大连海事大学2020-2021（2）学期 序号：21

移动应用开发课程实践

项目名称：宿舍管理系统

项目组名称：雄哥冲冲冲

项目组成员：

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 学号 | 任务 | 成绩 |
| 陈一可 | 2220181794 | 组长 |  |
| 温钧雄 | 2220181398 | 组员 |  |
| 何思源 | 2220180677 | 组员 |  |
| 张竣淇 | 2220181900 | 组员 |  |

项目组成员完成工作量：

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 学号 | 主要完成工作 | 占总工作量比例 |
| 陈一可 | 2220181794 | 管理员端代码、总管理员端代码 | 50% |
| 温钧雄 | 2220181398 | 数据库搭建、公告功能实现 | 20% |
| 何思源 | 2220180677 | 学生端代码 | 20% |
| 张竣淇 | 2220181900 | 界面布局设计、撰写实验报告 | 10% |

目 录

1.需求分析

2.功能设计

3.功能实现

4.运行测试

5.总结及展望

**1.需求分析**

·宿舍状态用图标显示，显示空闲或已住满，用图标颜色进行区分。

·宿舍详细状况显示，点击宿舍图标显示空床位信息。

·设置余额查询模块，显示宿舍水电费余额、学生个人网费余额。

·设置失物招领模块，实现失物招领登记、认领、查询。

·设置报修管理模块，实现学生报修登记，管理员报修处理登记、管理员和学生查询。

·床位管理：实现床位入住、退舍、交换。

·宿舍缴费情况：显示欠费学生信息，查询已欠费学生。

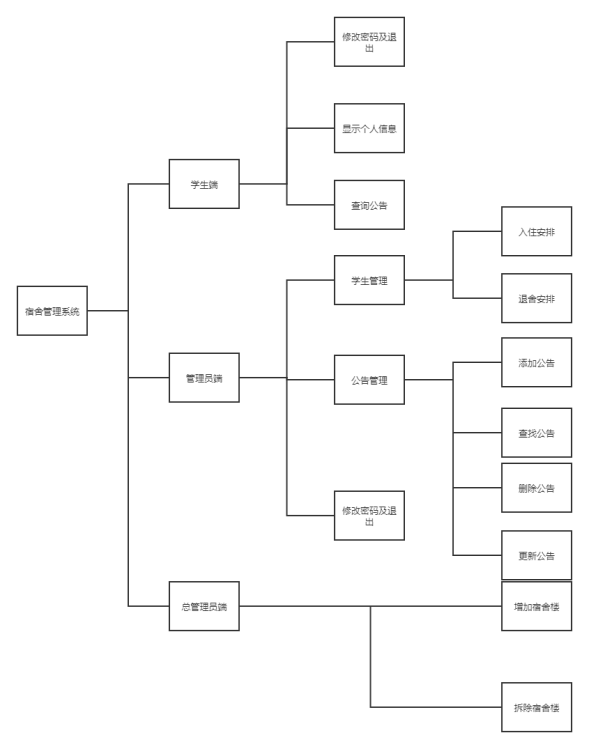
·留校申请：管理员留校审批、学生留校申请。

·设置公告，管理员可向所管理宿舍楼学生发布有关宿舍的公告，并包括增删改查功能，学生可浏览公告

·设置超级管理员管理所有宿舍楼的宿舍配置、宿舍楼修建、改造、废弃及学生批量入住、退校。

**2.功能设计**

系统结构：



模块设计：系统分三个模块，分别是学生端、管理员端和总管理员端。

接口：无

数据设计：

数据库表的设计如下：

**public class** MySqlite **extends** SQLiteOpenHelper {  
 **public static** String *User* = **"create table User( UserID int(10) primary key,Password varchar(15),RoomID int(10),UserName text,"** +  
 **"Sex int(4),CostComments int(4),NetFees int(10),Comments text)"**;//用户表  
 **public static** String *Room* = **"create table Room(RoomID int(10) primary key,Total int(4),Residual int(2),"** +  
 **"RoomType int(4),WAE\_Fees int(20),Comments text,BuildID int(10))"**;//Room表  
 **public static** String *Bed* = **"create table Bed(BedID int(10) ,UserID int(10),Free int(2),RoomID int(10) ,Comments text,primary key(BedID,RoomID))"**;//床表  
 **public static** String *Admin* = **"create table Admin(AdID int(10) primary key,PassWord varchar(20),BuildID int(10),Comments text)"**;//管理员表  
 **public static** String *Building* = **"create table Building(BuildID int(10) primary key,Floor int(4),Rsum int(10),Money int(10),Comments text)"**;//宿舍楼表  
 **public static** String *Notice* = **"create table Notice(NID integer PRIMARY KEY AUTOINCREMENT,AdID int(10),Title text,Content text,Time datetime,Comments text)"**;//公告表  
 **public static** String *General\_Admin* = **"create table General\_Admin(GAID int(10) primary key,password varchar(15),Comments text)"**;//总管理员表  
 **public static** String *Building\_Arrange* = **"create table Building\_Arrange(BuildID int(10) ,GAID int(10) ,Arrangement text,Time datetime,Comments text,primary key(BuildID,GAID))"**;//宿舍楼管理表  
  
 **public** MySqlite(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, **int** version) {  
 **super**(context, name, factory, version);  
 }

**3.功能实现**

3.1登录

在Login的活动里提供登录功能，包含了三种用户登录，分别为学生，管理员，总管理员。通过文本框输入账号名和密码，点击登录进行校验，通过游标对login传来的账号名进行数据库查询匹配，如果匹配到和账号密码相关的条目，就登录成功，通过intent跳转到相对应Activity并把它的账号通过putExtra传到下一个Activity，如管理员则为AdminHome，为反之用toast显示账号或密码错误。

3.1.1 Login

3.1.1.1布局

采用约束布局，通过使用ImageView,EditText,TextView的组合实现简单的登录界面。

3.1.1.2方法

3.1.1.2.1 onCreate()，dbQueryuserid()

设计思路：连接和创建数据库，并定义初始用户类型。以及先定义一个方法，作用是根据传递的用户类型到相应的表中查询，然后将查询值与输入的值对比，相同则返回1，不同则返回0，

关键代码：

String **type**;//用户类型

**sqlite** = **new** MySqlite(**this**, **"Dormitory.db"**, **null**, 1);  
 **myDb** = **sqlite**.getWritableDatabase();*//打开数据库myDb*

**public int** dbQueryuserid(String id, String types, String pwd) //判断用户类型

{  
 String sql = **null**;  
 **if** (types.equals(**"User"**)) {  
 sql = **"select password from '"** + types + **"'where userId='"** + id + **"'"**;  
 Cursor cursor = **myDb**.rawQuery(sql, **null**);  
 **if** (cursor.moveToNext()) {  
 String pwd1 = cursor.getString(cursor.getColumnIndex(**"Password"**));//从数 据库里读出用户密码  
 **if** (pwd.equals(pwd1))//密码比较  
 **return** 1;  
 }

3.1.1.2.2 onClick()

设计思路:onClick(),dbQueryuserid()根据返回值和用户类型决定跳转到相应的用户界面。

**public void** enter(View view) {  
 **int** j = 0;  
 EditText e1 = findViewById(R.id.***t***);  
 EditText e2 = findViewById(R.id.***t1***);  
 String id = e1.getText().toString();  
 String pwd = e2.getText().toString();  
 **if** (id != **null** && pwd != **null**)  
 j = dbQueryuserid(id, **type**, pwd);//调用判断方法，并赋值给j  
 **if** (j == 1) //说明用户输入密码和数据库密码一致

