插入图片性质类似，需要自行添加绘图模块。</p>
接口	<p>调用接口： 无</p> <p>被调用接口： 主界面按钮点击触发本模块，传入参数为笔记 ID； String ID；</p> <p>局部数据结构： 本地需要一个足够长的 byte 字符串来记录当前轨迹。</p>
数据结构与算法	<p>本地数据结构</p> <pre> Class bytes{     Int x;     Int y; }   </pre> <p>算法： Android 实现双缓冲绘图，在内存中创建一片内存区域，把将要绘制的图片预先绘制到内存中，在绘制显示的时候直接获取缓冲区的图片进行绘制。更具体一点来说：先通过 setBitmap 方法将要绘制的所有的图形绘制到一个 Bitmap 上也就是先在内存空间完成，然后再来调用 drawBitmap 方法绘制出这个 Bitmap，显示在屏幕上。</p>
输入	传入参数为笔记 ID；
输出	插入结果提示
存储分配	本地需要一个足够大的 Class bytes[]用于服务双缓冲内存内容。
测试要点	绘图笔迹的保真性
限制条件	无
补充说明	无

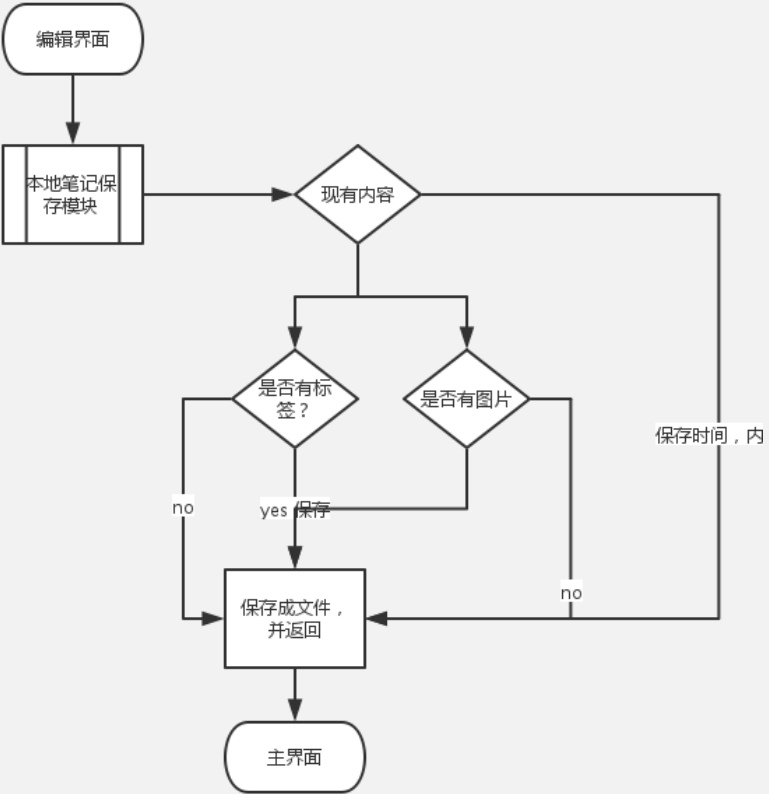
### 3.3.4 模块 C4 设计

模块名称	添加标签模块
------	--------

功能描述	实现本地全局标签的更新、同步和自动分类功能。
性能描述	用户输入标签，模块更新标签的实时性。
程序逻辑	用户输入标签，模块自动实时识别和更新。
接口	调用接口： 无 被调用接口： 编辑界面调用该模块，传入参数为 <b>String ID</b> 局部数据结构： 本地的标签数组
数据结构与算法	采用普通的数据结构。 采用轮询实时更新的算法。
输入	笔记的 ID
输出	无
存储分配	本地的笔记标签，标签是一个数组。
测试要点	更新标签的实时性。
限制条件	无
补充说明	无

### 3.3.5 模块 C5 设计

模块名称	本地笔记保存模块
功能描述	实现对本地笔记的保存。
性能描述	保存笔记的完整性，包括笔记创建时间、修改时间、标签、图片，一定是稳定的。

程序逻辑	 <pre> graph TD     A([编辑界面]) --&gt; B[本地笔记保存模块]     B --&gt; C{现有内容}     C --&gt; D{是否有标签?}     C --&gt; E{是否有图片?}     D -- no --&gt; F[保存成文件, 并返回]     D -- yes 保存 --&gt; F     E -- no --&gt; F     E -- yes 保存时间, 内容 --&gt; F     F --&gt; G([主界面])   </pre> <p>由编辑界面保存现有的笔记。</p>
接口	<p>调用接口： 无</p> <p>被调用接口： 由编辑界面调用，传入参数为 <b>String ID;</b></p> <p>局部数据结构： 局部存储笔记的所有信息。</p>
数据结构与算法	<p>采用普通的数据结构。</p> <p>采用顺序的算法。并使用 trid 树的插入算法：</p> <pre> algorithm insert(root : node, s : string, value : any):     node = root     i = 0     n = length(s)      while i &lt; n:         if node.child(s[i]) != nil:             node = node.child(s[i])             i = i + 1         else:             break   </pre>

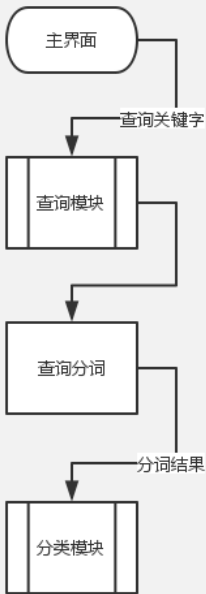
	<pre> (* append new nodes, if necessary *) while i &lt; n:     node.child(s[i]) = new node     node = node.child(s[i])     i = i + 1  node.value = value </pre>
输入	String ID;
输出	保存结果信息
存储分配	本地记录笔记的所有信息 String ID; 笔记 ID 和创建时间 String editDate; 笔记修改时间 String tags[]; 笔记标签 String note; 笔记内容 String pictures[]; 笔记插入的图片。
测试要点	无
限制条件	无
补充说明	无

## 3.4 子系统 D 的模块设计

### 3.4.1 模块 D1 设计

模块名称	查询模块
功能描述	实现查询关键字分词，识别功能。
性能描述	查询输入要求如果是日期的话严格按照日期格式的要求。Month-Day-Year。否则难以识别。

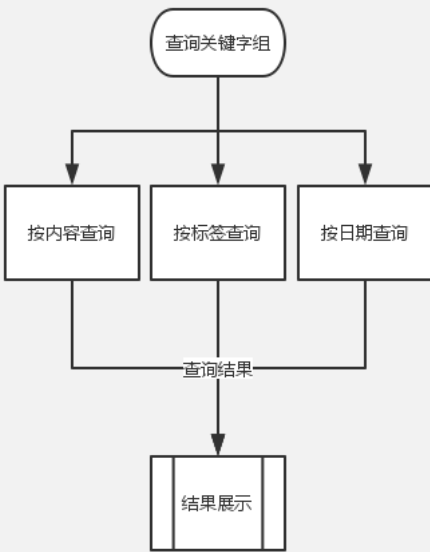


程序逻辑	 <pre>graph TD; A([主界面]) -- 查询关键字 --&gt; B[查询模块]; B --&gt; C[查询分词]; C -- 分词结果 --&gt; D[分类模块];</pre> <p>接受主界面的输入请求，对输入进行分析传递给下一个模块。</p>
接口	<p>被调用接口：</p> <p>主界面调用，传入参数为一个输入字符串。</p> <p><b>String searchWord;</b></p> <p>调用接口：</p> <p>传递给分类模块，参数为分词结果。</p> <p><b>String keys[];</b></p> <p>局部数据结构：</p> <p>保存本地搜索结果的 note 列表；</p> <pre>Class results{ String contents []; String dates[]; String tags[]; }</pre> <p>分别用于保存搜索结果的 notes id，依次是内容结果、日期结果、标签结果。</p>
数据结构与算法	<p>数据结构采用一般的数据结构。</p> <p>分词算法采用中科院的 ICTCLAS（最新版本改名为 NLPIR）中文分词系统。</p> <pre>package com.nlpir;  import kevin.zhang.NLPIR;  public class TestUTF8 {      public static void main(String[] args) {          try {              testUTF8();          } catch (Exception e) {</pre>

