**学长资源传承app设计规约说明书**

**计科04小组**

**组长** 朱 凌13349171

**组员** 张雨潇 13349155

郑玉婷 13349166

郑万山 13331360

张华键 13349152

**目录**

[1.工作范围 2](#_Toc450252140)

[1.1.系统目标 2](#_Toc450252141)

[1.2.运行环境 2](#_Toc450252142)

[1.3.主要软件需求 3](#_Toc450252143)

[2.体系结构设计 3](#_Toc450252144)

[2.1.数据流与控制流复审 3](#_Toc450252145)

[2.2.导出的程序结构 5](#_Toc450252146)

[3.数据库设计 5](#_Toc450252147)

[3.1.标识符和状态 5](#_Toc450252148)

[3.2.命名约定 5](#_Toc450252149)

[3.3.数据对象与形成的数据结构 5](#_Toc450252150)

[3.4.文件和数据库结构 7](#_Toc450252151)

[3.5.数据库实施 8](#_Toc450252152)

[4.接口设计 11](#_Toc450252153)

[4.1.人机界面设计规则 11](#_Toc450252154)

[4.2.人机界面设计过程 11](#_Toc450252155)

[4.3.外部接口设计 12](#_Toc450252156)

[4.4.内部接口设计 12](#_Toc450252157)

[5.各部件的过程设计 12](#_Toc450252158)

[5.1.部件设计与算法描述 12](#_Toc450252159)

[5.2.设计语言描述 15](#_Toc450252160)

[6.运行设计 15](#_Toc450252161)

[6.1.运行部件组合 15](#_Toc450252162)

[6.2.运行控制 15](#_Toc450252163)

[6.3.运行时间 16](#_Toc450252164)

[7.出错处理设计 16](#_Toc450252165)

[7.1.网络错误提示 16](#_Toc450252166)

[7.2.输入错误提示 16](#_Toc450252167)

[8.安全保密设计 17](#_Toc450252168)

[8.1.访问控制 17](#_Toc450252169)

[8.2.通信与网络 20](#_Toc450252170)

[9.测试部分 21](#_Toc450252171)

[9.1.测试方针 21](#_Toc450252172)

[9.2.集成策略 21](#_Toc450252173)

[9.3.特殊考虑 22](#_Toc450252174)

[10.附录 23](#_Toc450252175)

[10.1.项目名称 23](#_Toc450252176)

[10.2.项目logo 23](#_Toc450252177)

[10.3.项目进度时间表 23](#_Toc450252178)

1.工作范围

1.1.系统目标

我们这个系统旨在开发一款安卓应用“BANG”，主要功能包括：资料库、校友圈和发布。

这款app主要对接即将毕业的校内学生对学习资料、职业或行业信息、求职或保研或留学资料的需求和已经毕业的学长学姐们拥有的对应资料。我们希望通过这款软件，达到以下目标：

（1）建立更为平面化的校友沟通平台，降低获取校友资源成本；

（2）建立校内资源共享平台，减少重复获取资源的成本，让资源更好地传承以方便更多人的学习和工作；

（3）通过经验分享，将已被验证的知识或秘诀帮助到更多的人；

（4）聚合信息的同时，对信息进行整合分类，提高用户浏览的信息有效性。

1.2.运行环境

1.2.1.移动端

我们开发的软件针对移动端安卓（市场占有率 70%）。移动OS的有效适配能够保证硬件发挥其性能的最佳优势。

Android专门针对ARM进行了很多的优化适配工作，同时，根据 ARM核心技术的创新演进，积极跟进 Android操作系统层面的支持，如支持多核、增强多媒体性能及硬件安全技术等，保证硬件性能的最佳体现。

Android的系统架构和其操作系统一样，采用了分层的架构。



**图 1** Android架构

从架构图看，Android分为四个层，从高层到低层分别是应用程序层、应用程序框架层、系统运行库层和Linux内核层。

（1）应用程序

Android会同一系列核心应用程序包一起发布，该应用程序包包括客户端，SMS短消息程序，日历，地图，浏览器，联系人管理程序等。所有的应用程序都是使用JAVA语言编写的。

（2）应用程序框架

开发人员也可以完全访问核心应用程序所使用的API框架。该应用程序的架构设计简化了组件的重用；任何一个应用程序都可以发布它的功能块并且任何其它的应用程序都可以使用其所发布的功能块（不过得遵循框架的安全性）。同样，该应用程序重用机制也使用户可以方便的替换程序组件。隐藏在每个应用后面的是一系列的服务和系统, 其中包括：丰富而又可扩展的视图（Views)，可以用来构建应用程序， 它包括列表（Lists)，网格（Grids)，文本框（Text boxes)，按钮（Buttons)， 甚至可嵌入的web浏览器。内容提供器（Content Providers)使得应用程序可以访问另一个应用程序的数据（如联系人数据库)， 或者共享它们自己的数据资源管理器（Resource Manager)提供 非代码资源的访问，如本地字符串，图形，和布局文件（Layout files )。通知管理器 （Notification Manager) 使得应用程序可以在状态栏中显示自定义的提示信息。活动管理器（ Activity Manager) 用来管理应用程序生命周期并提供常用的导航回退功能。

（3）系统运行库

Android 包含一些C/C++库，这些库能被Android系统中不同的组件使用。它们通过 Android 应用程序框架为开发者提供服务。以下是一些核心库：

\* 系统 C 库 - 一个从BSD继承来的标准C系统函数库 Libc )， 它是专门为基于Embedded linux的设备定制的。

\* 媒体库 - 基于PacketVideo OpenCORE;该库支持多种常用的音频、视频格式回放和录制，同时支持静态图像文件。编码格式包括MPEG4, H.264, MP3, AAC, AMR, JPG, PNG 。

\* Surface Manager - 对显示子系统的管理，并且为多个应用程序提 供了2D和3D图层的无缝融合。

\* LibWebCore - 一个最新的web浏览器引擎用，支持Android浏览器和一个可嵌入的web视图。

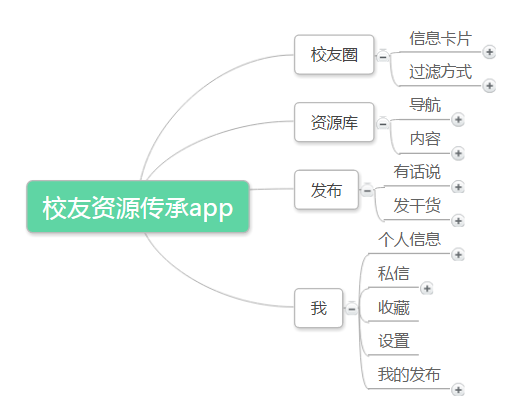
1.2.2.Web端（部分）

支持IE8以上版本及主流浏览器。

1.3.主要软件需求

1.3.1.校友圈

校友圈一方面是师兄师姐们可以向全平台发布自己的经验、经历记录和生活展示等，可以帮助未来毕业意向相近的师弟师妹们学习经验并对该方向有更深入的了解；另一方面，师弟师妹们可以发布自己的疑问和需求，可能有机会得到平台有能力的师兄师姐热心帮助。为了提高消息的针对性，对于方向较为明确的同学，我们考虑对于发布的消息进行过滤，只推送给这类同学对应方向的相关新消息。



**图 2** 信息架构图

1.3.2.资源库

资源库主要是希望提供一个资源共享平台，方便师兄师姐们分享自己手中可以传承的有助于师弟师妹学习的资料，可以减少师弟师妹花费大量时间去做重复性的收集资料的工作，此外，走出校园前也可能借助平台资源的力量得到更多热心师兄师姐的帮助，这对于即将毕业的大学生而言将是非常宝贵的人脉资源。

1.3.3.发布

发布的功能主要是作为资源库和校友圈的消息来源。分为两个模块，“有话说”对应的是发布到校友圈，而“发干货”对应的是发布到“资源库”。

1.3.4.我

我的页面除了个人信息，还有私信、收藏、我的发布和设置四个模块。

2.体系结构设计

2.1.数据流与控制流复审

2.1.1.数据流

本例采用分层方法制作数据流图，依次画出系统顶层图，系统内部，加工内部。

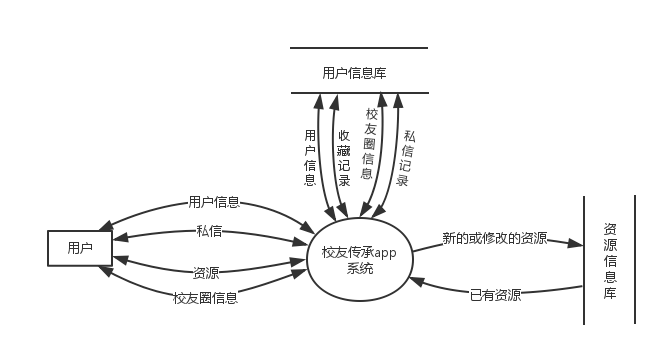
2.1.1.1.系统顶层图

a.确定信息源：包括用户、用户信息库和资源信息库，它们都向系统提供信息（上传资源）又向系统输出信息（下载资源），所以既是又是源又是宿

b.确定加工：顶层图只有一个加工，即校友资源传承APP系统

c.确定数据流：顶层图中的数据流是系统的输入输出信息，分析系统的功能说明，可以确定输入信息流为包括用户信息、校友圈、私信和资源，输出信息流包括用户信息、收藏记录、校友圈信息、私信记录、资源库已有资源等

d.顶层图



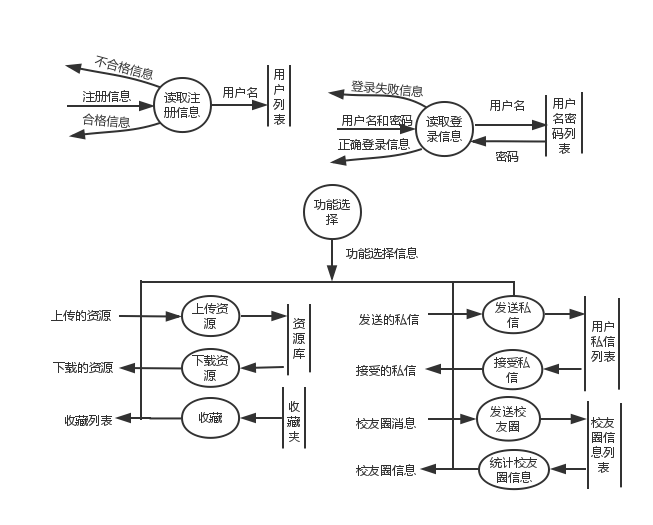
**图 3** 顶层图

2.1.1.2.系统内部

a.确定加工： 根据功能分解，校友传承app系统包括注册、登录、上传资源、下载资源、收藏资源、发送私信 、接受私信、发送校友圈、接受校友圈等子加工

b.确定数据流：用户把若干数据作为一个单位来处理时，将这些数据看作一个数据流。比如，本例中，注册加工从用户读入注册信息，向用户列表查询用户名是否已存在，如果注册信息不合格则输出不合格信息，如果合格则输出合格信息；登录加工从用户读取用户名和密码，向数据库查询用户名和密码，如果登录信息不合格则输出登录信息不合格信息，如果成功则进入主功能界面；功能选择加工从用户读入功能选择信息，对不同的选择进如不同子加工；上传资源加工从用户读入要上传的资源，向资源库写入资源；下载资源加工从用户读入要下载的资源，从资源库中读取资源；收藏加工从收藏夹读入收藏信息，向用户输出收藏列表；其他子加工类似。

c.确定文件： 在父图中某加工分解的子图中，如果需要保存某些中间数据，以备以后使用则可以将数据组成一个新的文件。登录和注册需要从用户数据库中读取用户名列表和用户名密码列表；上传资源和下载资源需要读取资源库列表；收藏加工需要读取收藏夹；发送和接受私信需要私信列表；发送和显示校友圈需要校友圈信息列表。