{  
 **if** (**type**.equals(**"User"**))//判断用户类型

{  
 Intent intent = **new** Intent(**this**, StudentNotice.**class**);  
 intent.putExtra(**"studentId"**, id);//将用户id存入Intent,传递到下一个页面  
 startActivity(intent);  
 } **else if** (**type**.equals(**"Admin"**)) {  
 Intent intent = **new** Intent(**this**, AdminHome.**class**);  
 intent.putExtra(**"adId"**, id);  
 startActivity(intent);  
 } **else if** (**type**.equals(**"General\_Admin"**)) {  
 Intent intent = **new** Intent(**this**, GeneralAdminHome.**class**);  
 intent.putExtra(**"GAId"**, id);  
 startActivity(intent);  
 }  
 }  
 **if** (j == 0)//密码输入错误  
 Toast.makeText(**this**, **"账户或密码错误"**, Toast.***LENGTH\_LONG***).show();  
}

3.2管理员房间搜索和显示

3.2.1：AdminHome

3.2.1.1：布局

采用约束布局，一个EditText回车搜索事件，采用RecyclerViwew嵌套一个线性布局，线性布局里用两个Button用来点击返回不同管理员功能页面

3.2.1.2：方法

3.2.1.2.1:onCreat()

打开数据库，将登录界面传过来的ID用来查找宿舍楼号（因为我们管理员的ID号是和宿舍楼绑定的），然后将房间的房间ID号和空余床数读出来，如果有空余的为房间图标为绿色，满员则房间颜色为红色。关于搜索，给EditText添加一个编辑监听器，用来返回搜索结果返回，通过数据库相似查询，得到相似查询的RoomID和剩余床数，然后再通过所创建的适配器返回给刷新过的listview里面，EditText设置为回车搜索。

关键代码以及注释：

**if** (cursor.moveToFirst()) {  
 **do** {  
 **int** roomId = cursor.getInt(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"RoomID"**))));  
 **int** residual = cursor.getInt(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"Residual"**))));  
 **if** (residual > 0) {  
 **room** = **new** Room(R.drawable.***ic\_room\_green***, Integer.*toString*(roomId));  
 } **else** {  
 **room** = **new** Room(R.drawable.***ic\_room\_red***, Integer.*toString*(roomId));  
 }  
 **rooms**.add(**room**);  
 } **while** (cursor.moveToNext());  
 }  
 **final** RecyclerView recyclerView = (RecyclerView) findViewById(R.id.***recyclerView***);  
 recyclerView.setLayoutManager(**new** StaggeredGridLayoutManager(4, StaggeredGridLayoutManager.***VERTICAL***));  
 recyclerView.setAdapter(**new** RoomAdapter(**rooms**));  
  
 **final** EditText searchBar = (EditText) findViewById(R.id.***searchBar***);  
 searchBar.setOnEditorActionListener(**new** TextView.OnEditorActionListener() {  
 @Override  
 **public boolean** onEditorAction(TextView textView, **int** actionID, KeyEvent keyEvent) {  
 **if** (actionID == EditorInfo.***IME\_ACTION\_SEND*** || ((keyEvent != **null** && keyEvent.getKeyCode() == KeyEvent.***KEYCODE\_ENTER***) && keyEvent.getAction() == KeyEvent.***ACTION\_DOWN***)) {  
 **rooms**.clear();  
 String searchMsg = searchBar.getText().toString();  
 Cursor cursor = **myDb**.rawQuery(**"select \* from Room where RoomID like ? and BuildID = ?"**, **new** String[]{**"%"** + searchMsg + **"%"**, **adId**});  
 **int** i = cursor.getColumnIndex(**"RoomID"**);  
 **int** j = cursor.getColumnIndex(**"Residual"**);  
 **if** (cursor.moveToFirst()) {  
 **do** {  
 String roomNumber = cursor.getString(i);  
 **int** residual = cursor.getInt(j);  
 **if** (residual > 0)  
 **room** = **new** Room(R.drawable.***ic\_room\_green***, roomNumber);  
 **else  
 room** = **new** Room(R.drawable.***ic\_room\_red***, roomNumber);  
 **rooms**.add(**room**);  
 Log.*d*(**"SS"**, **"dwada"** + **room**.getRoomNumber());  
 } **while** (cursor.moveToNext());  
 }  
 RoomAdapter adapter = **new** RoomAdapter(**rooms**);  
 recyclerView.setAdapter(adapter);  
 adapter.notifyDataSetChanged();  
 **return true**;  
 }  
 **return false**;  
 }  
 });  
  
}

3.2.1.2.2:onClick()

两个点击事件，用来返回管理员界面的两个菜单栏，为AdminMy和AdminNotice，一个为主页面，另一个为通知界面。通知界面功能为接下来说明

关键代码：

EditText监听器，用来模糊查询宿舍号

searchBar.setOnEditorActionListener(**new** TextView.OnEditorActionListener() {  
 @Override  
 **public boolean** onEditorAction(TextView textView, **int** actionID, KeyEvent keyEvent) {  
 **if** (actionID == EditorInfo.***IME\_ACTION\_SEND*** || ((keyEvent != **null** && keyEvent.getKeyCode() == KeyEvent.***KEYCODE\_ENTER***) && keyEvent.getAction() == KeyEvent.***ACTION\_DOWN***)) {  
 **rooms**.clear();  
 String searchMsg = searchBar.getText().toString();  
 Cursor cursor = **myDb**.rawQuery(**"select \* from Room where RoomID like ? and BuildID = ?"**, **new** String[]{**"%"** + searchMsg + **"%"**, **adId**});  
 **int** i = cursor.getColumnIndex(**"RoomID"**);  
 **int** j = cursor.getColumnIndex(**"Residual"**);  
 **if** (cursor.moveToFirst()) {  
 **do** {  
 String roomNumber = cursor.getString(i);  
 **int** residual = cursor.getInt(j);  
 **if** (residual > 0)  
 **room** = **new** Room(R.drawable.***ic\_room\_green***, roomNumber);  
 **else  
 room** = **new** Room(R.drawable.***ic\_room\_red***, roomNumber);  
 **rooms**.add(**room**);  
 Log.*d*(**"SS"**, **"dwada"** + **room**.getRoomNumber());  
 } **while** (cursor.moveToNext());  
 }  
 RoomAdapter adapter = **new** RoomAdapter(**rooms**);  
 recyclerView.setAdapter(adapter);  
 adapter.notifyDataSetChanged();  
 **return true**;  
 }  
 **return false**;  
 }  
});

跳转到管理员主页面

**public void** navigationMy(View view) {  
 Intent intent = **new** Intent(**this**, AdminMy.**class**);  
 intent.putExtra(**"adId"**, **adId**);  
 startActivity(intent);  
 overridePendingTransition(0, 0);  
 }  
跳转到管理员通知功能页面  
 **public void** navigationNotice(View view) {  
 Intent intent = **new** Intent(**this**, AdminNotice.**class**);  
 intent.putExtra(**"adId"**, **adId**);  
 startActivity(intent);  
 overridePendingTransition(0, 0);  
 }

3.3：管理员通知删除

3.3.1：NoticeAdapter

3.3.1.1:布局

LinearLyaout嵌套LinearLayout，每个被嵌套的Linearlayout里面有两个按钮和TextView，每个被嵌套的LinearLayout就是每个ListView的一项Item

两个按钮分别为删除公告和更新公告

3.3.1.2：方法

3.3.1.2.1：onCreate()

使用Adapter来支持ListView的Item点击事件，创建一个java类用来封装一个notice，通过封装的notice得到管理员账号（通知有发布这条公告的管理员ID），点击删除按钮，通过封装的notice得到notice的自增NID，然后通过数据库查询查出信息，最后用delete语句删除。