	<pre>         e.printStackTrace();     } }  static void testUTF8() throws Exception {      // 创建接口实例     NLPPIR nlpir = new NLPPIR();      // NLPPIR_Init     if (!NLPPIR.NLPPIR_Init("./file/".getBytes("utf-8"), 1)) {          System.out.println("NLPPIR 初始化失败...");          return;     }      String temp = ;      // 要统一编码，否则分词结果会产生乱码     byte [] resBytes = nlpir.NLPPIR_ParagraphProcess(temp.getBytes("UTF-8"), 1);      nlpir.NLPPIR_FileProcess(utf8File.getBytes("utf-8"), utf8FileResult.getBytes("utf-8"), 1);      // 退出，释放资源     NLPPIR.NLPPIR_Exit();  } } </pre>
输入	主界面调用，传入参数为一个输入字符串。 <b>String searchWord;</b>
输出	传递给分类模块，参数为分词结果。 <b>String keys[];</b>
存储分配	保存本地搜索结果的 <b>note</b> 列表； <b>Class results{</b> <b>String contents [];</b> <b>String dates[];</b> <b>String tags[];</b> <b>}</b> 分别用于保存搜索结果的 <b>notes id</b> ，依次是内容结果、日期结果、标签结果。
测试要点	无
限制条件	无
补充说明	无

### 3.4.2 模块 D2 设计

模块名称	分类模块
功能描述	实现查询，并对查询结果进行分类功能。
性能描述	无

程序逻辑	 <pre> graph TD     A([查询关键字组]) --&gt; B[按内容查询]     A --&gt; C[按标签查询]     A --&gt; D[按日期查询]     B --&gt; E[查询结果]     C --&gt; E     D --&gt; E     E --&gt; F[结果展示]   </pre> <p>对笔记进行分类查询。</p>
接口	<p>调用接口： 无</p> <p>被调用接口： 查询模块调入，输入参数为查询的关键字组</p> <p>局部数据结构： 保存查询结果。</p>
数据结构与算法	<p>对于每个关键字，分别保存对应的标签、日期、内容结果，对于关键字含义明确的只保存相应的查询结果。</p> <p>对日期的查询可采用 trid 树的查询算法：</p> <pre> def find(node, key):     for char in key:         if char in node.children:             node = node.children[char]         else:             return None   </pre>
输入	输入参数为查询的关键字组
输出	输出查询结果
存储分配	<p>对于每个关键字 keyword，保存以下数据结构：</p> <pre> Class searchResult{     String keyword;     String tags[];     String contents[];     String dates[]; }   </pre>

	用于分别保存查询结果的标签结果，内容结果，日期结果。
测试要点	无
限制条件	无
补充说明	无

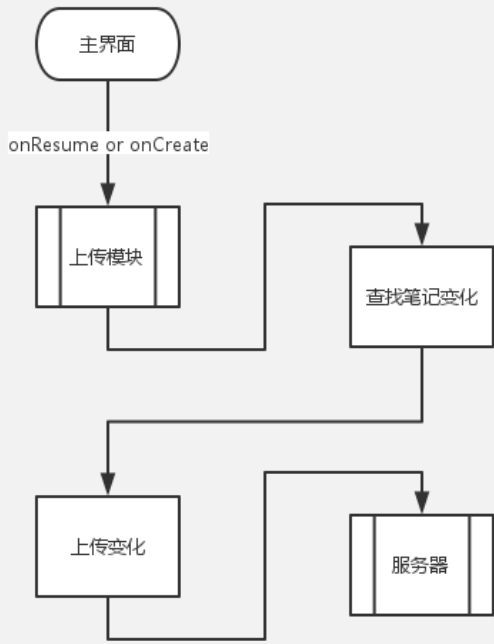
### 3.4.3 模块 D3 设计

模块名称	查询展示模块
功能描述	实现对查询结果的显示界面的布局。
性能描述	无
程序逻辑	对于得到的查询结果，建立一个 <b>fragment</b> 用于展示。
接口	调用接口： <b>Fragment</b> 接口 被调用接口： 分类模块调用，传入参数为 <b>Class searchResult[]</b> 数组； 局部数据结构： 无
数据结构与算法	采用 <b>searchResult</b> 的数据结构。 采用遍历的算法将每个查询结果从传入参数中提取出来。
输入	<b>Class searchResult[]</b> 数组；
输出	查询结果展示
存储分配	无需分配
测试要点	查询结果的可靠性，对于类别明显的关键字的识别成功率。
限制条件	无
补充说明	无

## 3.5 子系统 E 的模块设计

### 3.5.1 模块 E1 设计

模块名称	笔记上传模块
功能描述	实现对笔记的上传。
性能描述	对上传内容的比特要求尽量精简。

程序逻辑	 <pre> graph TD     A([主界面]) -- "onResume or onCreate" --&gt; B[上传模块]     B --&gt; C[查找笔记变化]     C --&gt; D[上传变化]     D --&gt; E[服务器] </pre> <p>在返回主界面 onResume 或者 onCreate 时调用上传模块。</p>
接口	<p>调用接口： 无</p> <p>被调用接口： 服务器端接口。</p> <p>局部数据结构： 局部分析变化，生成变化数据。</p>
数据结构与算法	<p>数据结构采用比特流。</p> <p>算法采用版本控制系统的版本变更算法，得到版本变更的比特流。</p>
输入	笔记数据
输出	笔记变更数据
存储分配	用比特流记录笔记变更。
测试要点	确保笔记变更记录的准确和最小数据记录。对于无网络情况的处理和重启动。
限制条件	无
补充说明	无

## 3.6 子系统 F 的模块设计

### 3.6.1 模块 F1 设计

模块名称	笔记下载模块
功能描述	实现对笔记的下载同步。
性能描述	对笔记记录的完整性

程序逻辑	<pre> graph TD     A[用户第一次登陆] --&gt; B[请求服务器]     B --&gt; C[得到用户笔记数据]     C --&gt; D[同步到本地] </pre> <p>在用户第一次登陆时进行下载同步。</p>
接口	<p>调用接口： 无，由登录模块完成后自动调用。</p> <p>被调用接口： 服务器的接口。</p> <p>局部数据结构： 要用到所有笔记的数据结构。 Class mainModule 用于标识笔记以及笔记的内容。</p>
数据结构与算法	采用笔记的数据结构。Class mainModule 算法采用普通的 socket 编程算法，用 socket 传递数据。
输入	无
输出	笔记的所有内容。
存储分配	要分配笔记的所有相关信息，尤其包括笔记的图片作为单独部分的附加同步 数据中。
测试要点	测试保证所有的笔记数据同步到本地。对于处理大量的同步数据是一个挑战。
限制条件	无
补充说明	无