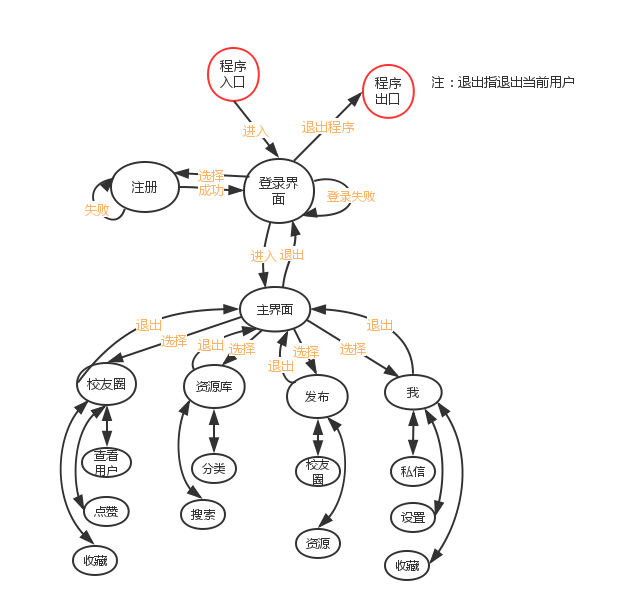
d.0层信息流图

**图 4** 0层信息流图

2.1.2.控制流

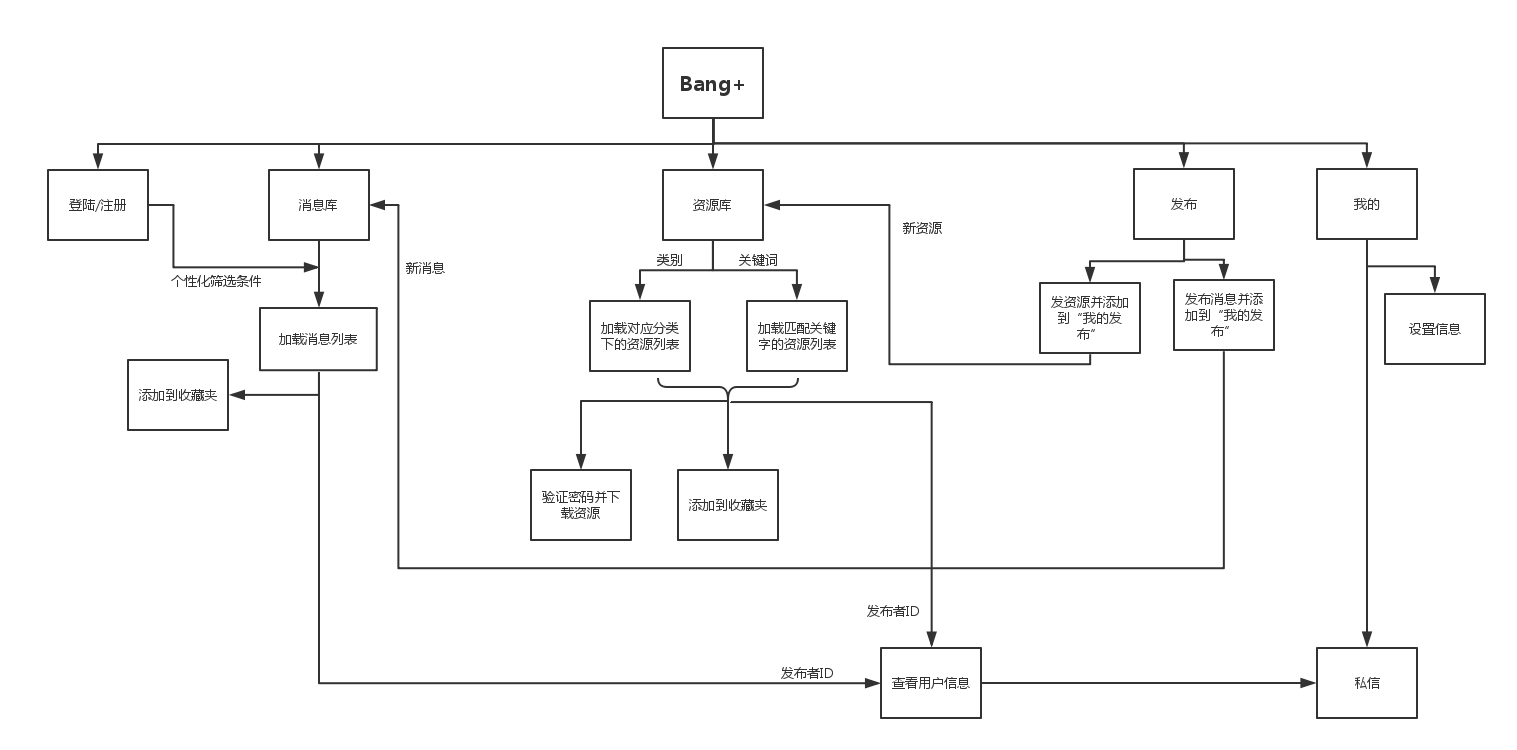
控制图用于描述程序运行所有可能经过的路径。控制流图用有向图表示，其中每个节点表示基本快，每条边代表一条控制流路径，有向边从开始块指向结束块，直线的代码段没有任何跳转。控制流图包含入度为0和出度为0的块，代表程序从入口开始执行，到达出口则终止运行。

本例中，程序首先进入界面，登陆界面块可以走三条路径，注册、登录和退出；注册块中如果失败则停留在注册界面，成功则跳转进入登录界面；登录成功后进入主界面模块，主界面中有功能选择模块，程序根据用户的功能选择信息进入相应功能模块，包括校友圈、资料库、发布、我四大模块。校友圈模块显示校友圈信息，本模块可以查看发布信息的用户，点赞和收藏，在进行相应操作后又会回到校友圈模块；资源库模块有搜索和分类模块，在设计报告中会具体设计控制流；发布模块包括校友圈信息发布和资源信息发布模块；‘我’模块包括私信模块和用户信息模块。为用户方便退出程序，本例可以在任何模块中退出并进入登录界面，然后退出程序。



**图 5** 控制流

2.2.导出的程序结构



**图 6** 程序结构

3.数据库设计

3.1.标识符和状态

数据库软件的名称：MySQL

数据库的名称：bang

3.2.命名约定

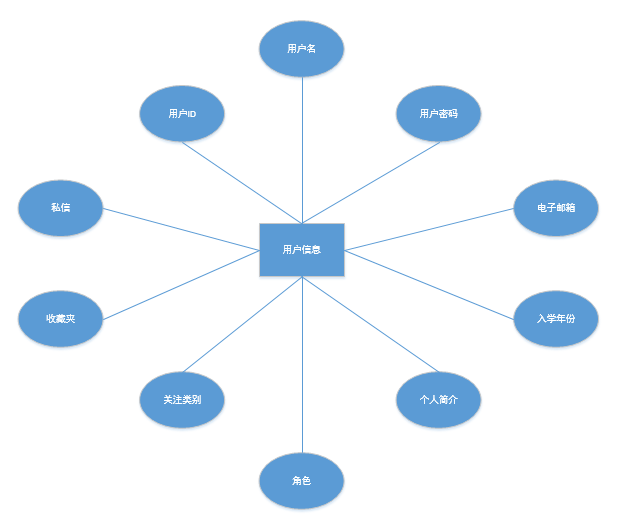
所有由多个单词组成的表名和字段名，单词之间由下划线分隔，这样能够统一数据库表和字段的命名，也能够更好地规范数据库表和字段的命名。与形成的数据结构

3.3.数据对象与形成的数据结构

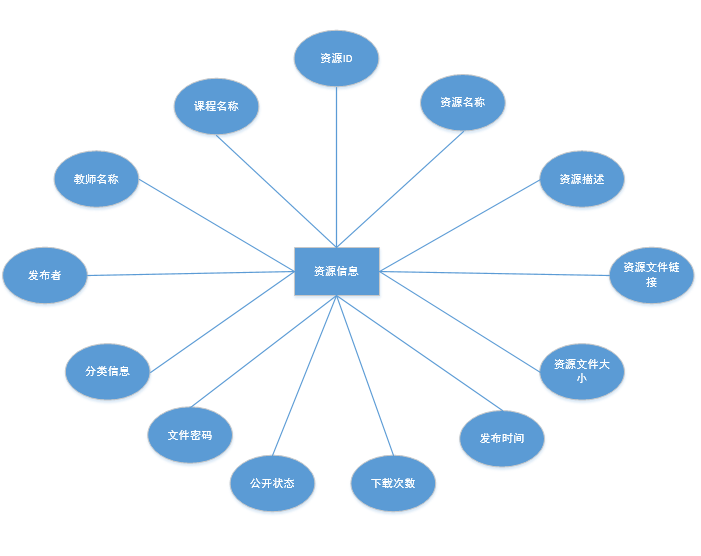
3.3.1.实体和属性的定义

①用户信息（用户ID，用户名，用户密码，电子邮箱，入学年份，个人简介，角色，关注类别，收藏夹，私信）

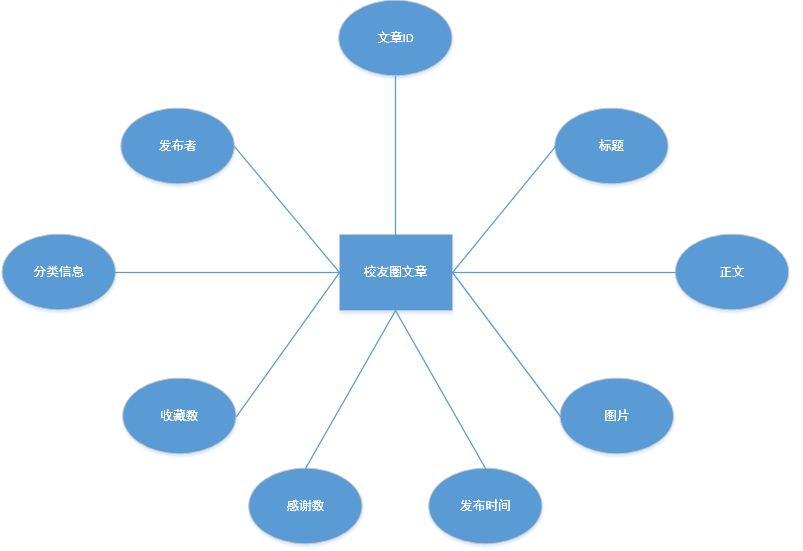
属性约束：

角色：1 = 师兄师姐， 0 = 师弟师妹；关注类别：null = 随便。

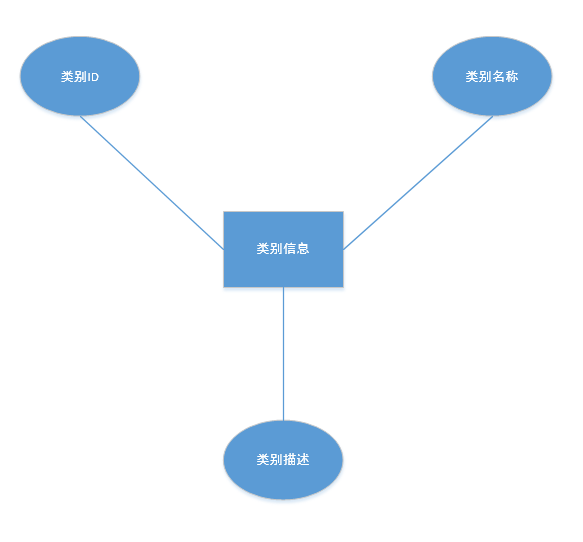
**图 7** 用户信息数据结构

②资源信息（资源ID，资源名称，资源描述，资源文件链接，资源文件大小，发布时间，下载次数，公开状态，文件密码，分类信息，发布者，教师名称，课程名称）属性约束：公开状态：0 = 加密， 1 = 公开；