关键代码

**public** NoticeAdapter(Context context, **int** resource, List<Notice> notices) {  
 **super**(context, resource, notices);  
 **resId** = resource;  
 **this**.**context** = context;  
}

**public** View getView(**int** position, @Nullable View convertView, @NonNull ViewGroup parent) {  
 View view;  
 view = LayoutInflater.*from*(getContext()).inflate(**resId**, parent, **false**);  
 **notice** = getItem(position);  
 TextView title = (TextView) view.findViewById(R.id.***title***);  
 title.setText(**notice**.getTitle());  
 TextView time = (TextView) view.findViewById(R.id.***time***);  
 time.setText(**notice**.getTime());  
 **AdID** = **notice**.getAdID();

3.3.1.2.2：onClick()

两个点击事件，一个为跳转到编辑界面，编辑界面为更新公告信息，另一个点击事件为删除公告信息，删除的具体实现在上面说明了。点击删除按钮，通过notice.getNID得到该条信息的NID，然后数据库定位删除。

关键代码：

跳转到公告更新页面，并传递NID过去

Button button1 = (Button) view.findViewById(R.id.***button1***);  
button1.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 Intent intent = **new** Intent(**context**, AdminNoticeUpdate.**class**);  
 intent.putExtra(**"NID"**, **notice**.getNid());  
 **context**.startActivity(intent);  
 }  
});

ListView 中的Item的删除按钮，用来删除公告

Button button2 = (Button) view.findViewById(R.id.***button2***);  
button2.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 AlertDialog.Builder dialog = **new** AlertDialog.Builder(**context**);  
 dialog.setTitle(**"是否要删除这条数据"**);  
 dialog.setCancelable(**true**);  
 dialog.setNegativeButton(**"确认"**, **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialogInterface, **int** i) {  
 **sqlite** = **new** MySqlite(**context**, **"Dormitory.db"**, **null**, 1);  
 **myDb** = **sqlite**.getWritableDatabase();  
 **myDb**.delete(**"Notice"**, **"NID=?"**, **new** String[]{Integer.*toString*(**notice**.getNid())});  
 Toast.*makeText*(**context**, **"删除成功!"**, Toast.***LENGTH\_LONG***).show();  
 Intent intent = **new** Intent(**context**, AdminNotice.**class**);  
 intent.putExtra(**"adId"**, **AdID**);  
 **context**.startActivity(intent);  
 }  
 });  
 dialog.setPositiveButton(**"取消"**, **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialogInterface, **int** i) {  
 }  
 });  
 dialog.show();  
  
 }  
});

3.4 用户密码的修改

在各个用户的个人主页界面，显示个人信息，通过点击修改密码，通过intent跳转到修改密码的页面，并传递用户的id到跳转页面，后用户输入原密码，输入新密码，再次确认密码，通过用户id查找对应的表的密码，对比原密码与输入的原密码，如果相等则修改密码，如果不等则通过Toast提示错误。

3.4.1：StudentMy,AdminMy,GeneralAdmin,ChangePassword

3.4.1.1:布局

采用约束布局，通过使用ImageView,EditText,TextView的组合实现简单的登录界面。

3.4.1.2：方法

onCreate() ChangePassword()

设计思路：创建连接数据库，设置点击事件，根据用户类型，判断去某个表查询密码,定义。

**private** String **id**;  
 **private** String **type**;//定义用户类型  
 **private** String **password** = **null**;  
 **private** String **newPassword** = **null**;

**sqlite** = **new** MySqlite(**this**, **"Dormitory.db"**, **null**, 1);  
 **myDb** = **sqlite**.getWritableDatabase();*//打开数据库myDb*

Intent intent = getIntent();  
 **id** = intent.getStringExtra(**"id"**);  
 **type** = intent.getStringExtra(**"type"**);//获取传递的用户类型

**final** EditText originalPassword = (EditText) findViewById(R.id.***originalPassword***);

Cursor cursor;  
 **if** (**type**.equals(**"admin"**)) {  
 cursor = **myDb**.rawQuery(**"select \* from Admin where AdID=?"**, **new** String[]{**id**});// **if** (cursor.moveToFirst()) {  
 **password** = cursor.getString(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"PassWord"**))));//根据用户类型查找密码并赋值给定义的password

}

onClick()

设计思路：将用户id通过Intent传递到修改密码的页面，如果输入的原密码和查找到的密码相等，则修改密码，反之提示错误。

**public void** onClick(View v) {  
 **if** (originalPassword.getText().toString().equals(**password**) && newPassword\_1.getText().toString().equals(newPassword\_2.getText().toString())) //判断用户输入的原密码是否正确以及两次输入的新密码是否相等

{  
 **newPassword** = newPassword\_1.getText().toString();  
 String sqliteUpdate = **null**;  
 **if** (**type**.equals(**"admin"**)) {  
 sqliteUpdate = **"UPDATE Admin SET PassWord ="** + **newPassword** + **" WHERE AdID="** + **id**;  
 **myDb**.execSQL(sqliteUpdate);//根据用户类型更新相应表的密码  
 } **else if** (**type**.equals(**"student"**)) {  
 sqliteUpdate = **"UPDATE User SET Password ="** + **newPassword** + **" WHERE UserID="** + **id**;  
 **myDb**.execSQL(sqliteUpdate);  
 } **else** {  
 sqliteUpdate = **"UPDATE General\_Admin SET password ="** + **newPassword** + **" WHERE GAID="** + **id**;  
 **myDb**.execSQL(sqliteUpdate);  
 }  
 Toast toast = Toast.makeText(ChangePassword.**this**, **"更改成功！"**, Toast.***LENGTH\_SHORT***);  
 toast.show();  
 finish();  
 } **else** {  
 **if** (!originalPassword.equals(**password**))//密码错误或者不相等  
 Toast.makeText(ChangePassword.**this**, **"原密码错误！"**, Toast.***LENGTH\_SHORT***).show();  
 **else** Toast.makeText(ChangePassword.**this**, **"两次密码不相同！"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
});

3.5：管理员公告更新

3.5.1：AdminNoticeUpdate

3.5.1.1布局 整体采用线性布局，嵌套TableLayout，使用TextView和EditText组合完成简单的布局。

3.5.1.2方法

3.5.1.2.1:onCreate()

链接数据库，通过上个页面NoticeAdapter通过intent传来的NID进行数据库定位查询，将该NID的公告标题，内容，时间，全部显示在公告的EditText里面。绑定点击事件，三个判断标题内容时间不能为空，最后通过getText.toString将目前的EditText的内容输入到该条NID的公告之中，更新原本的内容

**sqlite** = **new** MySqlite(AdminNoticeUpdate.**this**, **"Dormitory.db"**, **null**, 1);  
**myDb** = **sqlite**.getWritableDatabase();//打开数据库  
Intent intent = getIntent();  
**NID** = intent.getIntExtra(**"NID"**, 0);//获取NID

**ET\_noticeTitle** = (EditText) findViewById(R.id.***ET\_noticeTitle***);  
**ET\_noticeContent** = (EditText) findViewById(R.id.***ET\_noticeContent***);  
**ET\_noticeTime** = (TextView) findViewById(R.id.***ET\_noticeTime***);  
**ET\_ID** = (TextView) findViewById(R.id.***TV\_noticeId***);  
Cursor cv = **myDb**.rawQuery(**"select \* from Notice where NID=?"**, **new** String[]{Integer.*toString*(**NID**)});  
**if** (cv.moveToFirst()) {  
 **ET\_ID**.setText(String.*valueOf*(cv.getString(cv.getColumnIndex(**"NID"**))));  
 **ET\_noticeTime**.setText(String.*valueOf*(cv.getString(cv.getColumnIndex(**"Time"**))));  
 **ET\_noticeTitle**.setText(String.*valueOf*(cv.getString(cv.getColumnIndex(**"Title"**))));  
 **ET\_noticeContent**.setText(String.*valueOf*(cv.getString(cv.getColumnIndex(**"Content"**))));//将公告内容绑定到控件上  
}