## 3.7 子系统 G 的模块设计

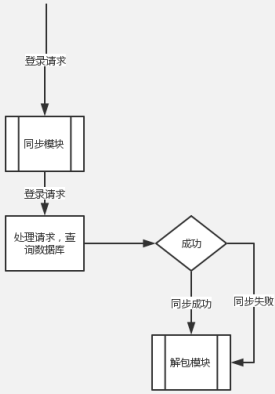
### 3.7.1 模块 G1 设计

模块名称	登录请求模块
功能描述	处理用户注册、登录。并响应。
性能描述	用户输入精度的要求取决于登录所需参数的精度要求：用户浏览的页面内如果需要用户输入相关的信息或参数将给出详细的数据类型说明，并且如果用户在非恶意的情况下 输入了错误的数据类型参数，系统将

	自动提示用户再次输入正确的参数
程序逻辑	<pre> graph TD     Start(( )) -- 登录请求 --&gt; Module1[解包模块]     Module1 -- 登录请求 --&gt; Process[处理请求, 查询数据库]     Process --&gt; Decision{成功}     Decision -- 登录成功 --&gt; Module2[解包模块]     Decision -- 登陆失败 --&gt; Module2   </pre> <p>对于用户的登录请求，返回登陆结果说明。</p>
接口	<p>调用接口： 调用数据库查询接口，传入的数据为用户名和密码</p> <p>被调用接口： 解包模块传入的处理请求，此模块接收的是</p> <pre> class loginCommand{     commandID id;     String userid;     String password; }   </pre> <p>局部数据结构： class loginCommand</p>
数据结构与算法	class loginCommand 数据结构用于表达登录请求。 算法处理即查询数据库处理匹配。
输入	class loginCommand 登录请求
输出	登录请求成功或失败信息
存储分配	无存储分配
测试要点	登录的安全性，保密性。
限制条件	无
补充说明	无

## 3.8 子系统 H 的模块设计

### 3.8.1 模块 H1 设计

模块名称	笔记同步模块
功能描述	处理用户多平台同步、笔记更新同步逻辑实现。
性能描述	无
程序逻辑	 <p>和登录请求基本处理流程相同，只是交付的处理单元不同。</p>
接口	<p>调用接口： 调用数据库插入接口，完成笔记同步。</p> <p>被调用接口： 由解包模块调用，采取以下数据结构</p> <pre>Class updateCommand{     commandID id;     String userid;     Byte s }</pre> <p>局部数据结构： Class updateCommand</p>
数据结构与算法	采用 Class updateCommand 的数据结构保存同步内容。 算法采用一般的数据库插入算法。
输入	Class updateCommand 同步请求
输出	同步结果消息
存储分配	无本地存储分配
测试要点	同步的可靠性，以及数据库检查出错时的处理。
限制条件	无
补充说明	无



## 3.9 子系统 I 的模块设计

### 3.9.1 模块 I1 设计

模块名称	笔记导出模块
功能描述	处理导出请求
性能描述	无
程序逻辑	<pre> graph TD     Start(( )) -- 导出请求 --&gt; ExportModule[导出模块]     ExportModule -- 登录请求 --&gt; Process[处理请求, 查询数据库, 打包, 导出]     Process --&gt; Success{成功}     Success -- 导出成功 --&gt; UnpackModule[解包模块]     Success -- 导出失败 --&gt; UnpackModule     UnpackModule --&gt; Success   </pre> <p>和登录请求基本处理流程相同，只是交付的处理单元不同。</p>
接口	<p>调用接口：</p> <p>调用数据库 select 接口，完成笔记导出。</p> <p>被调用接口：</p> <p>由解包模块调用，采取以下数据结构</p> <pre> Class backupCommand{     commandID id;     String userid; }   </pre> <p>局部数据结构：</p> <pre> Class backupCommand   </pre>
数据结构与算法	采用 Class backupCommand 的数据结构导出用户数据。 算法采用一般的数据库 select 算法。
输入	Class backupCommand 导出请求
输出	导出结果消息
存储分配	无本地存储分配
测试要点	导出的成功率。
限制条件	无
补充说明	无

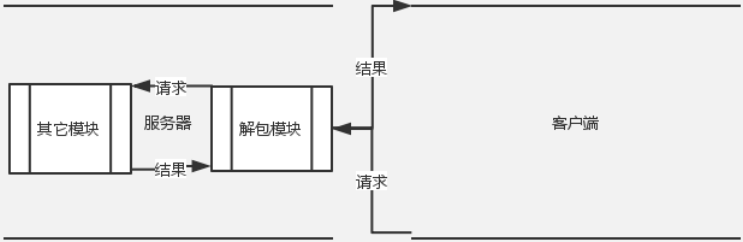
## 3.10 子系统 J 的模块设计

### 3.10.1 模块 J1 设计

模块名称	数据库更新模块
功能描述	更新数据库。
性能描述	用户输入精度的要求取决于相应功能所需参数的精度要求：用户浏览的页面内如果需要用户输入相关的信息或参数将给出详细的数据类型说明，并且如果用户在非恶意的情况下 输入了错误的数据类型参数，系统将自动提示用户再次输入正确的参数。
程序逻辑	<p>对于不同的请求，执行数据库操作语言得到结果返回给请求模块</p> <pre> graph TD     A[登录请求] --&gt; B[ ]     C[导出请求] --&gt; B     D[同步请求] --&gt; B     B --&gt; E[输出结果]     style B fill:#fff,stroke:#000,stroke-width:1px   </pre>
接口	<p>调用接口： 数据库操作</p> <p>被调用接口： 1、登录模块 2、同步模块 3、导出模块</p> <p>局部数据结构：   Class updateCommand  Class backupCommand  Class loginCommand</p>
数据结构与算法	<p>数据结构：   Class updateCommand  Class backupCommand  Class loginCommand</p> <p>算法 采用数据库操作的经典算法。</p>
输入	更新、登录、导出三个请求之一
输出	数据库操作结果
存储分配	对数据库结果进行保存。
测试要点	无
限制条件	无
补充说明	无

## 3.11 子系统 K 的模块设计

### 3.11.1 模块 K1 设计

模块名称	解包模块
功能描述	处理客户端发送的 socket 包
性能描述	用户输入精度的要求取决于相应功能所需参数的精度要求：用户浏览的页面内如果需要用户输入相关的信息或参数将给出详细的数据类型说明，并且如果用户在非恶意的情况下 输入了错误的数据类型参数，系统将自动提示用户再次输入正确的参数。
程序逻辑	 <p>所有经由服务器的数据均由解包模块进行处理</p>
接口	<p>调用接口：</p> <p>客户端的请求模块</p> <p>服务器端的响应模块</p> <p>被调用接口：</p> <p>客户端的请求模块</p> <p>服务器端的响应模块</p> <p>局部数据结构：</p> <pre>Class command{     Byte b; }</pre> <p>概括之，就是比特流的封包和解包。</p>
数据结构与算法	<p>采用比特流的数据结构。</p> <p>用 socket 的方法对数据进行封包和解包。</p>
输入	比特流或者数据
输出	比特流或者数据
存储分配	无需本地存储。
测试要点	比特流编码的正确和一致有效。
限制条件	无
补充说明	无