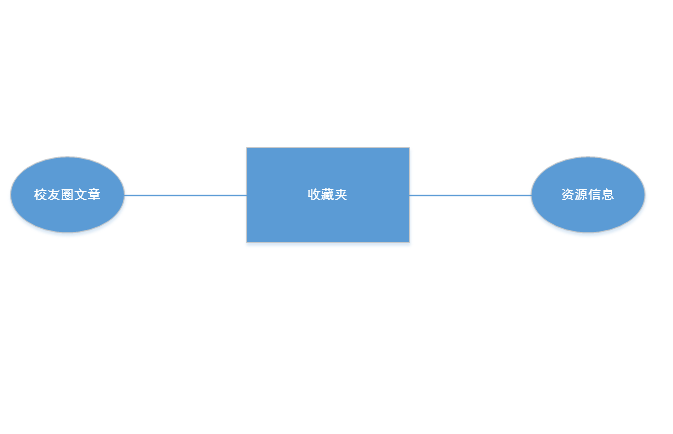
**图 8** 资源信息数据结构

③校友圈文章（文章ID，标题，正文，图片，发布时间，感谢数，收藏数，分类信息，发布者）

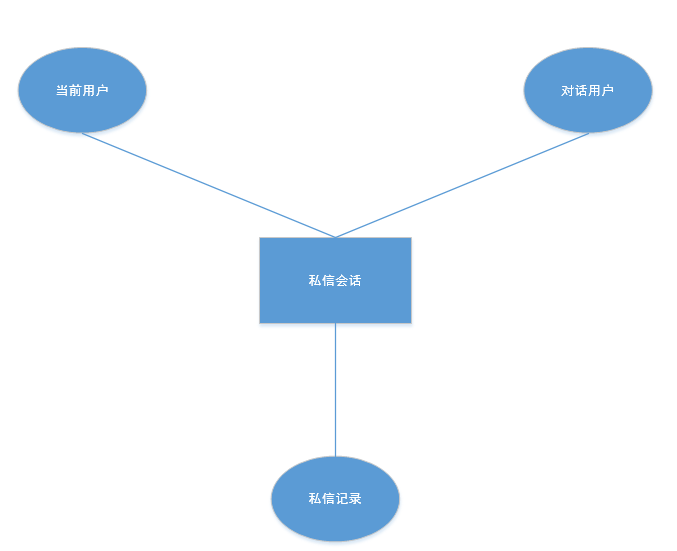
**图 9** 校友圈数据结构

④类别信息（类别ID，类别名称，类别描述）

**图 10** 类别信息数据结构

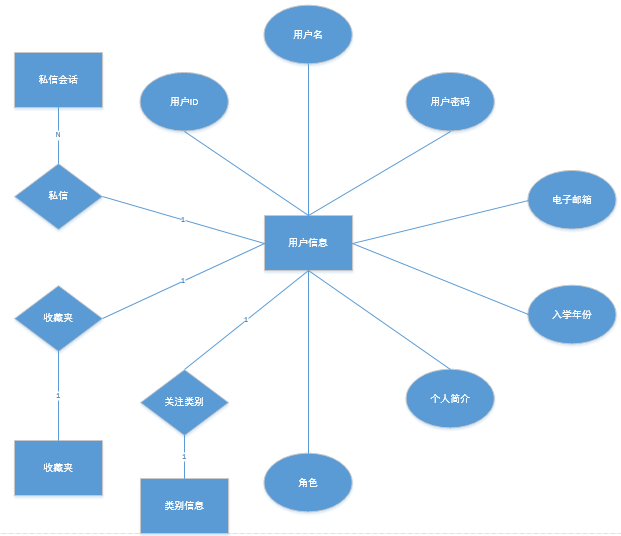
⑤收藏夹（校友圈文章，资源信息）

**图 11** 收藏夹数据结构

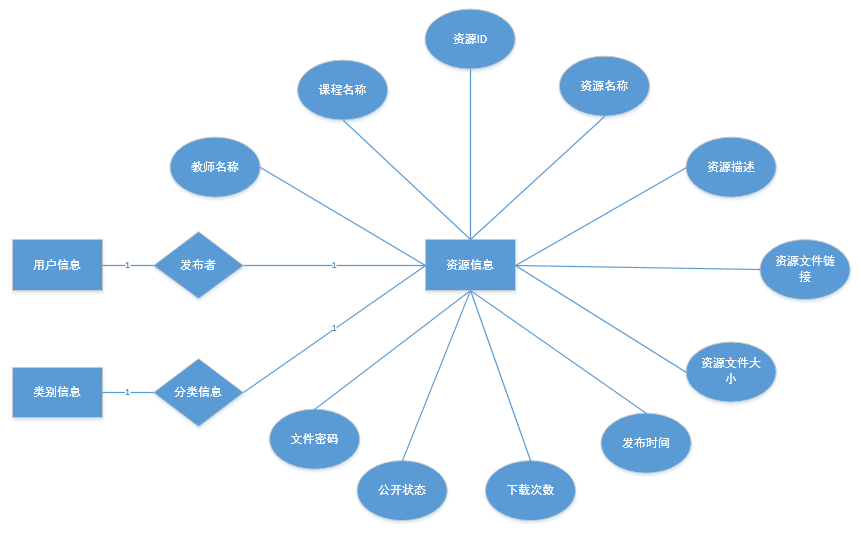
⑥私信会话（当前用户，对话用户，私信记录）

**图 12** 私信会话数据结构

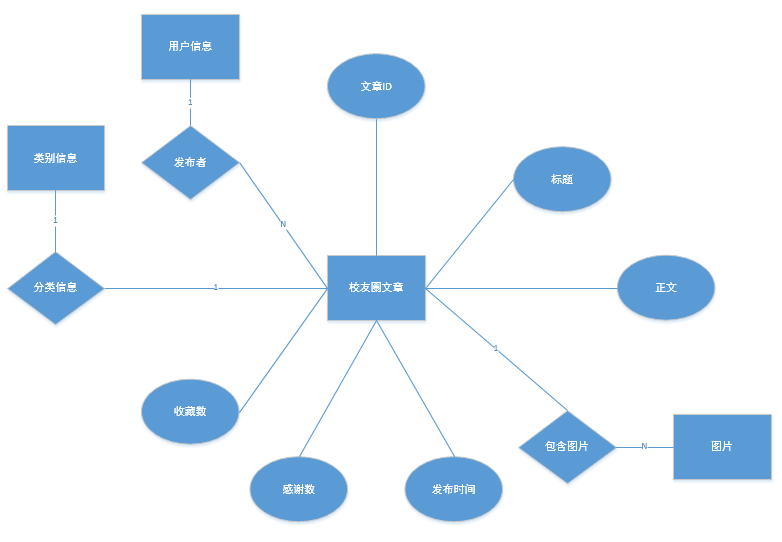
3.3.2.E-R模式设计

①用户信息局部E-R图：

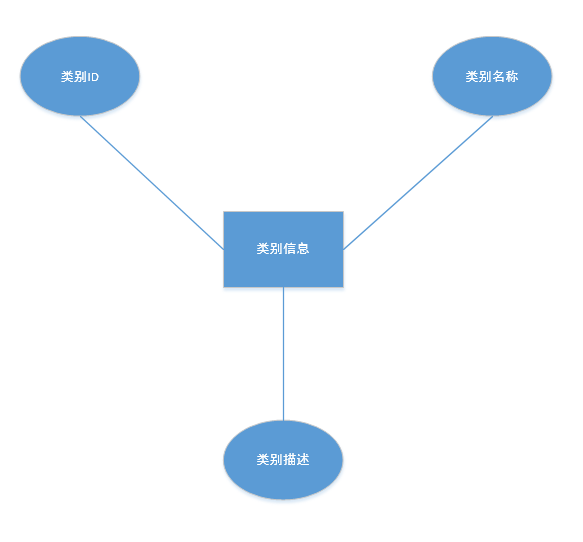
**图 13** 用户信息局部E-R图

②资源信息局部E-R图：

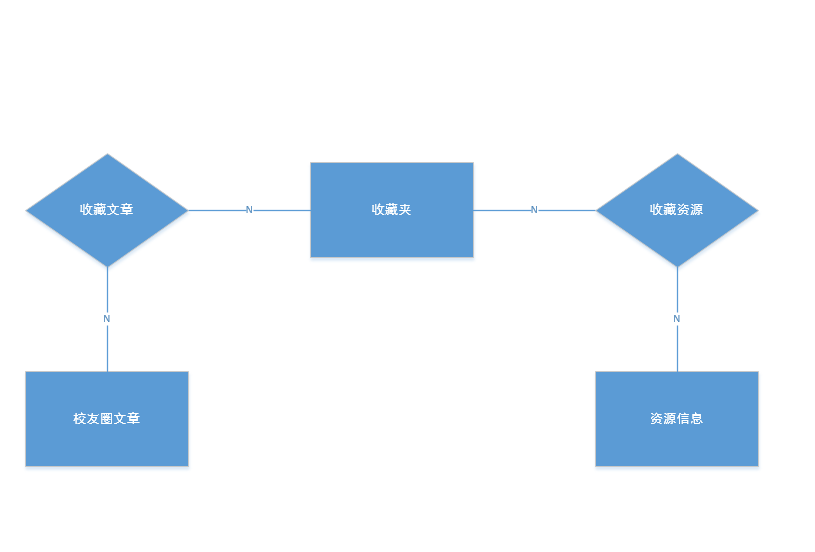
**图 14** 资源信息局部E-R图

③校友圈文章局部E-R图：

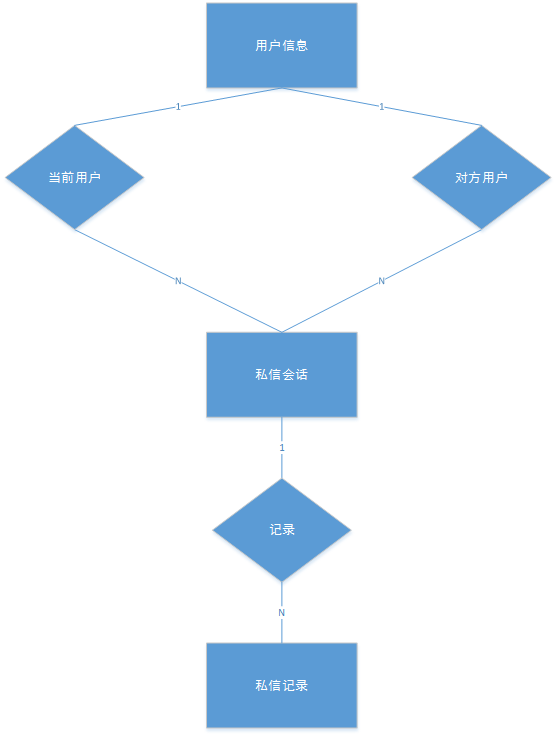
**图 15** 资源信息局部E-R图

④类别信息局部E-R图：

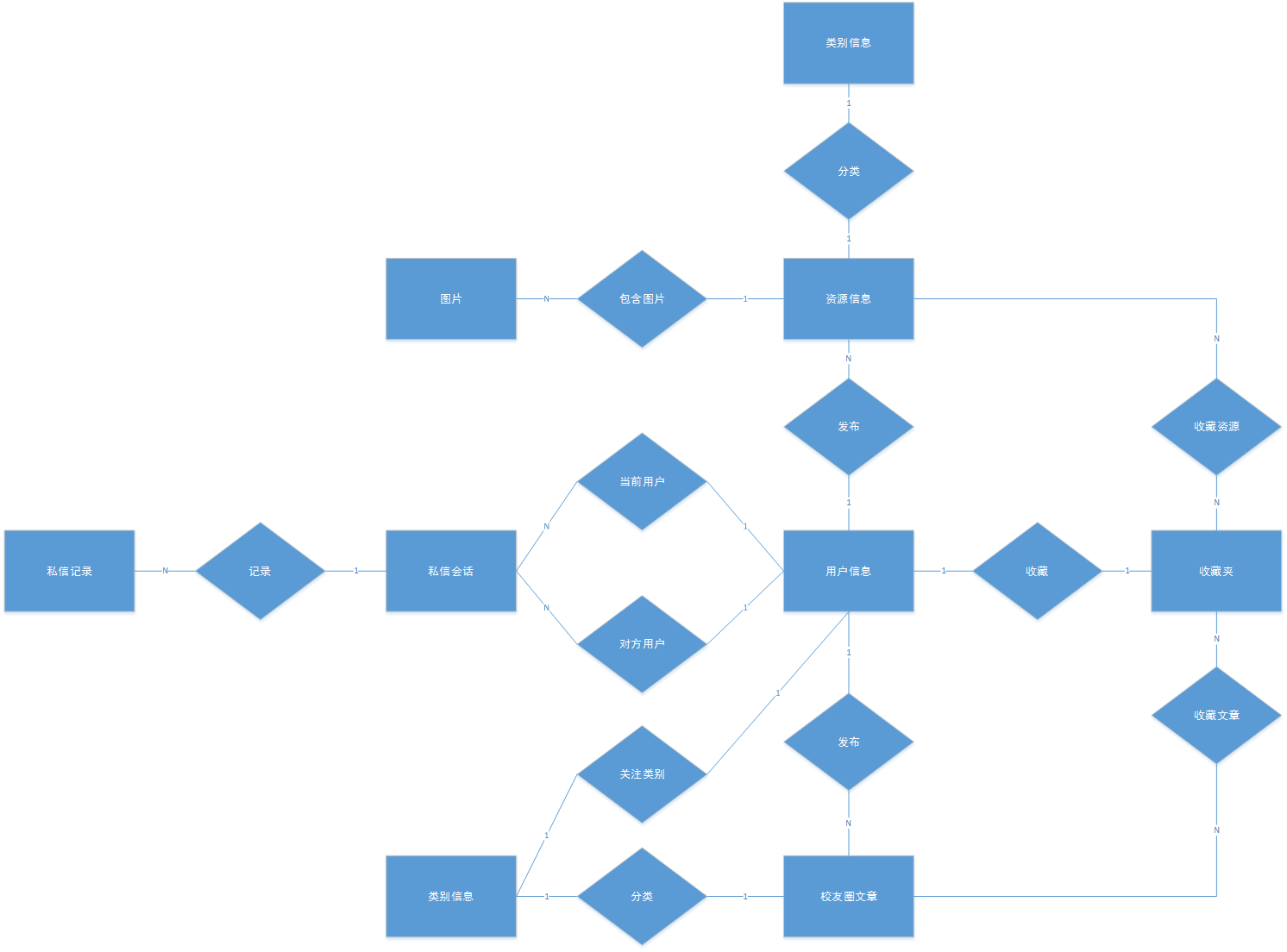
**图 16** 类别信息局部E-R图

⑤收藏夹局部E-R图：

**图 17** 收藏夹局部E-R图

⑥私信会话局部E-R图：

**图 18** 私信会话局部E-R图

⑦全局E-R图：

**图 19** 全局E-R图

3.4.文件和数据库结构

①用户信息：包括唯一的ID，昵称，用户密码，介绍，收藏夹，发布的消息（包括校友圈消息和资源库消息），内容筛选设置信息，私信；

②校友圈消息：数据包括标题，正文，图片，感谢数，收藏数，发布者，消息分类；

③资源：包括资源名称、资源文件链接、资源文件大小、发布者、下载次数、公开状态、文件密码、分类信息、教师名称、课程名称、资源唯一ID；

④收藏夹：包括了用户收藏的校友圈信息和资源库信息；

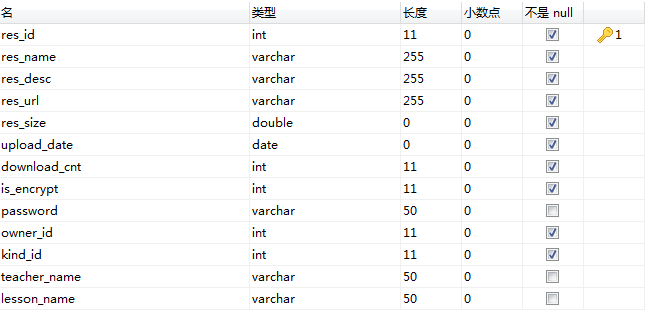
⑤内容筛选设置信息：包括三个数据，数据1表示师兄师姐或师弟师妹，数据2表示工作或读研或留学或学习或随便，数据3表示入学年份；

⑥私信：包括当前用户，对话用户和私信记录；

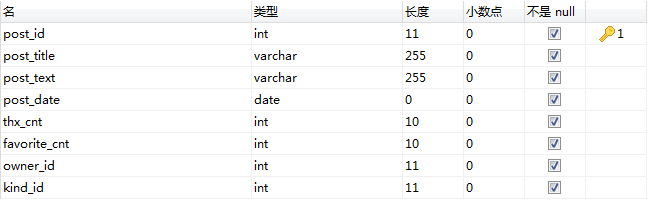
⑦消息分类：一个数据（表示工作或读研或留学或学习）；

用户信息表储存用户信息。表user的结构：

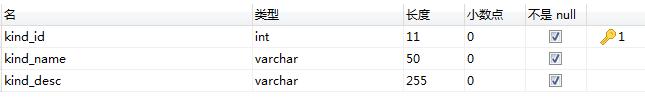
**图 20** 用户信息表结构

资源信息表储存资源的信息。表resource的结构：

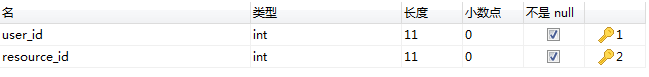
**图 21** 资源表结构

校友圈文章信息表存储在校友圈上发布的文章信息。表post的结构：

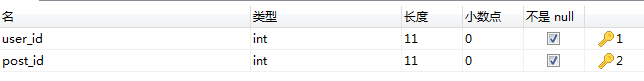
**图 22** 校友圈表结构

类别信息表存储分类的名称和描述。表kind的结构：

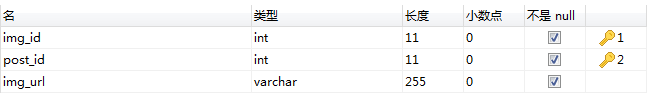
**图 23** 类别信息表结构

收藏夹-资源表存储用户id和资源id的对应关系。表favorites\_resource的结构：

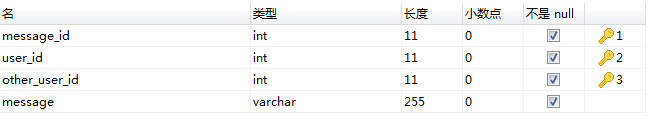
**图 24** 用户和资源关系表结构

收藏夹-文章表存储用户id和文章id的对应关系。表favorites\_post的结构：

**图 25** 用户和文章表结构

图片表存储文章id和图片url的对应关系。表image的结构：

**图 26** 文章和图片表结构

消息表存储当前用户id，对话用户id和私信内容的对应关系。表message的结构：

**图 27** 用户和私信表结构

3.5.数据库实施

SET FOREIGN\_KEY\_CHECKS=0;

**-- ----------------------------**

**-- Table structure for favorites\_post**

**-- ----------------------------**

DROP TABLE IF EXISTS `favorites\_post`;

CREATE TABLE `**favorites\_post**` (

`user\_id` int(11) NOT NULL COMMENT '0=文章，1=资源',

`post\_id` int(11) NOT NULL,

PRIMARY KEY (`user\_id`,`post\_id`),

KEY `post\_id` (`post\_id`),

CONSTRAINT `favorites\_post\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`),

CONSTRAINT `favorites\_post\_ibfk\_2` FOREIGN KEY (`post\_id`) REFERENCES `post` (`post\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

**-- ----------------------------**

**-- Table structure for favorites\_resource**

**-- ----------------------------**

DROP TABLE IF EXISTS `favorites\_resource`;

CREATE TABLE `**favorites\_resource**` (

`user\_id` int(11) NOT NULL,

`resource\_id` int(11) NOT NULL,

PRIMARY KEY (`user\_id`,`resource\_id`),

KEY `resource\_id` (`resource\_id`),

CONSTRAINT `favorites\_resource\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`),

CONSTRAINT `favorites\_resource\_ibfk\_2` FOREIGN KEY (`resource\_id`) REFERENCES `resource` (`res\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

**-- ----------------------------**

**-- Table structure for image**

**-- ----------------------------**

DROP TABLE IF EXISTS `image`;

CREATE TABLE `**image**` (

`img\_id` int(11) NOT NULL,

`post\_id` int(11) NOT NULL,

`img\_url` varchar(255) NOT NULL,

PRIMARY KEY (`img\_id`,`post\_id`),

KEY `post\_id` (`post\_id`),

CONSTRAINT `image\_ibfk\_1` FOREIGN KEY (`post\_id`) REFERENCES `post` (`post\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