3.5.1.2.2：onClick()

只有一个点击事件，该点击事件是三个判断语句和用来更新公告内容，也就是数据库UPDATE语句写在点击事件之内

Button update = (Button) findViewById(R.id.***BtnUpdate***);  
 update.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **if** (**ET\_noticeTitle**.getText().toString().equals(**""**)) {  
 Toast.makeText(AdminNoticeUpdate.**this**, **"公告标题输入不能为空!"**, Toast.***LENGTH\_LONG***).show();  
 **ET\_noticeContent**.setFocusable(**true**);  
 **ET\_noticeContent**.requestFocus();  
 **return**;  
  
 }  
 **if** (**ET\_noticeContent**.getText().toString().equals(**""**)) {  
 Toast.makeText(AdminNoticeUpdate.**this**, **"公告内容输入不能为空!"**, Toast.***LENGTH\_LONG***).show();  
 **ET\_noticeContent**.setFocusable(**true**);  
 **ET\_noticeContent**.requestFocus();  
 **return**;  
 }  
 **if** (**ET\_noticeTime**.getText().toString().equals(**""**)) {  
 Toast.makeText(AdminNoticeUpdate.**this**, **"公告时间输入不能为空!"**, Toast.***LENGTH\_LONG***).show();  
 **ET\_noticeContent**.setFocusable(**true**);  
 **ET\_noticeContent**.requestFocus();  
 **return**;  
 } **else** {  
 ContentValues cv = **new** ContentValues();  
 cv.put(**"Content"**, **ET\_noticeContent**.getText().toString());  
 cv.put(**"Time"**, **ET\_noticeTime**.getText().toString());  
 cv.put(**"Title"**, **ET\_noticeTitle**.getText().toString());//存入数据Content,Time,Title  
 **myDb**.update(**"Notice"**, cv, **"NID=?"**, **new** String[]{String.*valueOf*(**NID**)});//更新公告  
 Toast.makeText(AdminNoticeUpdate.**this**, **"更新成功!"**, Toast.***LENGTH\_SHORT***).show();  
 finish();  
 }  
 }  
 });  
}

3.6：管理员公告添加

3.6.1：AdminNoticeAdd

3.6.1.1布局

整体为LinearLayout，一个TextView用来显示添加公告，用TableLayout里嵌套三个TableRow，每个TableRow都有TextView用来提示接下里的EditText要输入什么内容。

3.6.1.2方法

3.6.1.2.1：onCreate()

链接数据库，三个判断语句公告标题，时间，内容不能为空，接收上个活动intent传来的AdID，通过sql语句将标题，时间和内容已经该公告绑定的管理员ID（也就是AdID）添入数据库之中

相关代码：

Intent intent = getIntent();  
 **adId** = intent.getStringExtra(**"adId"**);  
 Button add = (Button) findViewById(R.id.***add***);  
 **ET\_noticeTitle** = (EditText) findViewById(R.id.***ET\_noticeTitle***);  
 **ET\_noticeContent** = (EditText) findViewById(R.id.***ET\_noticeContent***);  
 **ET\_noticeTime** = (TextView) findViewById(R.id.***ET\_noticeTime***);  
 **ET\_noticeTime**.setText(**ly\_time**);

3.6.1.2.2：onClick()

监听器用来把数据添加到数据库之中，相关的sql语言放在监听器里面，点击就执行插入。

相关代码：

add.setOnClickListener(**new** View.OnClickListener() {  
 **public void** onClick(View view) {  
 **if** (**ET\_noticeTitle**.getText().toString().equals(**""**)) {  
 Toast.*makeText*(AdminNoticeAdd.**this**, **"公告标题不能为空!"**, Toast.***LENGTH\_LONG***).show();  
 **ET\_noticeContent**.setFocusable(**true**);  
 **ET\_noticeContent**.requestFocus();  
 **return**;  
 }  
 **if** (**ET\_noticeContent**.getText().toString().equals(**""**)) {  
 Toast.*makeText*(AdminNoticeAdd.**this**, **"公告内容不能为空!"**, Toast.***LENGTH\_LONG***).show();  
 **ET\_noticeContent**.setFocusable(**true**);  
 **ET\_noticeContent**.requestFocus();  
 **return**;  
 }  
 **if** (**ET\_noticeTime**.getText().toString().equals(**""**)) {  
 Toast.*makeText*(AdminNoticeAdd.**this**, **"公告时间不能为空!"**, Toast.***LENGTH\_LONG***).show();  
 **ET\_noticeContent**.setFocusable(**true**);  
 **ET\_noticeContent**.requestFocus();  
 **return**;  
 } **else** {  
 ContentValues contentValues = **new** ContentValues();  
 contentValues.put(**"Title"**, **ET\_noticeTitle**.getText().toString());  
 contentValues.put(**"Content"**, **ET\_noticeContent**.getText().toString());  
 contentValues.put(**"Time"**, **ly\_time**);  
 contentValues.put(**"AdID"**, **adId**);  
 **sqlite** = **new** MySqlite(AdminNoticeAdd.**this**, **"Dormitory.db"**, **null**, 1);  
 **myDb** = **sqlite**.getWritableDatabase();  
 **myDb**.insert(**"Notice"**, **null**, contentValues);  
 Toast.*makeText*(AdminNoticeAdd.**this**, **"发布成功!"**, Toast.***LENGTH\_LONG***).show();  
 Intent intent = **new** Intent(AdminNoticeAdd.**this**, AdminNotice.**class**);  
 intent.putExtra(**"adId"**, **adId**);  
 startActivity(intent);  
 }  
 }  
});

3.7：公告搜索（管理员和学生）

3.7.1：StudentNotice和AdminNotice

3.7.1.1布局

采用约束布局，布局头有一个EditText搜索框，回车返回搜索事件，然后采用ExpandableListView用来显示公告，接下来底部是LinearLayout带两个button，两个button用于管理员功能界面的跳转。

3.7.1.2方法

3.7.1.2.1：onCreate()

StudentNotice:

设计思路：打开数据库，通过学生的UserID将相关的RoomID读出来，然后通过RoomID把相关BuildID读出来，因为BuildID和管理员绑定，所以再通过管理员ID把和这栋楼的通知全部读出来赋值。再将这些值放在学生通知的布局文件相关控件里面。

相关代码：

**final** ExpandableListView expandableListView = (ExpandableListView) findViewById(R.id.***expandableList***);  
 Intent intent = getIntent();  
 **studentId** = intent.getStringExtra(**"studentId"**);  
 Cursor cursor = **myDb**.rawQuery(**"select \* from User where UserID = ? "**, **new** String[]{**studentId**});  
 **if** (cursor.moveToFirst()) {  
 **roomNumber** = cursor.getString(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"RoomID"**))));  
 }  
 cursor = **myDb**.rawQuery(**"select \* from Room Where RoomID = ?"**, **new** String[]{**roomNumber**});  
 **if** (cursor.moveToFirst()) {  
 **adId** = cursor.getString(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"BuildID"**))));  
 }  
 cursor = **myDb**.rawQuery(**"select \* from Notice Where AdID = ?"**, **new** String[]{**adId**});  
 **if** (cursor.moveToFirst()) {  
 String title;  
 String time;  
 String adId;  
 String content;  
 **do** {  
 title = cursor.getString(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"Title"**))));  
 time = cursor.getString(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"Time"**))));  
 adId = cursor.getString(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"AdID"**))));  
 content = cursor.getString(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"Content"**))));  
 **notice** = **new** NoticeExpandable(title, time, adId, content);  
 **notices**.add(**notice**);  
 } **while** (cursor.moveToNext());  
 }  
 Collections.*reverse*(**notices**);  
 NoticeExpandableAdapter adapter = **new** NoticeExpandableAdapter(**notices**, **this**);  
 expandableListView.setAdapter(adapter);  
  