## 4 数据库设计

### 4.1 数据库环境说明

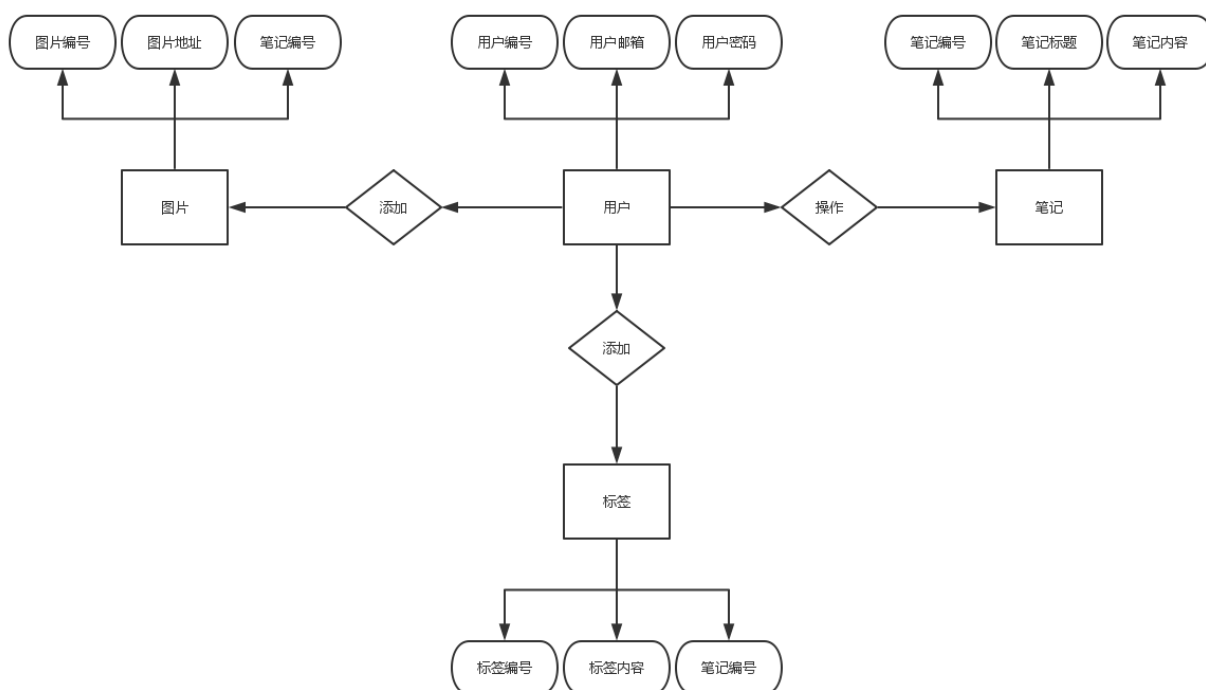
系统采用 Mysql 数据库。在服务器端采用 python 的编程语言和 PYMysql 数据库接口。数据库名字为 NoteDatabase

### 4.2 数据库的命名规则

数据表的命名规则为 XXtable，其中 xx 为数据表存储的内容的小写英文单词。表中每一个字段以字段代表含义的小写英文单词命名。

### 4.3 逻辑设计

#### 4.3.1 逻辑设计的 E-R 图



### 数据库逻辑结构设计

用户表: usertable

列名	数据类型	可否为空	说明
<b>UserID</b>	Int	Not NULL	Key
<b>Useremail</b>	Varchar	Not NULL	用户邮箱
<b>Userpassword</b>	Char	Not NULL	用户密码

笔记表: notetable

列名	数据类型	可否为空	说明
<b>NoteID</b>	Int	Not NULL	Key
<b>NoteName</b>	Nvarchar	NULL	笔记标题
<b>NoteCont</b>	Ntext	NULL	笔记文本内容
<b>SetTime</b>	Smalldatetime	Not NULL	笔记创建时间
<b>UpdateTime</b>	Smalldatetime	Not NULL	笔记最后更新时间
<b>WarTime</b>	Datetime	NULL	备忘提醒时间
<b>Finished</b>	Int	Not NULL	是否完成

图片表: picturetable

列名	数据类型	可否为空	说明
<b>PicID</b>	Int	Not NULL	Key
<b>PicCont</b>	Varchar	Not NULL	图片地址
<b>NoteID</b>	Int	Not NULL	对应笔记的 ID

标签表: labeltable

列名	数据类型	可否为空	说明
<b>LabelID</b>	Int	Not NULL	Key
<b>LabelName</b>	Nvarchar	Not NULL	标签名
<b>NoteID</b>	Int	Not NULL	对应笔记的 ID

## 4.4 物理设计

### 数据库环境

数据库采用 MySQL。

数据存储可采用 RAID5+RAID1 磁盘阵列的方式。

内存配置在 8G，来减少磁盘读取时间。

### 数据库参数设计

#### 数据库类型

由于数据库目标为企业级数据仓库，数据库类型选择 data warehouse 类型。

连接方式：同时连接类型选择专用方式连接，来满足数据装载时的大量批处理服务。

内存配置：根据服务器实际物理内存的大小，选择 70%-80%的内存作为数据库内存大小。

字符集：为了使数据库能够正确支持多国语言，需要将数据库字符集配置为 UTF-8 字符集。

其他参数：聚合内存使用，连接数最大为 10、数据块大小 2M、缓冲区大小为 100M。

#### 数据库存储设计

控制文件：控制文件中包含数据库重要信息，需要将控制文件存放在多个磁盘中，来保证数据库可恢复性。控制文件中参数设置，最大的数据文件数量不能小于数据库参数 db\_files。

日志文件：数据仓库通常为批处理装载，在装载时会产生大量日志。可选择关闭某些事实表日志，对通常的维表及高频率装载的数据表，可以选择打开日志功能。日志文件的大小由数据库事务处理量决定，在设计过程中，确保每 20 分钟切换一个日志文件。对于数据仓库系统，日志文件大小通常为几百兆到几千兆。为了确保日志能够镜像作用，每日志组的成员为 2 个，日志文件组为 5-10 组。

回滚段配置:  $\text{Undospace} = \text{UR} * \text{UPS} * \text{db\_block\_size} + \text{冗余量}$ 。UR: 表示在 undo 中保持的最长时间数 (秒), 由数据库参数 UNDO\_RETENTION 值决定。UPS: 表示在 undo 中, 每秒产生的数据库块数量。

临时段表空间配置: 数据库临时段表空间根据实际生产环境情况调整其大小, 表空间属性为自动扩展。

系统表空间配置: 系统表空间大小 1G 左右, 除了存放数据库数据字典的数据外, 其他数据不得存储在系统表空间。

表空间大小定义: 当表空间大小小于操作系统对最大文件限制时, 表空间由一个文件组成。如果表空间大小大于操作系统对最大文件限制时, 该表空间由多个数据文件组成, 表空间的总大小为估算为:  $\text{Tablespace} + \text{sum}(\text{数据段} + \text{索引段}) * 150\%$ 。

表空间扩展性设计原则: 表空间数据文件采用自动扩展的方式, 扩展容量快大小按 2 的整数倍 (1M、2M、4M、8M、16M、32M、64M) 进行扩展, 创建表空间时尽量采用 nologing 选项。表空间的最大限制一般采用 unlimited, 除非确切知道表空间数据文件的最大使用范围。

(64 位系统文件最大 128G), 最大为 2G。表空间采用 local 管理方式。

### 特殊表设计

分区表: 对于数据量比较大的表, 根据表数据的属性进行分区, 以得到较好的性能。如果表按某些字段进行增长, 则采用按字段值范围进行范围分区; 如果表按某个字段的几个关键值进行分布, 则采用列表分区; 对于静态表, 则采用 hash 分区或列表分区; 在范围分区中, 如果数据按某关键字段均衡分布, 则采用子分区的复合分区方法。

聚簇表: 如果某几个静态表关系比较密切, 则可以采用聚簇表的方法。

### 完整性设计

主键约束: 关联表的父表要求有主键, 主键字段或组合字段必须满足非空属性和唯一性要求。对于数据量比较大的父表, 要求指定索引段。

外键关联: 对于关联两个表的字段, 一般应该分别建立主键、外键。实际是否建立外键, 根据对数据完整性的要求决定。为了提高性能, 对于数据量比较大的表要求对外键建立索引。