**-- ----------------------------**

**-- Table structure for kind**

**-- ----------------------------**

DROP TABLE IF EXISTS `kind`;

CREATE TABLE `**kind**` (

`kind\_id` int(11) NOT NULL AUTO\_INCREMENT,

`kind\_name` varchar(50) NOT NULL,

`kind\_desc` varchar(255) NOT NULL,

PRIMARY KEY (`kind\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=utf8;

**-- ----------------------------**

**-- Table structure for message**

**-- ----------------------------**

DROP TABLE IF EXISTS `message`;

CREATE TABLE `**message**` (

`message\_id` int(11) NOT NULL,

`user\_id` int(11) NOT NULL,

`other\_user\_id` int(11) NOT NULL,

`message` varchar(255) NOT NULL,

PRIMARY KEY (`message\_id`,`user\_id`,`other\_user\_id`),

KEY `user\_id` (`user\_id`),

KEY `other\_user\_id` (`other\_user\_id`),

CONSTRAINT `message\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`),

CONSTRAINT `message\_ibfk\_2` FOREIGN KEY (`other\_user\_id`) REFERENCES `user` (`user\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

**-- ----------------------------**

**-- Table structure for post**

**-- ----------------------------**

DROP TABLE IF EXISTS `post`;

CREATE TABLE `**post**` (

`post\_id` int(11) NOT NULL AUTO\_INCREMENT,

`post\_title` varchar(255) NOT NULL,

`post\_text` varchar(255) NOT NULL,

`post\_date` date NOT NULL,

`thx\_cnt` int(10) unsigned NOT NULL DEFAULT '0',

`favorite\_cnt` int(10) unsigned NOT NULL DEFAULT '0',

`owner\_id` int(11) NOT NULL,

`kind\_id` int(11) NOT NULL,

PRIMARY KEY (`post\_id`),

KEY `owner\_id` (`owner\_id`),

KEY `kind\_id` (`kind\_id`),

CONSTRAINT `post\_ibfk\_1` FOREIGN KEY (`owner\_id`) REFERENCES `user` (`user\_id`),

CONSTRAINT `post\_ibfk\_2` FOREIGN KEY (`kind\_id`) REFERENCES `kind` (`kind\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

**-- ----------------------------**

**-- Table structure for resource**

**-- ----------------------------**

DROP TABLE IF EXISTS `resource`;

CREATE TABLE `**resource**` (

`res\_id` int(11) NOT NULL AUTO\_INCREMENT,

`res\_name` varchar(255) NOT NULL,

`res\_desc` varchar(255) NOT NULL,

`res\_url` varchar(255) NOT NULL,

`res\_size` double NOT NULL,

`upload\_date` date NOT NULL,

`download\_cnt` int(11) NOT NULL DEFAULT '0',

`is\_encrypt` int(11) NOT NULL DEFAULT '0',

`password` varchar(50) DEFAULT NULL,

`owner\_id` int(11) NOT NULL,

`kind\_id` int(11) NOT NULL,

`teacher\_name` varchar(50) DEFAULT NULL,

`lesson\_name` varchar(50) DEFAULT NULL,

PRIMARY KEY (`res\_id`),

KEY `owner\_id` (`owner\_id`),

KEY `kind\_id` (`kind\_id`),

CONSTRAINT `resource\_ibfk\_1` FOREIGN KEY (`owner\_id`) REFERENCES `user` (`user\_id`),

CONSTRAINT `resource\_ibfk\_2` FOREIGN KEY (`kind\_id`) REFERENCES `kind` (`kind\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8;

**-- ----------------------------**

**-- Table structure for user**

**-- ----------------------------**

DROP TABLE IF EXISTS `user`;

CREATE TABLE `**user**` (

`user\_id` int(11) NOT NULL AUTO\_INCREMENT,

`username` varchar(50) NOT NULL,

`userpass` varchar(50) NOT NULL,

`email` varchar(100) NOT NULL,

`school\_year` year(4) NOT NULL,

`user\_desc` varchar(50) DEFAULT NULL,

`role` int(11) DEFAULT NULL,

`concern\_kind\_id` int(11) DEFAULT NULL,

PRIMARY KEY (`user\_id`),

UNIQUE KEY `username` (`username`),

KEY `concern\_kind\_id` (`concern\_kind\_id`),

CONSTRAINT `user\_ibfk\_1` FOREIGN KEY (`concern\_kind\_id`) REFERENCES `kind` (`kind\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8;