 **final** EditText searchBar = (EditText) findViewById(R.id.***searchBar***);  
 searchBar.setImeOptions(EditorInfo.***IME\_ACTION\_SEARCH***);  
 searchBar.setInputType(EditorInfo.***TYPE\_CLASS\_TEXT***);  
 searchBar.setOnEditorActionListener(**new** TextView.OnEditorActionListener() {  
 @Override  
 **public boolean** onEditorAction(TextView textView, **int** actionID, KeyEvent keyEvent) {  
 **if** (actionID == EditorInfo.***IME\_ACTION\_SEND*** || ((keyEvent != **null** && keyEvent.getKeyCode() == KeyEvent.***KEYCODE\_ENTER***) && keyEvent.getAction() == KeyEvent.***ACTION\_DOWN***)) {  
 **notices**.clear();  
 String str = searchBar.getText().toString();  
 Cursor cursor = **myDb**.rawQuery(**"select \* from Notice where Title like ?"**, **new** String[]{**"%"** + str + **"%"**});  
 **int** i = cursor.getColumnIndex(**"AdID"**);  
 **int** j = cursor.getColumnIndex(**"Title"**);  
 **int** o = cursor.getColumnIndex(**"Time"**);  
 **int** k = cursor.getColumnIndex(**"Content"**);  
 **if** (cursor.moveToFirst()) {  
 **do** {  
 String adId = cursor.getString(i);  
 String time = cursor.getString(o);  
 String title = cursor.getString(j);  
 String content = cursor.getString(k);  
 **notice** = **new** NoticeExpandable(title, time, adId, content);  
 **notices**.add(**notice**);  
 } **while** (cursor.moveToNext());  
 }  
 Collections.*reverse*(**notices**);  
 NoticeExpandableAdapter adapter = **new** NoticeExpandableAdapter(**notices**, StudentNotice.**this**);  
 expandableListView.setAdapter(adapter);  
 adapter.notifyDataSetChanged();  
 **return true**;  
 }  
 **return false**;  
  
 }  
 });  
}

AdminNotice：

管理员搜索是通过管理员ID来搜索绑定的公告，每个管理员只能管理和他相关的公告，所以先用数据库搜索语句将等于管理员ID的公告筛选出来，然后通过公告标题模糊查询将公告信息存下来，主要有NID，Title，Time。

相关代码：

**final** EditText searchBar = (EditText) findViewById(R.id.***searchBar***);  
 searchBar.setImeOptions(EditorInfo.***IME\_ACTION\_SEARCH***);  
 searchBar.setInputType(EditorInfo.***TYPE\_CLASS\_TEXT***);  
 searchBar.setOnEditorActionListener(**new** TextView.OnEditorActionListener() {  
 @Override  
 **public boolean** onEditorAction(TextView textView, **int** actionID, KeyEvent keyEvent) {  
 **if** (actionID == EditorInfo.***IME\_ACTION\_SEND*** || ((keyEvent != **null** && keyEvent.getKeyCode() == KeyEvent.***KEYCODE\_ENTER***) && keyEvent.getAction() == KeyEvent.***ACTION\_DOWN***)) {  
 **notices**.clear();  
 String str = searchBar.getText().toString();  
 Cursor cursor = **myDb**.rawQuery(**"select \* from Notice where Title like ?"**, **new** String[]{**"%"** + str + **"%"**});  
 **int** i = cursor.getColumnIndex(**"NID"**);  
 **int** j = cursor.getColumnIndex(**"Title"**);  
 **int** o = cursor.getColumnIndex(**"Time"**);  
 **if** (cursor.moveToFirst()) {  
 **do** {  
 **int** NID = cursor.getInt(i);  
 String time = cursor.getString(o);  
 String title = cursor.getString(j);  
 Log.*d*(**"TTT"**, **"NID"** + NID);  
 **notice** = **new** Notice(NID, title, time, **adId**);  
 **notices**.add(**notice**);  
 } **while** (cursor.moveToNext());  
 }  
 Collections.*reverse*(**notices**);  
 **adapter** = **new** NoticeAdapter(AdminNotice.**this**, R.layout.***notice***, **notices**);  
 **listView**.setAdapter(**adapter**);  
 **adapter**.notifyDataSetChanged();  
 **return true**;  
 }  
 **return false**;  
  
 }  
 });  
  
}  
  
**public void** init() {  
 **listView** = (ListView) findViewById(R.id.***lv\_text\_view***);  
 **notices** = **new** ArrayList<Notice>();  
}  
  
**public void** initDate(List<Notice> notices) {  
 Intent intent = getIntent();  
 String adId = intent.getStringExtra(**"adId"**);  
 *//模拟创建数据* SimpleDateFormat sdf = **new** SimpleDateFormat(**"yyyy-MM-dd HH:mm:ss"**);  
 String ly\_time = sdf.format(**new** java.util.Date());  
 Cursor cursor = **myDb**.rawQuery(**"select \* from Notice where AdID = ?"**, **new** String[]{adId}, **null**);  
 **if** (cursor.moveToFirst()) {  
 **do** {  
 **int** NID = cursor.getInt(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"NID"**))));  
 **time** = cursor.getString(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"Time"**))));  
 **title** = cursor.getString(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"Title"**))));  
 **notice** = **new** Notice(NID, **title**, **time**, adId);  
 notices.add(**notice**);  
 } **while** (cursor.moveToNext());  
 }  
}

设计思路：

3.7.1.2.2：onClick()

StudentNotice：

学生通知为一个点击按钮，该按钮用来跳转学生功能主页面

相关代码：

1：EditText监听器，用来实现学生通过搜索框回车搜索公告

searchBar.setOnEditorActionListener(**new** TextView.OnEditorActionListener() {  
 @Override  
 **public boolean** onEditorAction(TextView textView, **int** actionID, KeyEvent keyEvent) {  
 **if** (actionID == EditorInfo.***IME\_ACTION\_SEND*** || ((keyEvent != **null** && keyEvent.getKeyCode() == KeyEvent.***KEYCODE\_ENTER***) && keyEvent.getAction() == KeyEvent.***ACTION\_DOWN***)) {  
 **notices**.clear();  
 String str = searchBar.getText().toString();  
 Cursor cursor = **myDb**.rawQuery(**"select \* from Notice where Title like ?"**, **new** String[]{**"%"** + str + **"%"**});  
 **int** i = cursor.getColumnIndex(**"AdID"**);  
 **int** j = cursor.getColumnIndex(**"Title"**);  
 **int** o = cursor.getColumnIndex(**"Time"**);  
 **int** k = cursor.getColumnIndex(**"Content"**);  
 **if** (cursor.moveToFirst()) {  
 **do** {  
 String adId = cursor.getString(i);  
 String time = cursor.getString(o);  
 String title = cursor.getString(j);  
 String content = cursor.getString(k);  
 **notice** = **new** NoticeExpandable(title, time, adId, content);  
 **notices**.add(**notice**);  
 } **while** (cursor.moveToNext());  
 }  
 Collections.*reverse*(**notices**);  
 NoticeExpandableAdapter adapter = **new** NoticeExpandableAdapter(**notices**, StudentNotice.**this**);  
 expandableListView.setAdapter(adapter);  
 adapter.notifyDataSetChanged();  
 **return true**;  
 }  
 **return false**;  
  