### 索引设计

对于查询中需要作为查询条件的字段, 可以考虑建立索引。最终根据性能的需要决定是否建立索引。对于复合索引, 索引字段顺序比较关键, 把查询频率比较高的字段排在索引组合的最前面。在分区表中, 尽量采用 local 分区索引以方便分区维护。

### 视图设计

视图是虚拟的数据库表, 在使用时要遵循以下原则: 从一个或多个库表中查询部分数据项; 为简化查询, 将复杂的检索或字查询通过视图实现; 提高数据的安全性, 只将需要查看的数据信息显示给权限有限的人员;

视图中如果嵌套使用视图, 级数不得超过 3 级;

由于视图中只能固定条件或没有条件, 所以对于数据量较大或随时间的推移逐渐增多的库表, 不宜使用视图; 可以采用实体化视图代替。

视图中尽量避免出现数据排序的 SQL 语句。

### 包设计

存储过程、函数、外部游标必须在指定的数据包对象

PACKAGE 中实现。存储过程、函数的建立如同其它语言形式的编程过程, 适合采用模块化设计方法; 当具体算法改变时, 只需要修改需要存储过程即可, 不需要修改其它语言的源程序。当和数据库频繁交换数据是通过存储过程可以提高运行速度, 由于只有被授权的用户才能执行存储过程, 所以存储过程有利于提高系统的安全性。

存储过程、函数必须检索数据库表记录或数据库其他对象, 甚至修改 (执行 Insert、Delete、

Update、Drop、Create 等操作) 数据库信息。如果某项功能不需要和数据库打交道, 则不得通过数据库存储过程或函数的方式实现。在函数中避免采用 DML 或 DDL 语句。

在数据包采用存储过程、函数重载的方法, 简化数据包设计, 提高代码效率。存储过程、函数必须有相应的出错处理功能。

### 安全性设计

管理默认用户: 在生产环境中, 必须严格管理 sys 和 system 用户, 必须修改其默认密码, 禁止用该用户建立数据库应用对象。删除或锁定数据库测试用户。

数据库级用户权限设计: 必须按照应用需求, 设计不同的用户访问权限。包括应用系统管理用户, 普通用户等, 按照业务需求建立不同的应用角色。用户访问另外的用户对象时, 应该通过创建同义词对象 synonym 进行访问。

角色与权限: 确定每个角色对数据库表的操作权限, 如创建、检索、更新、删除等。每个角色拥有刚好能够完成任务的权限, 不多也不少。在应用时再为用户分配角色, 则每个用户的权限等于他所兼角色的权限之和。

应用级用户设计: 应用级的用户帐号密码不能与数据库相同, 防止用户直接操作数据库。用户只能用帐号登录到应用软件, 通过应用软件访问数据库, 而没有其它途径操作数据库。

用户密码管理: 用户帐号的密码必须进行加密处理, 确保在任何地方的查询都不会出现密码的明文。

### 备份恢复设计原则

数据库热备份恢复: 数据库通常提供了数据快速的热备份和热恢复手段, 提供了数据库级、用户级和表级的数据备份恢复方式。这种方法一般作为数据库辅助备份手段。

数据库冷备份原则: 数据库冷备份必须符合以下原则: 数据库容量比较小。数据库允许关闭的情况。

数据库级备份原则: 在数据库的数据量比较小, 或数据库初始建立的情况下采用。不适合 7\*24 的在线生产环境数据库备份。

用户级备份原则: 在用户对象表数据容量比较小、或则用户对象初始建立的情况下使用。

表级备份原则: 主要在以下场合采用的备份方式: 参数表备份、静态表备份、分区表的分区备份。

## 4.4.1 数据库表一览表

表名	功能说明
Usertable	用户表
Notetable	笔记表
Picturetable	图片表
Labeltable	标签表

## 4.4.2 数据库表定义

用户表: usertable

列名	数据类型	可否为空	说明
UserID	Int	Not NULL	Key
Useremail	Varchar	Not NULL	用户邮箱
Userpassword	Char	Not NULL	用户密码

笔记表: notetable

列名	数据类型	可否为空	说明
<b>NoteID</b>	Int	Not NULL	Key
<b>Notename</b>	Nvarchar	NULL	笔记标题
<b>Notecont</b>	Ntext	NULL	笔记文本内容
<b>Settime</b>	Smalldatetime	Not NULL	笔记创建时间
<b>Uptime</b>	Smalldatetime	Not NULL	笔记最后更新时间
<b>Warntime</b>	Datetime	NULL	备忘提醒时间
<b>Finished</b>	Int	Not NULL	是否完成

图片表: picturetable

列名	数据类型	可否为空	说明
<b>PicID</b>	Int	Not NULL	Key
<b>Piccont</b>	Varchar	Not NULL	图片地址
<b>NoteID</b>	Int	Not NULL	对应笔记的 ID

标签表: labeltable

列名	数据类型	可否为空	说明
<b>LabelID</b>	Int	Not NULL	Key
<b>Labelname</b>	Nvarchar	Not NULL	标签名
<b>NoteID</b>	Int	Not NULL	对应笔记的 ID

## 4.5 安全性设计

### 4.5.1 防止用户直接操作数据库的方法

定义视图，授予不同角色不同的权限。

### 4.5.2 用户帐号密码的加密方法

采用 MD5 加密方式存储密码信息，改变密码字段 Password 的类型为 16 位二进制数方式。

### 4.5.3 角色与权限

【确定每个角色对数据库表的操作权限，如创建、检索、更新、删除等。】

角色	可以访问的表与列	操作权限
管理员	所有的表与列	所有
注册用户	笔记表	创建、检索、更新、删除
	图片表	创建、检索、更新、删除
	标签表	创建、检索、更新、删除



## 4.6 数据库管理与维护说明

数据的管理和维护基于 Mysql 系统自身的功能进行，使用该管理器的导入数据，导出数据，备份数据库和还原数据库等功能对数据库进行管理和维护。

## 5 界面设计

### 5.1 安卓版界面

#### 5.1.1 登录界面



登录界面说明：

注册邮箱：注册邮箱即用户注册时所使用的邮箱。

密码：密码即用户注册时所设置的密码。

自动登录：勾选自动登录选项后每次打开应用就会自动登录，无需输入注册邮箱和密码，登录成功直接进入主界面。

注册：没有注册的用户需要点击注册选项进行注册。

## 5.1.2 主界面



主界面说明：

左上角的 **Backup**: 该选项用于备份，按下该按键则会向服务器端备份笔记。

右上角的 “+”：该选项用于创建新的笔记，点击该选项会进入编辑笔记界面。

右上角的放大镜图标：该选项用于搜索。

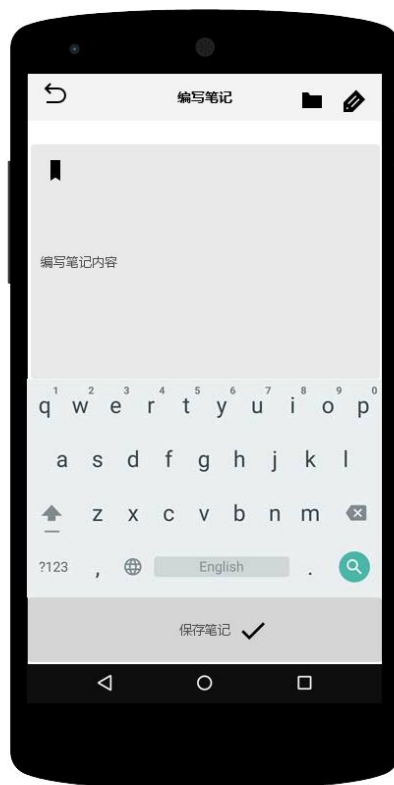
中间的标签：每个比较都有一个标签，标签会显示笔记标题和第一行的内容，右端显示最后编辑的时间。向左滑动标签会有动画效果，同时出现三个选项。第一个是闹钟图标，用于设置备忘；第二个是对勾图标，用于确认是否完成，确认完成的任务将会从笔记备忘栏中移除并加入到已完成栏中；第三个是垃圾桶图标，用于删除笔记。