4.接口设计

4.1.人机界面设计规则

4.1.1 一般可交互性

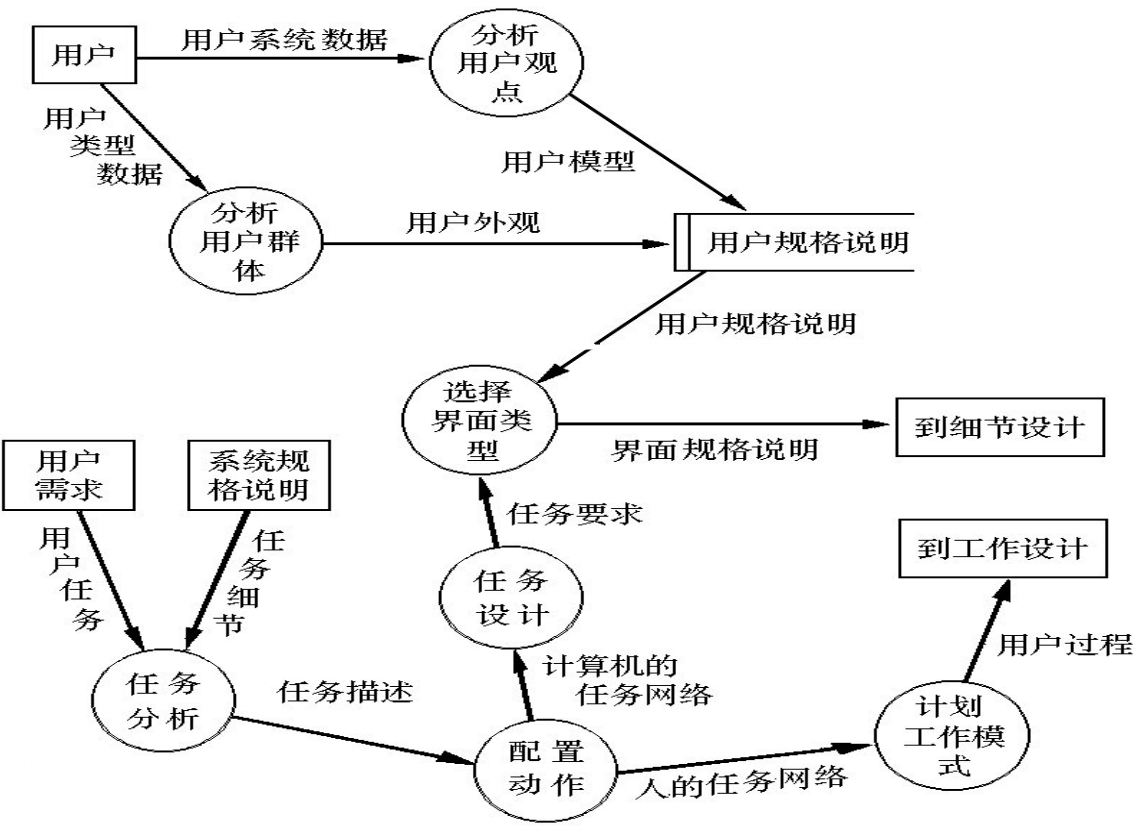
* 在同一界面中应保持同一种风格
* 根据用户的感知器官，保持用户与界面双向通信
* 对可能的损害操作，坚持要用户确认
* 对多数操作，可提供操作后允许恢复
* 尽量减少用户的记忆负担
* 提高对话、移动和思考的效率
* 用户出错采取宽容的态度
* 按功能分类组织界面上的活动
* 提供上下文敏感的求助系统
* 用简短的动词和动词短语提示命令

4.1.2 信息显示

* 仅显示与当前上下文有关的信息
* 避免因数据过于费解造成用户烦恼
* 采用统一的标号、约定俗成的缩写和预先定义好的颜色
* 允许用户对可视环境进行维护
* 只显示有意义的出错信息
* 用大、小写，缩进和按意群分组等方法提高可理解性
* 用窗体分割不同种类的信息
* 用类比方法，生动形象地表示信息
* 合理划分并高效使用显示屏

4.1.3 数据输入

* 尽量减少用户输入的动作
* 保证信息显示方式与数据输入方式协调一致
* 允许用户定制输入格式
* 采用灵活多样的交互方式，允许用户自选输入方式
* 隐藏当前状态下不可用的命令
* 允许用户控制交互过程
* 为所有输入动作提供帮助信息
* 删除所有无实现意义的输入