 }  
});

2：跳转回学生详细信息页面

**public void** navigationMy(View view) {  
 Intent intent = **new** Intent(StudentNotice.**this**, StudentMy.**class**);  
 intent.putExtra(**"studentId"**, **studentId**);  
 startActivity(intent);  
 overridePendingTransition(0, 0);  
}

AdminNotice

有三个点击事件，一个为添加公告事件，用来跳转到添加公告页面。另外两个点击事件用来跳转到除公告界面的其他管理员功能界面

相关代码：

1：跳转到管理员添加公告界面

add.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 Intent intent = **new** Intent(AdminNotice.**this**, AdminNoticeAdd.**class**);  
 intent.putExtra(**"adId"**, **adId**);  
 startActivity(intent);  
  
 }  
});

2：跳转到管理员功能中的主界面

**public void** navigationHome(View view) {  
 Intent intent = **new** Intent(**this**, AdminHome.**class**);  
 intent.putExtra(**"adId"**, **adId**);  
 startActivity(intent);  
 overridePendingTransition(0, 0);  
}  
3：跳转到管理员个人界面  
**public void** navigationMy(View view) {  
 Intent intent = **new** Intent(**this**, AdminMy.**class**);  
 intent.putExtra(**"adId"**, **adId**);  
 startActivity(intent);  
 overridePendingTransition(0, 0);  
}

3.8 宿舍修建与废弃

宿舍的修建主要是添加层数，宿舍数目和床数，并自动分配一个管理员。宿舍楼的废弃是直接对某一栋宿舍进行废弃处理，相应的学生和管理员信息也会被更改。

3.8.1：GeneralAdminBuidingBuild与GeneralAdminBuidingAbandon

3.8.1.1布局

采用约束布局，通过使用Button,EditText,TextView,Spinner的组合实现简单的宿舍楼更换添加界面。

3.8.1.2 方法

onCreate()

设计思路：打开数据库，通过游标moveToLast()方法确定宿舍楼的数量，添加点击事件。以及通过游标读取数据库宿舍信息并添加在Spinner里面显示。

**sqlite** = **new** MySqlite(**this**, **"Dormitory.db"**, **null**, 1);  
 **myDb** = **sqlite**.getWritableDatabase();*//打开数据库myDb*

Spinner spinner = (Spinner) findViewById(R.id.***buildingAbandonSpinner***);

Cursor cursor = **myDb**.rawQuery(**"select \* from Building"**, **null**);  
**if** (cursor.moveToFirst()) {  
 **do** {  
 **buildingNumber** = cursor.getString(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"BuildID"**))));  
 **buildingNumbers**.add(**buildingNumber** + **"号宿舍楼"**);  
 } **while** (cursor.moveToNext());//为 spinner视图绑定宿舍楼数据  
}

ArrayAdapter<String> spinnerAdapter = **new** ArrayAdapter<String>(**this**, android.R.layout.***simple\_spinner\_dropdown\_item***, **buildingNumbers**);//绑定适配器  
spinnerAdapter.setDropDownViewResource(android.R.layout.***simple\_spinner\_dropdown\_item***);  
spinner.setAdapter(spinnerAdapter);  
spinner.setSelection(-1, **true**);  
spinner.setOnItemSelectedListener(**new** AdapterView.OnItemSelectedListener() {  
 @Override  
 **public void** onItemSelected(AdapterView<?> parent, View view, **int** position, **long** id) {  
 **int** length = parent.getSelectedItem().toString().length();  
 **buildingNumber** = parent.getSelectedItem().toString().substring(0, length - 4);//获取宿舍楼号  
 }  
  
 @Override  
 **public void** onNothingSelected(AdapterView<?> parent) {  
  
 }  
});

onClick()

设计思路：通过数据库的execSQL实现删除选中的宿舍楼，并通过Toast显示信息。建楼的时候，还需要判断楼层数目能否整除宿舍数目，得到每层楼宿舍数目。

Button监听事件：用来检测用户输入信息，实现宿舍楼的删除功能

Button ok = (Button) findViewById(R.id.***ok***);  
ok.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 String sqlDelete = **"delete from Building where BuildID = "** + **buildingNumber**; *//building的删除* **myDb**.execSQL(sqlDelete);  
 sqlDelete = **"delete from Admin where BuildID = "** + **buildingNumber**; *//admin删除* **myDb**.execSQL(sqlDelete);  
 Cursor cursor1 = **myDb**.rawQuery(**"select \* from Room where BuildID = ?"**, **new** String[]{**buildingNumber**});  
 **if** (cursor1.moveToFirst()) {  
 **int** roomNumber;  
 String sqliteUpdate;  
 **do** {  
 roomNumber = cursor1.getInt(Integer.*parseInt*(String.*valueOf*(cursor1.getColumnIndex(**"RoomID"**))));  
 **myDb**.delete(**"Room"**, **"RoomID=?"**, **new** String[]{Integer.*toString*(roomNumber)}); *//room删除* **myDb**.delete(**"Bed"**, **"RoomID=?"**, **new** String[]{Integer.*toString*(roomNumber)}); *//bed删除* sqliteUpdate = **"UPDATE User SET RoomID = NULL WHERE RoomID = "** + roomNumber; *//user更新* **myDb**.execSQL(sqliteUpdate);  
 } **while** (cursor1.moveToNext());  
 }  
 Toast.makeText(GeneralAdminBuildingAbandon.**this**, **"删除成功！"**, Toast.***LENGTH\_SHORT***).show();  
 finish();  
 }

Button监听事件： 响应用户输入，实现宿舍的建立与自动分配管理员

Button ok = (Button) findViewById(R.id.***ok***);  
 ok.setOnClickListener(**new** View.OnClickListener()

{  
 @Override  
 **public void** onClick(View v) {  
 String floorSum = floor.getText().toString();  
 String roomSum = room.getText().toString();  
 String bedSum = bed.getText().toString();  
 **if** (Integer.*parseInt*(roomSum) % Integer.*parseInt*(floorSum) != 0) {  
 Toast.makeText(GeneralAdminBuildingBuild.**this**, **"楼层数应整除宿舍数！"**, Toast.***LENGTH\_SHORT***).show();  
 **return**;  
 }  
 Log.*d*(**"SSS"**, **"sdawd "** + **buildingNumber**);  
 ContentValues contentValues = **new** ContentValues(); *//building* contentValues.put(**"BuildID"**, Integer.*parseInt*(**buildingNumber**));  
 contentValues.put(**"Floor"**, Integer.*parseInt*(floorSum));  
 contentValues.put(**"Rsum"**, Integer.*parseInt*(roomSum));  
 **myDb**.insert(**"Building"**, **null**, contentValues);  
  
 **int** roomPerFloor = Integer.*parseInt*(roomSum) / Integer.*parseInt*(floorSum); *//room* String roomNumber = **null**;  
 **for** (**int** i = 1; i <= Integer.*parseInt*(floorSum); i++) {  
 **for** (**int** j = 1; j <= roomPerFloor; j++) {//楼层和寝室动态分配  
 roomNumber = **buildingNumber** + Integer.*toString*(i \* 100 + j);  
 ContentValues contentValues1 = **new** ContentValues();  
 contentValues1.put(**"RoomID"**, roomNumber);  
 contentValues1.put(**"Total"**, bedSum);  
 contentValues1.put(**"Residual"**, bedSum);  
 contentValues1.put(**"WAE\_Fees"**, 0);  
 contentValues1.put(**"BuildID"**, **buildingNumber**);  
 **myDb**.insert(**"Room"**, **null**, contentValues1);  
 **for** (**int** k = 1; k < Integer.*parseInt*(bedSum); k++) { *//bed和宿舍数目动态匹配* ContentValues contentValues2 = **new** ContentValues();  
 contentValues2.put(**"BedID"**, k);  
 contentValues2.put(**"RoomID"**, roomNumber);  
 **myDb**.insert(**"Bed"**, **null**, contentValues2);  
  