左下角的 “笔记备忘”：笔记备忘处于高亮状态说明这时显示的是笔记备忘的标签。

下方中间的 “\*\*\*条笔记”：\*\*\*表示的是数字，该选项用于表示栏目中的笔记条数。

右下角的 “已完成”：高亮时表示显示的是已完成栏目。

### 5.1.3 笔记编辑界面



笔记编辑界面说明：

左上角的返回按钮：点击后返回主界面，不会修改原笔记。

右上角的“文件图标”：点击该图标可以向笔记内添加图片。

右上角的“画笔图标”：点击该图标后可以向笔记内画图。

中间空白：空白处会有光标，通过键盘可以向光标处输入字符。

下方“保存笔记”按钮：点击后会保存编辑后的笔记并返回主界面。

## 5.2 ios 版界面

### 5.2.1 登录界面



登录界面说明：

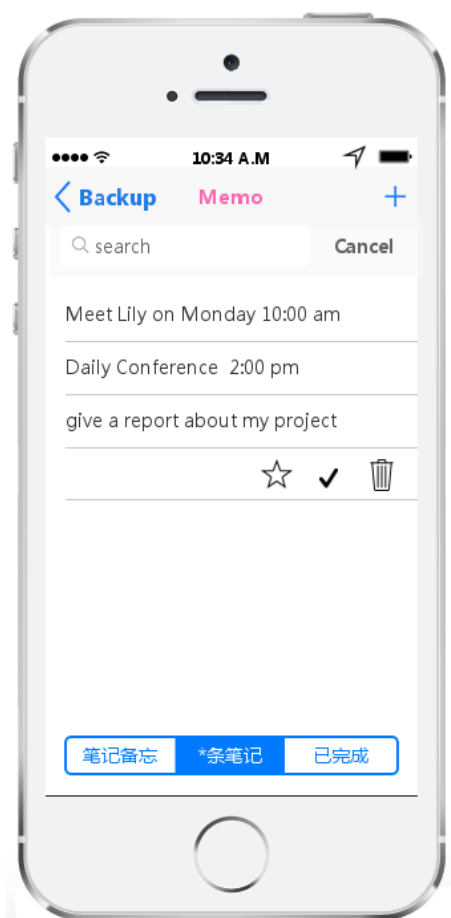
注册邮箱：注册邮箱即用户注册时所使用的邮箱。

密码：密码即用户注册时所设置的密码。

自动登录：勾选自动登录选项后每次打开应用就会自动登录，无需输入注册邮箱和密码，登录成功直接进入主界面。

注册：没有注册的用户需要点击注册选项进行注册。

## 5.2.2 主界面



主界面说明：

左上角的 **Backup**: 该选项用于备份，按下该按键则会向服务器端备份笔记。

右上角的 “+”：该选项用于创建新的笔记，点击该选项会进入编辑笔记界面。

中间偏上的搜索项：该选项用于搜索。

中间的标签：每个比较都有一个标签，标签会显示笔记标题和第一行的内容，右端显示最后编辑的时间。向左滑动标签会有动画效果，同时出现三个选项。第一个是闹钟图标，用于设置备忘；第二个是对勾图标，用于确认是否完成，确认完成的任务将会从笔记备忘栏中移除并加入到已完成栏中；第三个是垃圾桶图标，用于删除笔记。

左下角的 “笔记备忘”：笔记备忘处于高亮状态说明这时显示的是笔记备忘的标签。

下方中间的 “\*\*\*条笔记”：\*\*\*表示的是数字，该选项用于表示栏目中的笔记条数。

右下角的 “已完成”：高亮时表示显示的是已完成栏目。

### 5.2.3 笔记编辑界面



笔记编辑界面说明：

左上角的返回按钮：点击后返回主界面，不会修改原笔记。

右上角的“+”：点击该图标可以向笔记内添加图片。

左上角的“+”：点击该图标后可以选择向笔记内画图。

中间空白：空白处会有光标，通过键盘可以向光标处输入字符。

下方“完成”按钮：点击后会保存编辑后的笔记并返回主界面。

## 5.3 web 版界面

### 5.3.1 登录界面



笔记&备忘

用户邮箱：

密码：

☐ 记住密码

提交 注册

登录界面说明：

用户邮箱：用户邮箱即用户注册时所使用的邮箱。

密码：密码即用户注册时所设置的密码。

记住密码：勾选自动登录选项后每次打开应用就会自动登录，无需输入注册邮箱和密码，登录成功直接进入主界面。

注册：没有注册的用户需要点击注册选项进行注册。

### 5.3.2 主界面



主界面说明：

“新建笔记”按钮：点击新建笔记，进入笔记编辑界面。

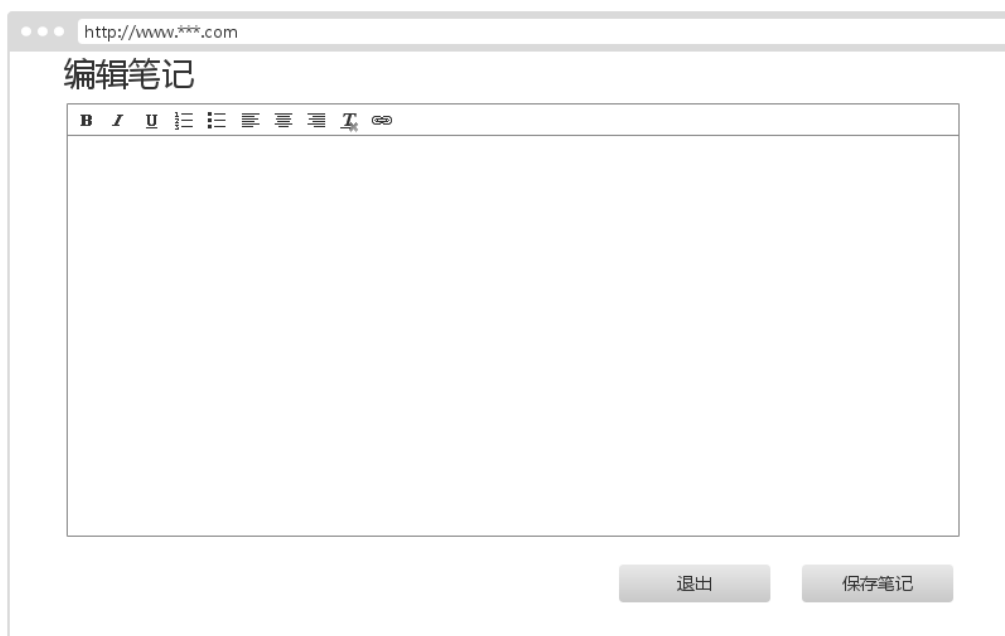
“搜索”按钮：点击搜索，在用户的所有笔记（包括已完成和未完成）中搜索搜索框中的内容。