4.2.人机界面设计过程

1）创建系统功能的外部模型

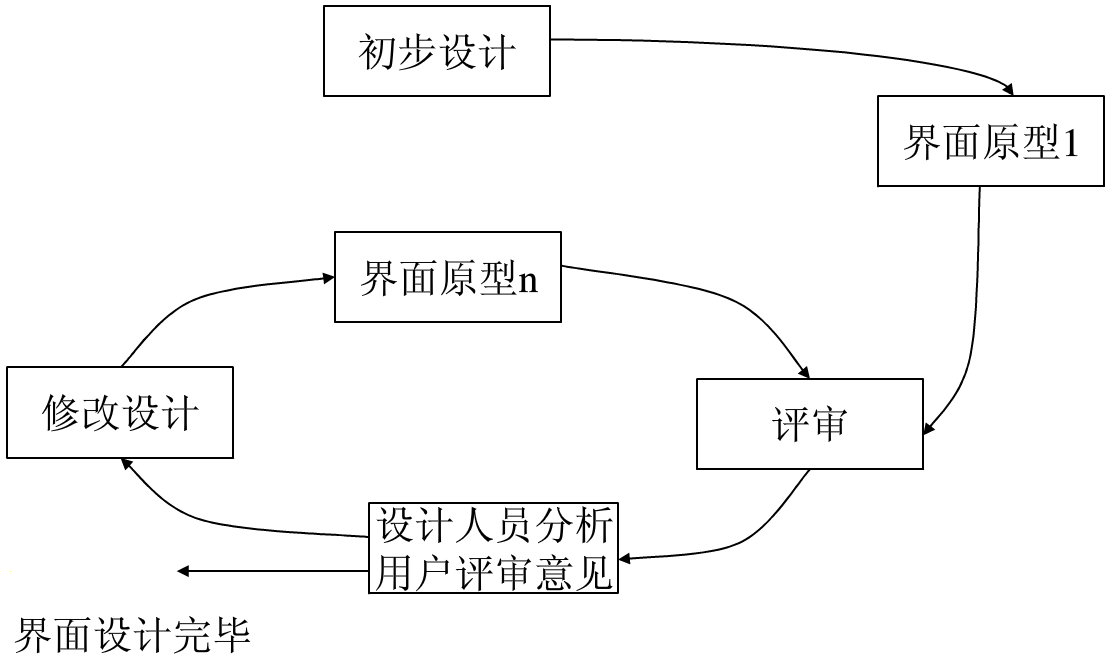
2）确定为完此系统功能，人和计算机应分别完成的任务

3）考虑界面设计中的典型问题

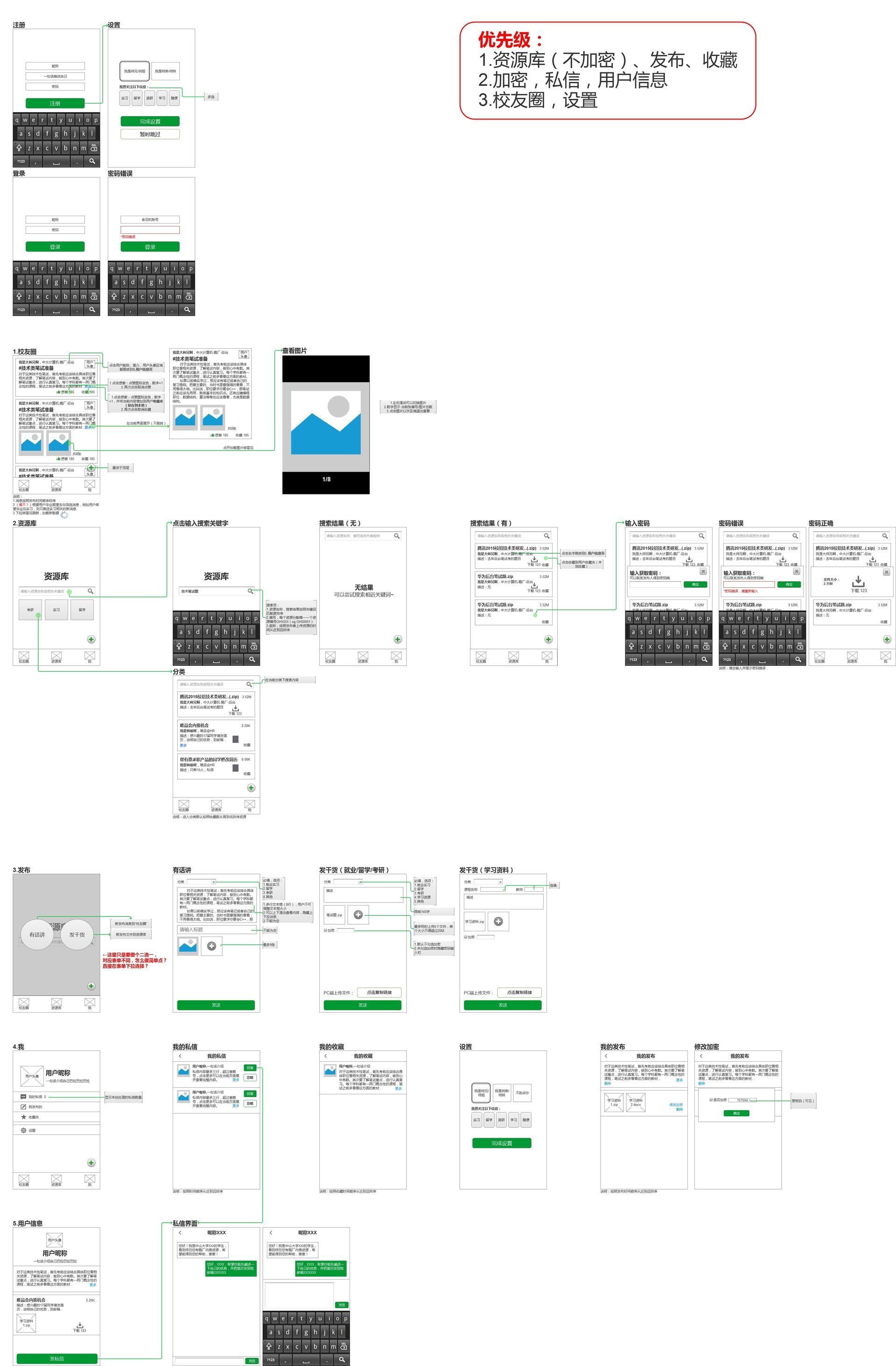
4）借助CASE工具构造界面原型

5）实现设计模型

6）评估界面质量



Bang完整原型界面：



4.3.外部接口设计

说明本系统同外界的所有接口的安排，包括软件与硬件之间的接口，本系统与各支持软件之间的接口关系。 内部和外部的接口设计必须结合模块内部的数据验证和出错处理。

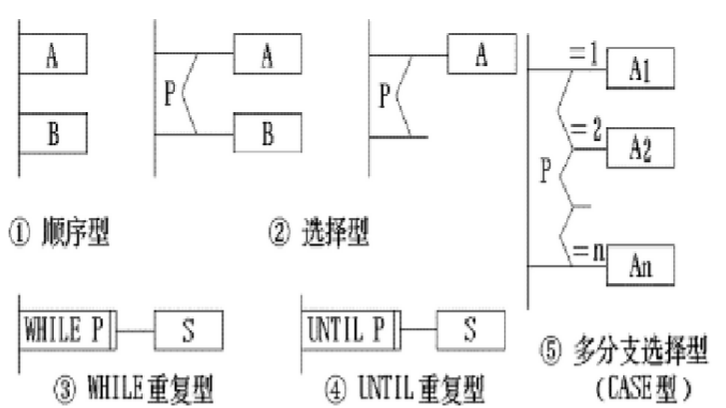
4.4.内部接口设计

说明软件内部模块间的接口安排。

5.各部件的过程设计

5.1.部件设计与算法描述

在本文档中统一使用PAD图，即树形问题分析图（problem analysis diagram）。



（1）启动模块

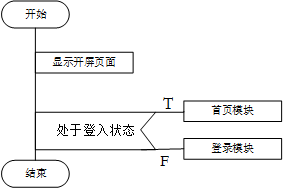
a.处理

功能：1.显示app信息 2.提醒用户更新

输入：1.是否选择更新 2.网络状态 3.登录状态

输出：1.展示首屏图片 2.加载首屏文字（slogan和版本号） 3.跳转至更新 4.网络状态差时，提示网络状态 5.跳转页面

b.算法流程



（2）登录模块

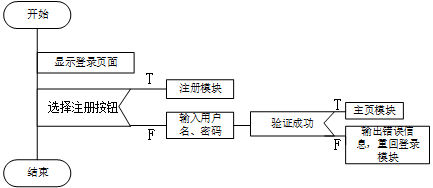
a.处理

功能：用户登录

输入：账号、密码

输出：1.提示信息：账号密码是否匹配；2.跳转页面：主页

b.算法流程



（3）注册模块

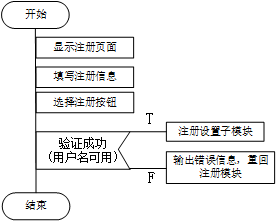
a.处理

功能：用户注册。

输入：注册信息如用户名、密码、描述等。

输出：注册验证成功跳转注册设置子模块，失败返回注册模块。

b.算法流程



（4）注册设置子模块

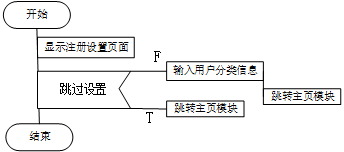
a.处理

功能：对用户进行分类。

输入：类别选择信息。

输出：完成设置跳转主页模块。

b.算法流程



（5）主页模块

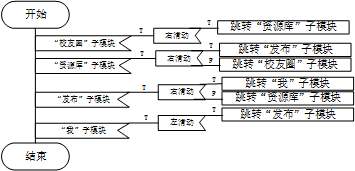
a.处理

功能：显示系统主界面，由“校友圈”、“资源库”、“发布”、“我”四个Tab对应的子模块构成。默认初始显示“校友圈”子模块。

输入：屏幕滑动。

输出：相应子模块跳转。

b.算法流程



（6）主页“校友圈”子模块

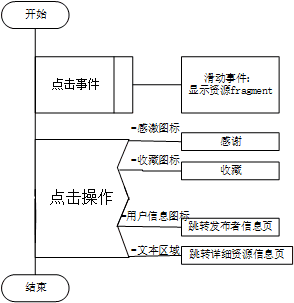
a.处理

功能：显示发布的资源消息，以fragment呈现。

输入：点击感谢、收藏、查看更多。点击查看发布者。

输出：已感谢、已收藏，跳转资源详细页，跳转发布者信息详细页。

b.算法流程



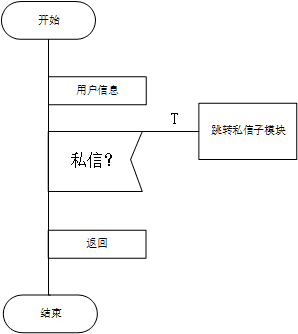
（7）“校友圈”用户信息子模块

a.处理

功能：显示资源发布者基本信息，提供私信功能。

输入：点击发布用户头像。

输出：发布者基本信息。

b.算法流程

（8）主页“资源库”子模块

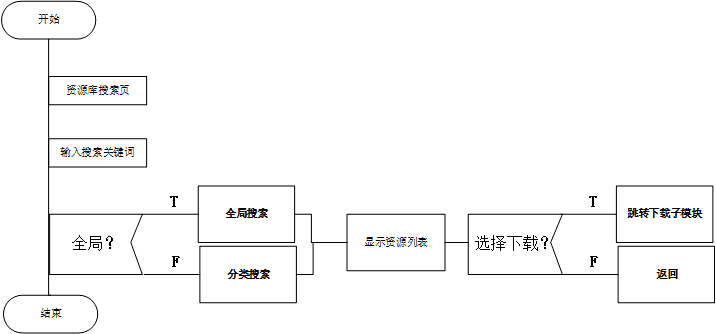
a.处理

功能：通过筛选提供相应资源。

输入：搜索类别，搜索关键词。

输出：资源列表。

b.算法流程



（9）“资源库”下载子模块

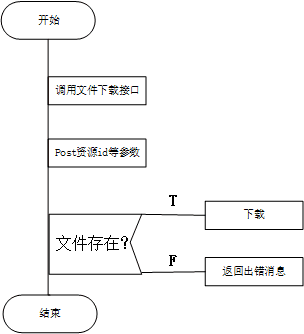
a.处理

功能：从服务器下载相应资源至手机本地。

输入：资源id。

输出：返回资源文件。

b.算法流程



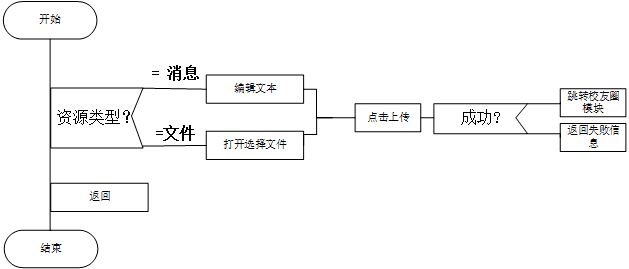
（10）主页“发布”子模块

a.处理

功能：发布消息、文件。

输入：资源类型，资源文件，加密信息。

输出：上传结果消息。

b.算法流程

（11）主页“我”子模块

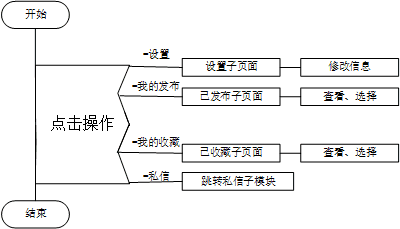
a.处理

功能：显示个人基本信息，显示私信消息。

输入：点击设置；点击私信；点击我的发布、收藏。

输出：个人信息修改页；跳转私信页面；发布、收藏页面。

b.算法流程



（12）私信子模块

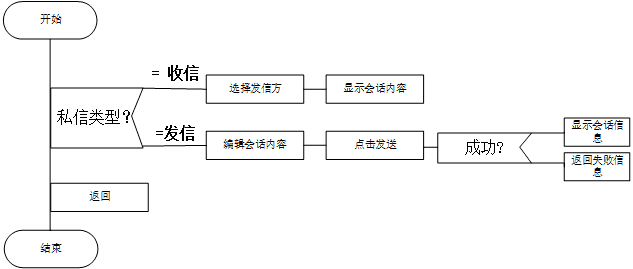
a.处理

功能：实现不同系统用户的沟通交流。

输入：发送会话内容。

输出：接收会话内容。

b.算法流程



5.2.设计语言描述

Android操作系统的Linux内核用汇编和C语言编写、Android的外部模块用C++语言编写、Android的图形界面（UI）则采用Java语言开发。Android的跨平台应用程序的开发一般采用Java语言（使用JDK和Android SDK），本地应用的开发采用C/C++语言和Android NDK（Native Development Kit，本机开发工具包），对安装了.NET跨平台运行环境（如Mono或Xamarin.Android）的设备，则可以采用C# 语言和Xamarin.Form及.NET框架类库。传统的Android应用程序开发，采用的IDE为安装了ADT（Android Development Tools，安卓开发工具）插件的（开源的基于Java的可扩展）多语言开发平台Eclipse（日食/月食）、使用Java语言及其开发工具包JDK、还有Google的Android SDK。

6.运行设计

6.1.运行部件组合

登陆注册功能：运行系统用户管理模块。

资源检索功能：系统界面模块，资源检索模块。

资源浏览功能：系统界面模块，资源浏览模块，文件下载模块。

发布功能：系统界面模块，信息发布模块，文件上传模块。

私信功能：系统界面模块，即时通信模块。

收藏夹功能：用户管理模块，资源浏览模块。

6.2.运行控制

注册：验证注册信息。

用户进行个性化配置。

保存配置并自动登陆。

登陆：输入用户名，密码进行验证。

查看校友圈：显示校友圈文章的内容。

资源检索： 输入需要搜索的资源名称关键字。

搜索用户指定的资源。 返回搜索结果。

资源下载：选择需要下载的资源，检查权限，如果资源已加密，输入密码进行验证。

执行下载操作。

显示下载速度及相关信息。

发布校友圈文章：填写表单并提交。

发布资源：填写表单，选择需要上传的资源并选择上传功能。

执行上传操作。

显示上传速度及相关信息。

私信：选择私信对象，编辑私信内容，发送，显示发送状态。

获取并查看私信回应。收藏：选择需要收藏的资源或文章并选择收藏功能。

保存该资源或文章到收藏夹。

6.3.运行时间

登陆注册功能：占用少量时间。

资源检索功能：占用大量时间进行查找。

资源浏览功能：占用少量时间获取相关资源信息。

发布功能：文件上传可能占用大量时间。

私信功能：占用部分时间。

收藏夹功能：占用少量时间。

7.出错处理设计

7.1.网络错误提示

（1）查看界面网络错误

提示用户当前网络较差，并提供刷新方式重试加载。

（2）在发送页面网络错误

提示用户网络问题，并提醒用户重新操作，提供用户“重试”按钮。

发送界面：提示发送失败，并可以重试提交发布；

个人信息设置界面：提示设置失败，并可以重试提交设置。



7.2.输入错误提示

（1）登录&注册

数据库没有该用户名：用户名无效

密码错误：密码错误

重复输入的密码与密码不一致：密码不一致，请重新输入

（2）发布

尚未填写必填项：信息未填写完整

（3）资源库

密码错误：密码错误

（4）私信

消息为空时点击“发送”：消息为空

（5）搜索

没有符合要求的搜索结果：无搜索结果

8.安全保密设计

8.1.访问控制

ISO所定义的5 大安全服务功能是：认证、访问控制、数据保密性、数据完整性和防止否认服务。其中访问控制服务在系统安全体系结构中起着不可替代的作用。 访问控制是通过某种途径显式地准许或限制访问能力及范围的一种方法。它是针对越权使用资源的防御措施，通过限制对关键资源的访问，防止非法用户的侵入或因为合法用户的不慎操作而造成的破坏．从而保证系统资源受控地、合法地被使用。用户只能在自己的权限内访问系统资源，不得越权访问。访问控制技术通常和身份认证密切联系，但并不能取代身份认证，它是建立在身份认证的基础之上的，通俗地说，身份认证解决的是‘你是谁，你是否真的具有你所声称的身份”，而访问控制技术解决的是‘你能做什么，你有什么样的权限”这个问题。 访问控制系统一般包括主体（subject）、客体（object）及安全访问政策几个实体。访问控制的目的是：限制主体对访问客体的访问权限，从而使计算机系统在合法范围内使用；决定用户能做什么，也决定代表一定用户利益的程序能做什么。

8.1.1.访问控制的实现方法

访问控制的常见实现方法有访问控制矩阵、访问能力表、访问控制表和授权关系表等几种。

1 .访问控制矩阵

从数学角度看，访问控制可以表示为一个矩阵的形式， 其中行表示客体（各种资源），列表示主体（通常为用户），行和列的交叉点表示某个主体对某个客体的访问权限（如读、写、执行、修改、删除等）。

2 .访问能力表

访问控制矩阵虽然直观，但并非每个主体和客体之间都存在着权限关系，相反，实际的系统中虽然可能有很多的主体和客体，但主体和客体之间的权限关系可能并不多，这样就存在着很多空白项。为了减轻系统的开销与浪费，我们从主体（行）出发，用访问能力表达矩阵某一行的信息；也可以从客体（列）出发，用访问控制表（Access Control List）表达矩阵某一列的信息。 能力（capacity）是受一定机制保护的客体标志，标记了客体以及主体（访问者）对客体的访问权限。只有当一个主体对某个客体拥有访问的能力时，它才能访问这个客体。在访问能力表中，由于它着眼于某一主体的访问权限，并以主体为出发点描述控制信息，所以很容易获得一个主体所被授权可以访问的客体及其权限，但如果要求获得对某一特定客体有特定权限的所有主体就比较困难。在一个安全系统中，正是客体本身需要得到可靠的保护，访问控制服务应该能够控制可访问某一客体的主体集合，并能够授予或取消主体的访问权限，于是出现了以客体为出发点的实现方式― ACL（访问控制表），现代的操作系统大都采用基于ACL 的方法。

3 .访问控制表

访问控制表是目前采用最多的一种实现方式。它可以对某一特定资源指定任意一个用户的访问权限，还可以将有相同权限的用户分组，并授予组的访问权， ACL的优点在于它的表述直观、易于理解，而目比较容易查出对某一特定资源拥有访问权限的所有用户，以便有效实施授权管理。在一些实际应用中还对ACL 进行了扩展，从而进一步控制用户的合法访问时间，是否需要审计等。 尽管ACL灵活方便，但将它应用到网络规模较大、需求复杂的企业内部网络时，就暴露出了一些问题：