 }  
 }  
 }  
  
 ContentValues contentValues3 = **new** ContentValues();  
 contentValues3.put(**"AdID"**, Integer.*parseInt*(**buildingNumber**));  
 contentValues3.put(**"PassWord"**, **buildingNumber** + **buildingNumber** + **buildingNumber**);  
 contentValues3.put(**"BuildID"**, Integer.*parseInt*(**buildingNumber**));  
 **myDb**.insert(**"Admin"**, **null**, contentValues3);//分配管理员  
 Toast.makeText(GeneralAdminBuildingBuild.**this**, **"修建成功"**, Toast.***LENGTH\_SHORT***).show();  
 finish();  
 }  
  
 });  
}

3.9 学生入住寝室的分配与删除

管理员点击相应的空宿舍，输入学生的id，为学生分配宿舍，删除就是点击对应的宿舍，选择对应的学生床号，进行删除。

3.9.1 BedAdapter

3.9.1.1布局 采用约束布局，通过TextView,Imageview,EditText的组合实现简单的宿舍更换添加界面。

3.9.1.2 方法

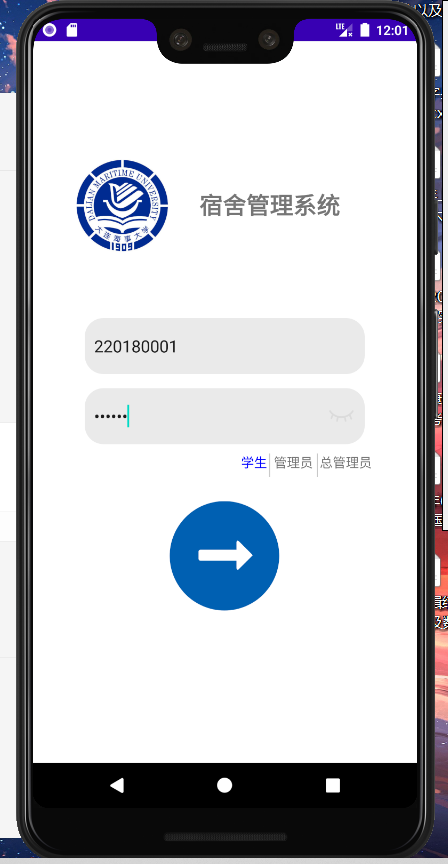
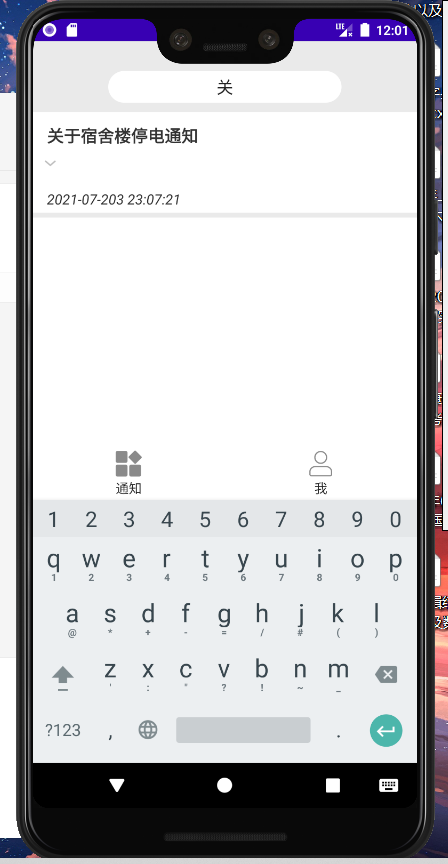
onClick(),打开数据库，运用游标通过RoomID查找用户ID，然后通过execSQL删除对应的用户的床位信息，入住是通过用户ID结合当前的RoomID更新用户表的床位信息。

Button监听事件：根据用户输入查找对应的床位信息，并删除

Button quit = (Button) view.findViewById(R.id.***quit***);  
 quit.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **sqlite** = **new** MySqlite(**context**, **"Dormitory.db"**, **null**, 1);  
 **myDb** = **sqlite**.getWritableDatabase();*//打开数据库myDb* ContentValues contentValues = **new** ContentValues();  
 **int** bedNumber = bed.getBedNumber();  
 **int** residual;  
 String studentId = student.getStudentId();  
 String roomNumber = **null**;  
 Cursor cursor = **myDb**.rawQuery(**"select RoomID from User where UserID=?"**, **new** String[]{studentId});//根据用户ID 得到RoomID  
 **if** (cursor.moveToFirst()) {  
 roomNumber = cursor.getString(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"RoomID"**))));  
 String sqliteUpdate = **"UPDATE User SET RoomID = NULL WHERE UserID = "** + studentId;  
 **myDb**.execSQL(sqliteUpdate);//更新对应的用户的RoomID  
 }  
 cursor = **myDb**.rawQuery(**"select \* from Room where RoomID=?"**, **new** String[]{roomNumber});  
 **if** (cursor.moveToFirst()) {  
 residual = Integer.*parseInt*(cursor.getString(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"Residual"**)))));  
 residual++;  
 String sqliteUpdate = **"UPDATE Room SET Residual ="** + Integer.*toString*(residual) + **" WHERE RoomID = "** + roomNumber;  
 **myDb**.execSQL(sqliteUpdate);  
 }  
 String sqliteUpdate = **"UPDATE Bed SET UserID = NULL WHERE RoomID = "** + roomNumber + **" and BedID = "** + Integer.*toString*(bedNumber);//更新相应宿舍对应床位的用户信息  
 **myDb**.execSQL(sqliteUpdate);  
 Toast.*makeText*(**context**, **"退出成功！"**, Toast.***LENGTH\_SHORT***).show();  
  
 }  
 });  
  
 } **else** {  
 view = LayoutInflater.*from*(**context**).inflate(R.layout.***bed\_null\_child***, parent, **false**);  
 学生入住监听事件：根据用户输入信息实现宿舍的分配，以及相应表的更新

Button roomIn = (Button) view.findViewById(R.id.***roomIn***);  
 roomIn.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **sqlite** = **new** MySqlite(**context**, **"Dormitory.db"**, **null**, 1);  
 **myDb** = **sqlite**.getWritableDatabase();*//打开数据库myDb* EditText editTextStudentId = (EditText) view.findViewById(R.id.***studentId***);  
 String studentId = editTextStudentId.getText().toString();  
 String roomNumber = bed.getRoomNumber();  
 **int** bedNumber = bed.getBedNumber();  
 String sqliteCount = **"SELECT \* FROM User WHERE UserID="** + studentId;  
 Cursor cursor = **myDb**.rawQuery(sqliteCount, **null**);  
 cursor.moveToFirst();  
 **long** count = cursor.getLong(0);  
 **if** (count > 0) {  
 String sqliteUpdate = **"UPDATE User SET RoomID = "** + roomNumber + **" WHERE UserID ="** + studentId;//更新用户的roomNumber/  
 **myDb**.execSQL(sqliteUpdate);  
 sqliteUpdate = **"UPDATE Bed SET UserID = "** + studentId + **" WHERE BedID ="** + Integer.*toString*(bedNumber) + **" and RoomID ="** + roomNumber;  
 **myDb**.execSQL(sqliteUpdate);//为bed更新对应的userid  
 cursor = **myDb**.rawQuery(**"select \* from Room where RoomID=?"**, **new** String[]{roomNumber});  
 **if** (cursor.moveToFirst()) {  
 **int** residual = cursor.getInt(Integer.*parseInt*(String.*valueOf*(cursor.getColumnIndex(**"Residual"**))));  
 residual--;  
 sqliteUpdate = **"UPDATE Room SET Residual = "** + Integer.*toString*(residual) + **" WHERE RoomID = "** + roomNumber;  
 **myDb**.execSQL(sqliteUpdate);  
 Toast.*makeText*(**context**, **"入住成功！"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 } **else** {  
 Toast.*makeText*(**context**, **"学号错误！"**, Toast.***LENGTH\_SHORT***).show();  
 }  
  