“删除”按钮：点击删除该笔记。

已完成选项：勾选则将笔记加入已完成队列。



### 5.3.3 笔记编辑界面



笔记编辑界面说明：

编辑框：在编辑框中可以编辑笔记。

退出：点击退出笔记编辑界面，而且不保存已修改的内容。

保存笔记：点击保存笔记。

## 5.4 美学设计

我们的设计理念是实现一个简洁的轻量级笔记&备忘应用程序，所以在界面设计上我们尽量做到简洁轻便。与此同时，我们使用平面化的抽象图标，使得各个按钮既美观又易于理解。

在配色上我们首先使用的是自己喜欢的颜色，在产品推出后我们会听取用户的意见后再做调整。

## 5.5 界面资源设计

### 5.5.1 图标资源

#### 5.5.1.1 应用程序图标



安卓版和 ios 版图标如上，web 版没有图标。

#### 5.5.1.2 安卓登录&安卓注册按钮



按钮样式如上图红色框中所示。

### 5.5.1.3 自动登录勾选



选项样式如上图红色框中所示。

### 5.5.1.4 安卓 backup 按钮



按钮样式如上图红色框中所示。

### 5.5.1.5 安卓创建笔记按钮



创建笔记按钮如红色框中所示。

### 5.5.1.6 安卓搜索按钮



搜索按钮如红色框中所示。

### 5.5.1.7 安卓笔记文档按钮



如图中红色框所示，点击即可进入该笔记的编辑界面。

### 5.5.1.8 安卓设置提醒按钮



如图中红色框中所示，点击进入提醒设置。

### 5.5.1.9 安卓确认完成按钮



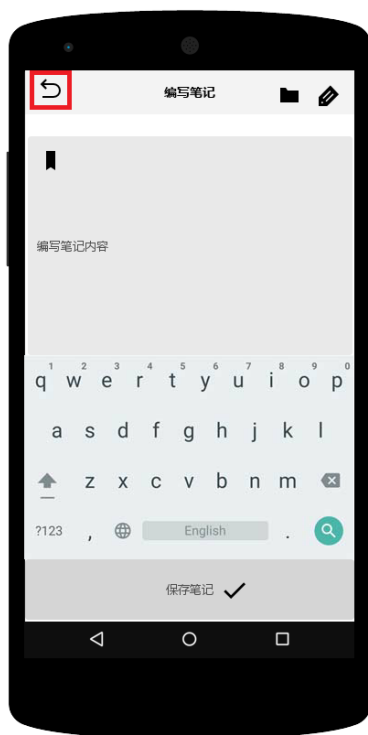
如图红色框中所示，点击该按钮将笔记从“笔记备忘”栏移入“已完成”栏。

### 5.5.1.10 安卓笔记删除按钮



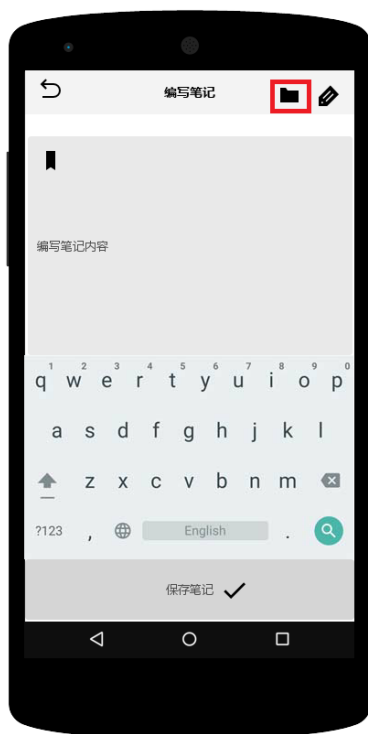
如上图红色框中所示，点击该按钮将删除该条笔记。

#### 5.5.1.11 安卓编辑笔记返回按钮

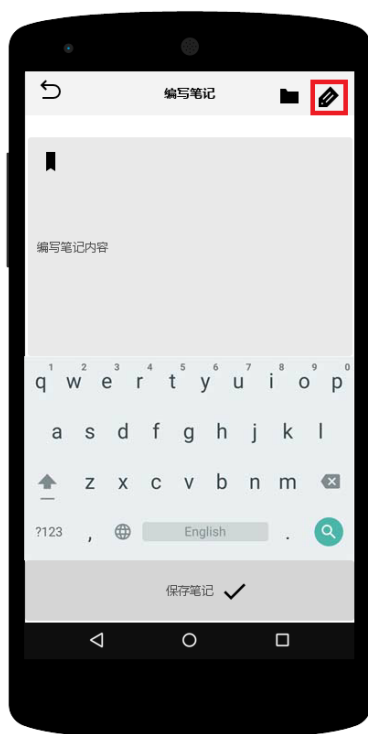


如图所示，红色框中即为编辑模式中的返回按钮，注意从该按钮退出编辑模式不会保存已修改的内容。

## 5.5.1.12 安卓添加本地图片按钮



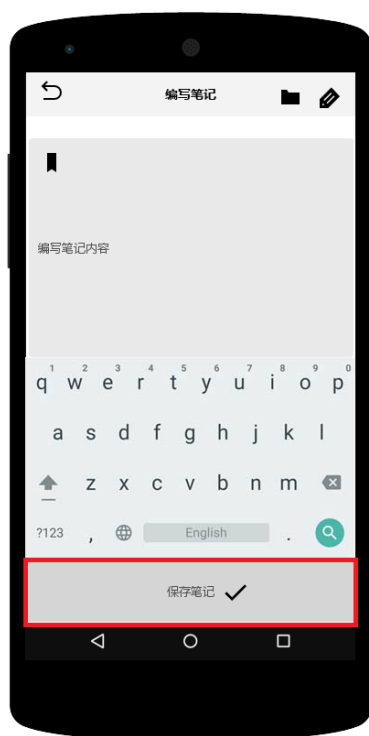