（1）ACL需对每个资源指定可以访问的用户或组以及相应的权限。当网络中资源很多时，需要在ACL设定大量的表项。而且，当用户的职位、职责发生变化时，为反映这些变化，管理员需要修改用户对所有资源的访问权限。另外，在许多组织中，服务器一般是彼此独立的，各自设置自己的ACL ，为了实现整个组织范围内的一致的控制政策，需要各管理部门的密切合作。所有这些，都会使得访问控制的授权管理变得费力而繁琐，且容易出错。

（2）单纯使用ACL ，不易实现最小权限原则及复杂的安全政策。

4 .授权关系表

授权关系表弥补了基于ACL 和基于访问能力表的方法的不足，它的每一行（或称一个元组）都表示主体和客体的一个权限关系。如果该表按客体进行排序就拥有访问能力表的优势，如果按主体进行排序就拥有了访问控制表的好处。这种实现方式也特别适合采用关系数据库。

综合以上特点以及我们这个软件的需求，我们决定采用搜权关系表来实现访问控制。当用户输入用户名与密码时，我们将其与数据库内存储的数据进行比较、判断后，授予其进入软件、访问这个用户信息的权限。同理，对于每一个加锁的资源也有一张对应的表，通过密码来获得对这个资源的操作权限。

8.1.2.访问控制策略

自主访问控制、强制访问控制和基于角色的访问控制是系统中常用的3种访问控制策略，其中前两种（DAC 和MAC ）属于传统的访问控制策略。 以上每种策略不是绝对互斥的，我们可以把几种策略综合起来应用，以获得更好、更安全的系统保护效果。当使用多重策略的时候，只有各种策略的交集策略被允许，访问才被许可。当然，在某些场合下，也可能存在着一些冲突，比如被某一策略许可的访问被另一策略所禁止，这样就产生了冲突，这种情况需要在管理层通过协商来协调。

1 .自主访问控制 自主访问控制（DAC）是目前计算机系统中使用最多的访问控制机制，它是在确认主体身份以及（或）它们所属组的基础上对访问进行限制的一种方法，称其为自主型是因为在DAC 系统中，一个拥有一定访问权限的主体可以直接或间接地将权限传给其他主体。其基本思想是：允许某个主体显式地指定其他主体对该主体所拥有的信息资源是否可以访问以及可执行的访问类型。Windows、UNIX系统都采用了自主型的访问控制技术。 自主访问控制根据访问者的身份和授权来决定访问模式，主体访问者对访问的控制有一定的权利。但正是这种权利使得信息在移动过程中的访问权限关系会被改变。如用户A 可以将其对客体目标O 的访问权限传递给用户B ，从而使不具备对0 访问权限的B 也可以访问O ，这样就很容易产生安全漏洞，所以自主访问控制的安全级别很低。

2 .强制访问控制 强制访问控制是“强加”给访问主体的，即系统强制主体服从访问控制政策。MAC 主要用于多层次安全级别的军事应用当中，它预先将主体和客体分级，即定义用户的可信任级别及信息的重要程度（安全级别，比如可以分成绝密级、机密级、秘密级、无密级等），然后根据主体和客体的级别标记来决定访问模式，用户的访问必须遵守安全政策划分的安全级别以及有关访问权限的设定。当用户提出访问请求时，系统对主客体两者进行比较以确定访问是否合法。 在典型应用中，MAC 的访问控制关系可以分为两种：用上读／下写来保证数据完整性以及利用下读／上写来保证数据的保密性。它们都是通过梯度安全标签来实现信息的单向流通的。 强制访问控制（MAC）的优势最主要在于它能阻止“特洛伊木马”。一个“特洛伊木马”是在一个执行某些合法功能的程序中隐藏的代码，它利用运行此程序的主体权限违反安全策略，通过伪装成有用的程序在进程中泄漏信息。一个“特洛伊木马”能够以直接与非直接泄漏两种方式泄漏信息。对于前者，“特洛伊木马”使信息的安全表示不正确并泄漏给非授权用户；对于后者，“特洛伊木马”通过编制返回给一个主体的合法信息中的方式非直接泄露信息，例如，可能表面上某些提问需要回答，而实际上用户回答的内容被传送给了“特洛伊木马”。 阻止“特洛伊木马”的策略是基于非循环信息流，所以在一个级别上读信息的主体一定不能在另一个违反非循环规则的安全级别上写。同样，在一个安全级别上写信息的主体一定不能在另一个违反非循环规则的安全级别上读。由于MAC 策略是通过梯度安全标签实现信息的单向流通，因而它可以很好地阻止“特洛伊木马”的泄密。 强制访问控制（MAC）的主要缺陷在于实现工作量太大、管理不便、不够灵活，而且MAC 由于过于偏重保密性，对其他方面如系统连续工作能力、授权的可管理性等考虑不足。

3 .基于角色的访问控制 20 世纪如年代出现的一种基于角色的访问控制（RBAC）技术有效地克服了传统访问控制技术的不足，可以减少授权管理的复杂性，降低管理开销，而且还能为管理者提供一个比较好的安全实现政策的环境，从而成为了实施面向企业的安全策略的一种有效的访问控制方式。 角色是一个或一群用户在组织内可执行的操作组合。用户在一定的部门中具有一定的角色，其所执行的操作与其所扮演的角色的职能相匹配，这正是基于角色的访问控制的基本特征、即依据RBAC 策略，系统定义了各种角色，每种角色可以完成一定的职能，不同的用户根据其职能被赋予相应的角色，一旦某个用户成为某个角色的成员，则此用户可以完成该角色所具有的职能。 RBAC 根据用户在组织内所处的角色进行授权与访问控制。也就是说，传统的访问控制直接将访问主体（发出访问操作、存取要求的主动方）和客体（被调用的程序或欲存取的数据访问）相联系，而RBAC在中间加入了角色，通过角色沟通主体与客体。在RBAC 中，用户标识对于身份认证以及审计记录是十分有用的，但真正决定访问权限的是用户对应的角色标识。由于用户与客体无直接联系，他只有通过角色才能享有该角色所对应的权限，从而访问相应的客体，因此用户不能自主地将访问权限授予别的用户，这是RBAC与DAC的根本区别。RBAC 与MAC 的区别在于：MAC 是基于多级安全需求的，而RBAC 则不是，因为军用系统主要关心的是防止信息从高安全级流向低安全级，即限制“谁可以读／写什么信息”，而RBAC 的系统主要关心的是保护信息的完整性，即“谁可以对什么信息执行何种动作”．角色控制比较灵活，根据配制可以使某些角色接近DAC ，而某些角色更接近于MAC。角色由系统管理员定义，角色成员的增减也只能由系统管理员来执行，即只有系统管理员有权定义和分配角色，而且授权规定是强加给用户的，用户只能被动接受，不能自主地决定，用户也不能自主地将访问权限转而赋予他人，这是一种非自主型访问控制。最后要指出的是角色和组的区别。组通常仅仅作为用户的集合，而角色一方面是用户的集合，另一方面又是权限的集合，作为中间媒介将用户和权限连接起来。当然角色可以在组的基础上实现，这样就对保持原有系统非常有利。此时角色就成为了一个策略部件，与组织的授权、责任关系相联系，而组成为实现角色的工具，两者间是策略与实现机制的关系。

这里，我们采用的是基于角色的访问控制。因为这种实现方法便于用户访问、参与到各个环节中。

8.1.3.数据与文件的加密

数据加密的基本过程就是对原来为明文的文件或数据按某种算法进行处理，使其成为不可读的一段代码，通常称为“密文”，使其只能在输入相应的密钥之后才能显示出本来内容，通过这样的途径来达到保护数据不被非法人窃取、阅读的目的。 该过程的逆过程为解密，即将该编码信息转化为原来数据的过程。 根据加密原理的不同加密方式可分为对称式加密和非对称式加密，对称式加密就是加密和解密使用同一个密钥，通常称之为“Session Key ”，这种加密技术目前被广泛采用；非对称式加密就是加密和解密所使用的不是同一个密钥，通常有两个密钥，称为“公钥”和“私钥”，它们两个必需配对使用，否则不能打开加密文件，我们所熟悉的EFS加密便属于此种加密方式。 根据加密方式不同可以分为软件加密和硬件加密，软件加密顾名思义，基于软件层面的加密，由软件利用加密算法，对文件进行加密，需要通过加密时设置的密码才可访问文件，此类加密方式较常见，诸如OFFICE办公文档加密，RAR加密等等。而硬件加密则是利用外接的硬件设备的唯一特征对数据进行加密，比如加密狗，指纹加密，虹膜加密等。 根据加密算法不同，又分为以下多种：DES、3DES、RC2、RC4、IDEA、RS、DSA、AES、BLOWFIS、MD5、SSF33、SSF28、SCB2、ElGamal、Deffie-Hellman、新型椭圆曲线算法ECC等。 对于一个加密的数据来说，它具体被分为以下三个部分：数据部分、加密算法、密钥。数据部分由于加密已被彻底改变为不可读写的乱码，当然也有极少数软件只对文件入口地址加密而不对文件本身加密，此部分为解密的主要目的；加密算法则是数据加密时采用的一种算法的一些公式、法则或程序的组合；密钥则是加密是设置的口令密码，或者硬件特征代码。 根据以上所述我们可以看出，对于解密需要做的一是破解密码，这是最常见的方法，大多数文件的都采取了对称性加密，只有一个密钥，这个密钥大多会以暗文的形式存放在文件的某个位置，利用专门的软件和直接手工提取这串代码，而后进行专业破解，破解出的密码，在绝大多数情况下是可以解除文件的加密的。 当然如果使用了非对称加密，那么我们就必须要知道公钥和私钥，大多数情况下非对称加密的难度颇高，因为此类加密的解密密文存放在公钥中，而私钥则是用来从公钥里配对提取解密密文的，两者缺一不可，所以恢复难度十分大，一般来说如果公钥丢失多采用重建虚拟公钥，接着使用私钥进行解密，如果私钥丢失则利用特殊软件手工从公钥中提取解密的密文，但是这两种方法的成功率并不太高。 至于使用了硬件加密的文件，由于设备和精力有限，目前看来只有国外的极少数专业团体以及国家级的安全机构才具备破除此类密码的能力。 系统密码加密，此类加密破解难度不大，网上介绍的文章和工具也不少，因为系统的密码是存放在硬盘上一个特殊文件中的，只要删除该文件，绝大数情况便可以清除掉系统的密码，但也不排除该文件清除后非法登陆系统的情况。 最后是特殊的硬盘加密，硬盘加密其实是基于硬盘固件加密的一个子安全系统，被加密后的硬盘无法正常读写，需要使用专业的软件才能进行解密或者密码清除的工作。

这里，由于我们使用的是mysql数据库，所以是使用其提供的加密函数（AES\_ENCRYPT() ）和解密函数（AES\_DECRYPT()）。在其底层的实现中，应该是使用SHA1算法。

8.2.通信与网络

8.2.1.网络环境下危及安全的因素

在网络环境下可能危及信息安全的因素包括如下几个方面。

(1）伪装欺骗：指非法用户假冒合法用户身份获取敏感信息的行为。

(2）非法存取：窃取、篡改或破坏网络中存储的信息。

(3）否认抵赖：指通信方事后否认曾参与某次活动的行为。

(4）服务拒绝：指合法用户的正当申请被拒绝、更改或延迟等。

(5）截取破译：指以不正当手段获取门令或密钥，或者对加密信息与加密机制进行分析破译或篡改等。

危及路由安全方面的因素包括如下几个方面。

(1）侦听窃密：在通信信道中窃听所传输的信息。

(2）指定路由：非法用户精心设计一些绕过安全机制的路由或将信息误导向不正常的路由。

(3）身份攻击：指用户的身份或口令在通信时被非法截取。

(4）截取破坏：非法截取通信过程中的数据或者伪造、篡改与破坏信道中传输的信息。

(5）中继攻击：非法用户截取信息后延迟时间再发送。

(6）阻塞通信：利用网上的软硬件漏洞，发送大量无效信息使网络过载，或者设法使一些路径或网上关键互联设备瘫痪。

8.2.2.通信与网络的信息安全技术

1.IP层安全性： 在TCP/IP网络中，IP层实现的安全技术通常包括IP过滤技术和IP加密传输信道技术。其中IP 过滤技术被路由器和防火墙产品所广泛采用，是最常见的Internet安全技术。在IP 加密传输信道技术方面，IETF指定IPSEC 制订IP 安全协议（IPSP, IP Security Protocol）和对应的Internet密钥管理协议的标准。IPSP的主要目的是使需要安全服务的用户能够使用相应的加密安全体制。该体制应该是与算法独立的，可以自己选择和替换加密算法而不会对应用和上层协议产生任何影响。此外，该体制必须能支持多种安全政策，并且对其他不使用该体制的用户完全透明，IPSEC技术能够在两个网络节点间建立透明的安全加密信道。现在一些防火墙产品使用AH(Authentication Header，认证头部）或ESP ( Encapsulating Security Payload，安全内容封装）支持IPSEC来实现IP 层的加密，一些主要的路由器厂商也声称支持IPSEC。 IP 层安全性的主要优点是它的透明性，也就是说，安全服务的提供不需要应用程序、其他通信层次和网络部件做任何改动。它的最主要的缺点是：IP层一般对属于不同进程的包不作区别。对所有去往同一地址的包，它将按照同样的加密密钥和访问控制策略来处理，这会导致性能下降。针对面向主机密钥分配的这些问题，RFC1825推荐使用面向用户的密钥分配，其中不同的连接会得到不同的加密密钥。但是，面向用户的密钥分配需要对相应的操作系统内核进行比较大的改动。IP 层非常适合提供基于主机的安全服务。相应的安全协议可以用来在Internet上建立安全的IP 通道和虚拟专用网。例如，利用它对IP包的加密和解密功能，可以方便快捷地强化防火墙系统的防卫能力。