## 5.5.1.13 安卓添加画图按钮



如上图红色框中所示，点击后可在屏幕上画图并添加到笔记中。



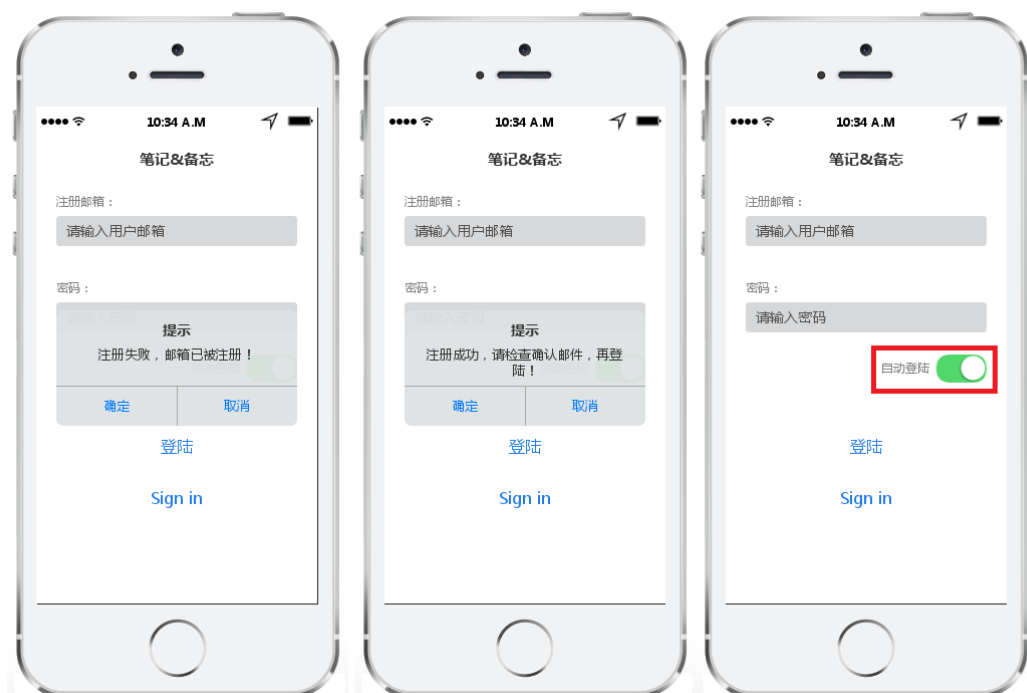
## 5.5.1.14 安卓保存笔记按钮



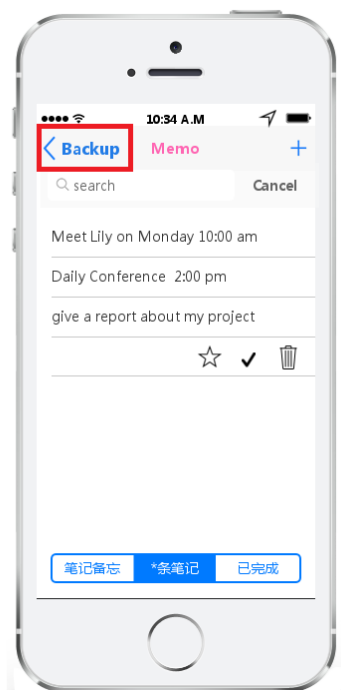
## 5.5.1.15 ios 登录&amp;注册按钮



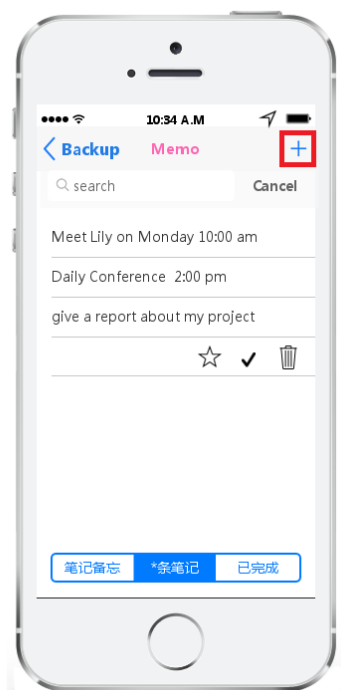
## 5.5.1.16 ios 自动登录选项



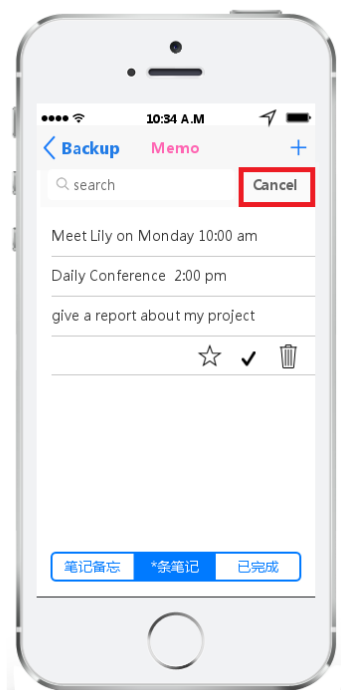
## 5.5.1.17 ios backup 按钮



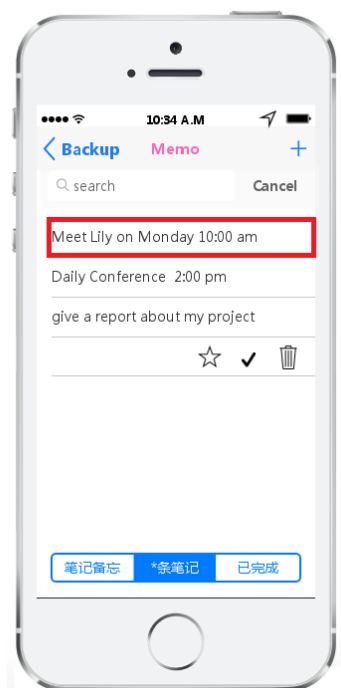
### 5.5.1.18 ios 创建笔记按钮



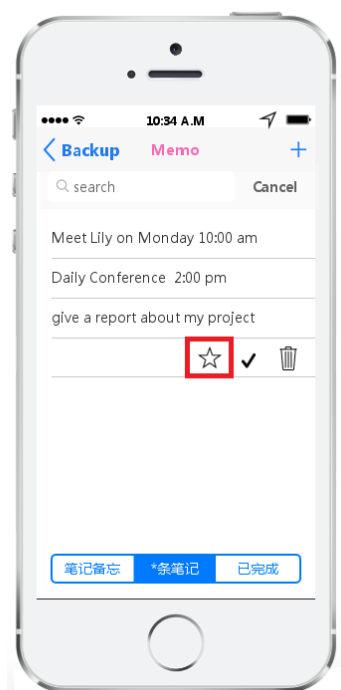
### 5.5.1.19 ios 搜索 cancel 按钮



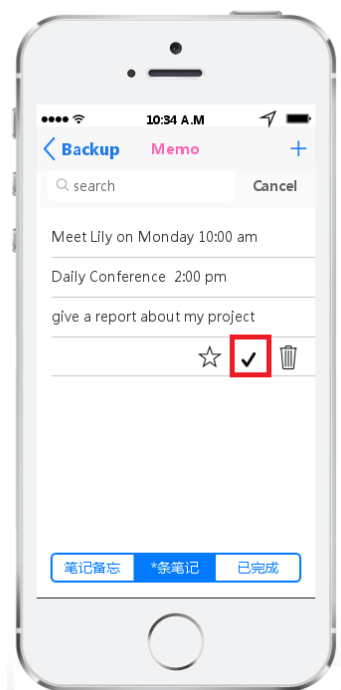
## 5.5.1.20 ios 笔记文档按钮



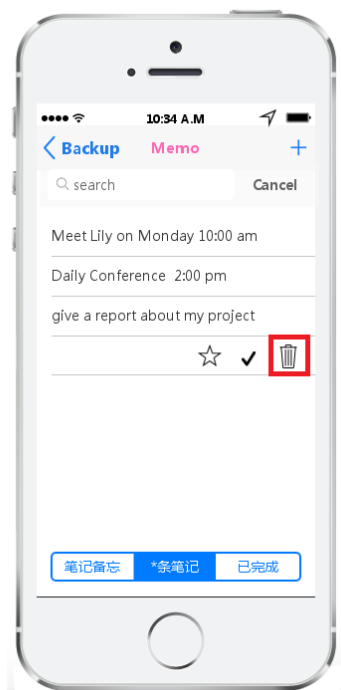
## 5.5.1.21 ios 设置提醒按钮



### 5.5.1.22 ios 确认完成按钮



### 5.5.1.23 ios 笔记删除按钮



## 5.5.1.24 ios 编辑笔记返回按钮



### 5.5.1.25 ios 添加本地图片&画图按钮



## 5.5.2 图像资源

图像资源相对较少，已包含在上述说明中。

## 5.5.3 界面组件

### 5.5.3.1 输入框



如上图红色框中所示。



### 5.5.3.2 状态条



## 6 开发任务分配

编号	任务/模块	描述	承担人/组	开始时间	预计完成时间	备注
1	Web 开发	利用 Django 框架搭建 Web 应用	刘慧琦	5. 24	6. 14	
2	Android 开发	利用 Android Studio 创建应用	刘慧琦、夏昊	6. 14	7. 04	
3	iOS 开发	利用 xcode 创建 iOS 应用	夏昊	7. 04	8. 24	