2 .传输层安全性：最常见的传输层安全技术有SSL、SOCKS和安全RPC等。 网景公司（Netscape）设计的安全套接层（SSL）协议分为两层，上面是SSL 协商层，双方通过协商层约定有关加密的算法、进行身份认证等；下面是SSL 记录层，它把上层的数据经分段、压缩后加密，由传输层传送出去。SSL 采用公钥方式进行身份认证，但是大量数据传输仍使用对称密钥方式。通过双方协商，SSL 可以支持多种身份认证、加密和检验算法。它提供3类基本的安全服务：信息加密服务（在客户机与服务器之间的所有业务都使用握手协议、协商密钥与算法进行加密）、信息完整性服务（采用机密共享与HASH 函数组确保SSL 业务可正常全部到达目的地）、相互认证服务（采用公开密钥技术编码标识符，使客户机与服务器在SSL握手时交换各自的标识符以实现相互认证）。 传输层安全机制的主要缺点就是对应用层不透明，应用程序必须修改以使用SSL 应用接口，而且要对传输层建立安全机制。同时公钥体系存在的不方便性SSL 也同样存在，例如用户很难记住自己的公钥和私钥，必须依靠某些物理设备如IC 卡或者磁盘来存储，这样对用户终端就有一定的要求，再有就是服务方和客户方必须依赖一个证书授权（CA ) 中心来签发证书，双方都必须将CA 的公钥存放在本地。为了保持Internet 上的通用性．目前一般的SSL 协议实现只要求服务器方向客户方出示证书以证明自己的身份，而不要求用户方同样出示证书。在建立起SSL信道后再加密传输用户名和口令以实现客户方的身份认证。 同网络层安全机制相比，传输层安全机制的主要优点是它提供基于进程对进程的（而不是主机对主机的）安全服务和加密传输信道，利用公钥体系进行身份认证，安全强度高，支持用户选择的加密算法。这一成果再加上应用级的安全服务，就可以提供更加安全可靠的安全性能了。

3.应用层安全性：IP 层和网络层（传输层）的安全协议都无法根据所传送内容不同的安全要求区别对待。如果确实想要区分每个具体文件的不同的安全性要求，就必须在应用层采用安全机制。本质上，这意味着真正的（或许再加上机密的）数据通道还是建立在主机（或进程）之间，但却不可能区分在同一通道上传输的一个个具体文件的安全性要求。比如说，如果一个主机与另一个主机之间建立起一条安全的IP 通道，那么两个进程间传输的所有报文都会自动被加密。提供应用层的安全服务实际上是最灵活的处理单个文件安全性的手段。例如，一个电子邮件系统可能需要对要发出的信件的个别段落实施数据签名。较低层的协议提供的安全功能一般不会知道任何要发出的信件的段落结构，从而也不可能知道该对哪一部分进行签名。应用层是惟一能够提供这种安全服务的层次。

4.访问控制与目录管理：资源的合法使用主要依靠网络管理员进行必要的访问控制与目录管理，用以保证网络资源不被非法用户使用或破坏，也可保证数据资源不被非法读出或改写。 根据企业的安全策略，网络管理员应当使用由网络操作系统、数据库管理系统或相应的网络安全软件等提供的手段，进一步实施访问控制与目录管理方面的措施。

5.数据加密 随着密码技术的进步，现在己发展了许多种数据加密算法与技术。

6.身份验证与鉴别 除了对用户身份进行验证与鉴别，也可以对信息的真实可靠性进行验证与鉴别，用来解决冒充、抵赖、伪造或篡改等问题，除了标识用户加口令的机制外，最常用的是“数字签名”技术： 数字签名可以采用秘密密钥或公开密钥技术实现，也可以使用“消息摘要”方式配合实现。签名算法可用RSA、DSS数字签名标准等。考虑到身份验证与鉴别时可能采用相当复杂的算法或技术，目前已开始流行委托专业认证鉴别机构或公司进行身份鉴别与验证，目的是采用更先进的技术做好认证与鉴别这一关键性的安全保卫工作。 RADIUS（远程拨入用户认证服务）是一种管理标准，可为分布式网络上的认证服务器提供管理服务，用来实现集中管理方式下的用户认证、口令加密、服务选择、信息过滤以及账户核对等有关任务。一个单独的RADIUS数据库服务器可能在多个网络上同时管理多个安全系统，其中保存有与网内所有用户有关的认证信息（如存取限制、特定的路由表、数据分组过滤规则或约定、可供核对的账户信息等），可以用来维护成千上万个用户的信息安全。

7.Kerberos协议 Kerberos是为分布式系统提供的认证方案，能为每种服务提供可信任的第三方认证服务。在Kerberos系统中，只有事先在其中登记过的客户才可以申请服务。客户希望得到某一服务，需要向认证服务器申请一张能使用此服务的入场券（ticket)，然后该客户把入场券传给入场券分配服务器（TGS）实施入场券的检查，在得到证实后才能使用相应的服务。其中加入了个人身份确认机制以及客户与服务器通信中的数据加密，通过系统维护数据库的方式，保存着客户、服务、权限与密钥等方面的相应信息。这也是我们实现的方法。

9.测试部分

9.1.测试方针

软件测试是软件质量保证的主要活动之一，其目的就是通过测试来验证和确认每一个软件项目的提交，通过一种及时的方式来识别这些交付物的缺陷。具体方针是：对于项目中的每一个提交都建立相应的定性和定量的测试标准，然后及时地通过测试验证其是否达到标准。发现缺陷后，必须对其进行记录和跟踪，直到缺陷被修正。其中，每个阶段都必须有对应的提交，对每个提交都必须组织一套有效的模型和工具来测试。必要时进行回归测试。

9.2.集成策略

软件测试的集成策略包括：1）大爆炸集成

优点：可以迅速完成集成测试，并且只要极少数的驱动和桩模块，用例也是最少的，简单，资源利用率高。

缺点：一次试运行成功的可能性不大，问题定位和修改比较困难，许多接口错误很容易躲过测试。

适应于一个维护型项目或被测试系统较小的情况。

2）自顶向下集成

优点：较早地验证了主要控制和判断点，按深度优先可以首先实现和验证一个完整的软件功能，功能较早证实，带来信心，只需一个驱动，减少驱动器开发的费用，支持故障隔离。

缺点：柱的开发量大，底层验证被推迟，底层组件测试不充分。

适应于产品控制结构比较清晰和稳定，高层接口变化较小，底层接口未定义或经常可能被修改，接口控制组件具有较大的技术风险，需要尽早被验证，希望尽早能看到产品的系统功能行为。

3）自底向上集成

优点：对底层组件行为较早验证，工作最初可以并行集成，比自顶向下效率高，减少了桩的工作量，支持故障隔离。

缺点：驱动的开发工作量大，对高层的验证被推迟，设计上的错误不能被及时发现。

适应于底层接口比较稳定，高层接口变化比较频繁，底层组件较早被完成。

4）三明治集成

优点：集合了自顶向下和自底向上两种策略的优点

缺点：中间层测试不充分

适应于大部分软件开发项目

5）基干集成

优点：具有三明治集成的优点，更适合于大型复杂项目的集成。

缺点：必须对系统的结构和相互依存性进行仔细的分析，驱动和桩开发量大，局部采用了大爆炸的策略，有些接口可能测试不充分。

嵌入式系统中常用。

6）分层集成

适应于有明显层次关系的系统。

7）基于功能的集成

优点：优先验证关键功能的正确性，减少驱动的开发，进度要快。

缺点：对接口测试不充分，有较大的冗余测试。

8）基于消息的集成

优点：优先验证关键消息的正确性；减少驱动的开发，进度要快。

缺点：对接口测试不充分，有较大的冗余测试。

9）基于风险的集成

优点：最具有风险的组件最早进地验证，有助于系统的快速稳定。

缺点：需要对各组件的风险有一个清晰的分析。

10）基于进度的集成

优点：具有较高的并行度，能够有效缩短项目的开发进度。

缺点：桩和驱动工作量较大，有些接口测试不充分，有些测试重复和浪费。

在此，我们采用了基于功能、基于风险以及基于进度的集成策略。同时，由于项目工程较小，在局部还采用了大爆炸集成策略。

9.3.特殊考虑

|  |  |
| --- | --- |
| **下拉列表** | 1、 条目内容的检查，对照需求说明察看条目内容和实际内容是否一一对应。 |
| 2、 条目的功能能否实现，逐一执行列表框中每个条目的功能。 |
| 3、 在列表框中能否输入数据，检查能否输入或则粘贴数据向组合列表框内。 |
| 4、 能及时获取得到新增加的数据并显示。 |
| **文本框** | 1、 边界值和等价类测试用例方法。 |
| 2、 可以采用随机测试进行测试用例的补充。 |
| 3、 输入符合规定的数据。 |
| 4、 输入已经存在的内容。 |
| 5、 输入超常字符。 |
| 6、 输入特殊字集。 |
| 7、 输入空白，或则空格。 |
| **复选框** | 1、 多个复选框被选中。 |
| 2、 多个复选框可以被部分选中。 |
| 3、 多个复选框可以不被选中。 |
| 4、 逐一执行每个复选框的功能单选框的测试 。 |
| **单选框** | 1、 单选按钮是否只能同时选中选中一个。 |
| 2、 单选按钮的功能是否正确完成。 |
| 3、 是否有默认被选中的选项命令按钮的测试。 |

此外，对非法操作应有足够的说明，让用户明白错误出在何处。同时，对于无法恢复的操作应给予足够的信息提示，并提供用户放弃操作的机会。在进行以上操作后，还必须考虑一些连续的动作是否会对结果造成影响，例如一些认证后是否会造成安全证书的泄露或者被利用，同时还必须考虑在使用高峰期对于应用性能的压力，以及此时系统性能的分配问题，以下应该是我们考虑资源分配的优先级：

1.资源库、发布、收藏；

2.加密、私信、用户信息；

3.校友圈、设置。

最后，我们还应考虑在响应时间和成功率之间的权衡。

10.附录

10.1.项目名称

确定项目名字为“Bang+”，寓意有二，一是校友通过资源直接互帮互助，二是重“磅”干货。希望能够通过这个项目的app来帮助即将走出校园的更多人。

10.2.项目logo



10.3.项目进度时间表

10.3.1.项目历史

|  |  |  |
| --- | --- | --- |
| **活动** | **产出** | **时间** |
| 讨论 | 确定项目选题 | 03.25 |
| 需求分析 | 产品原型，需求规约报告 | 03.28 |
| 设计分析 | 设计规约报告 | 04.10 |
| 代码实现 | 服务器端部分 | 04.20 |
|  | 安卓发布模块 | 04.24 |
| 10.3.2.接下来要做的事 | | |
| 需求重分析 | 完善产品原型 | 04.30 |
| 代码实现 | 连接服务端和安卓端 | 05.03 |
|  | 安卓资源库模块 | 05.08 |
|  | 测试资源库和发布模块 | 05.11 |
|  | 安卓用户信息和加密模块 | 05.25 |
|  | 测试，包括第一级 | 05.31 |
|  | 视完成情况评估是否完成第三级 | 06.06 |