 }  
 });  
 }  
 **return** view;  
  
}

**4.运行测试**

****

4.1登录页面 4.2学生公告模糊搜索（1）

****

4.3学生查看公告 4.4学生公告模糊搜索（2）

****

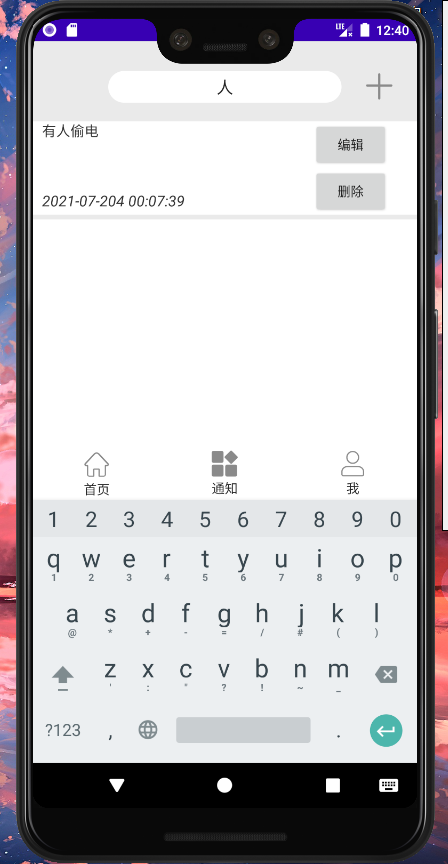
4.5学生个人页面 4.6学生更改密码

****

4.7管理员查看房间详情 4.8学生查看公告

****

4.9管理员安排入住 4.10管理员安排学生退出

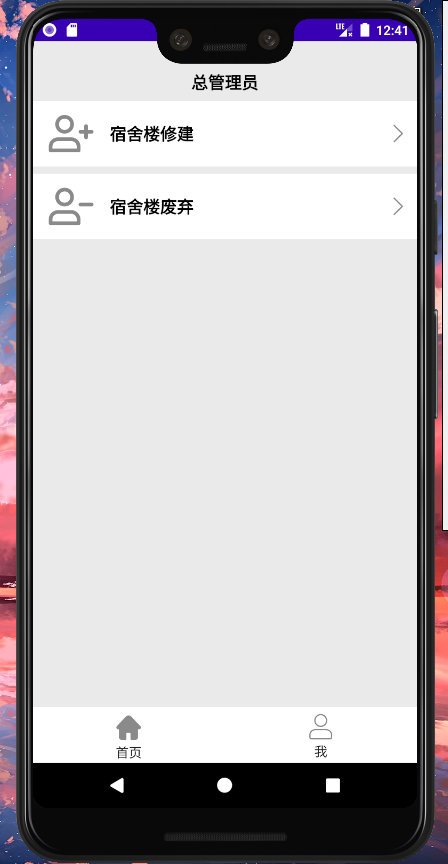




4.11管理员公告删除 4.12管理员公告搜索

****

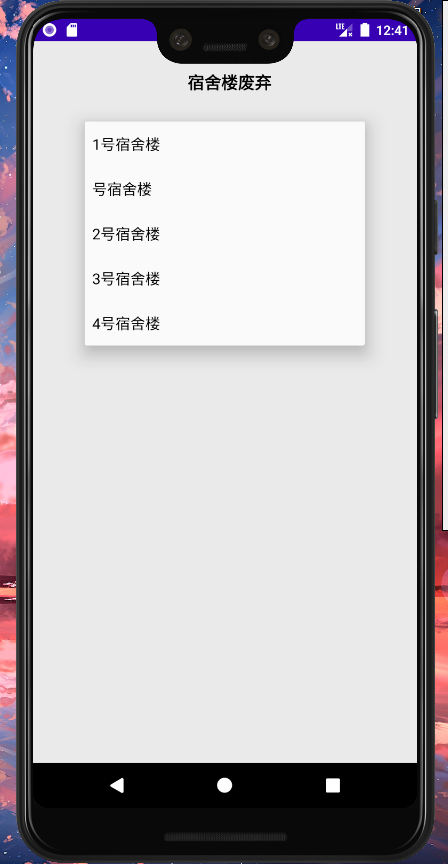
4.13管理员公告更新 4.14管理员公告发布



4.15管理员个人页面 4.16总管理员首页



4.17总管理员个人页面 4.18总管理员增加宿舍楼并分配宿舍楼内属性



4.19宿舍楼废弃 4.20选择废弃项

**5.总结与展望**

5.1遇到的问题

①在使用Android studio的时候，发现不能编译东西，说我丢失工具平台（Platform-tools），但是我确确实实是安装了，也试过重装都没办法解决。

解决方法：通过自己摸索把build.gradle里面的最大SDK版本改成当前虚拟机的版本，buildtoolsVersion也要改成能安装的虚拟机版本，就能运行了

②在进行Notice功能ListView单项Item进行修改的时候，无法定位点击按钮所返回的position值

解决方式：我们通过每次将Notice里面的主键NID封装在一个notice类里面，然后点击每个公告的点击时间，通过每个公告独有的NID进行定位得到position，就能够定位需要修改的公告。

③在constraint layout中，如何在两个组件中间，水平或垂直得插入一个组件并铺满整个中间的空间

解决方法：通过设置weight为1，并把水平或垂直的约束都设好。

④如何取消activity跳转时的动画

解决方法：通过在stratActivity 和 finishi 后写上overridePendingTransition(0, 0);

⑤如何将activity的action bar不显示

解决方法：通过在style中写一个style 内容是

<style name="AppTheme.NoActionBar">

<item name="actionBarSize">40dp</item>

<item name="windowNoTitle">true</item>

</style>

并在manifes中每个activity的theme都设为这个

⑥每次activity开启时，edittext会自动获取焦点，并出现键盘

解决方法：在activity的onCreate中写入getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT\_INPUT\_STATE\_ALWAYS\_HIDDEN);

⑦edittext想要设置成没有边框的形式，或者四周都有边框的形式

解决方法：在drawable中写入selector对stroke进行设置

⑧边框想要设置成只显示一部分

解决方法：在drawable中写入lier-list，通过堆叠的背景的形式，表现出边框

⑨数据库操作时报错 Attempt to invoke virtual method “xxx” on a null object reference

解决方法：写监听器时，在方法里没连接数据库

⑩ListView 写在ScrollView中会只在一行半的高度中显示ListView

解决方法：删掉ScrollView，ListView支持实现ScrollView功能

⑪ ：edittext输入完后如何给回车键添加搜索、进入等功能？

解决方法：

5.2课程收获与展望

为期两周的实训课结束了。在此过程中，我们小组四人都获益匪浅。按照老师教的知识一步步的做下来，虽然过程中遇到了许多问题，但是通过这些疑问，让自己对操作步骤更加记忆深刻，操作越来越熟练。而且，创建成功后，看着自己建立的第一个app,即使功能照预期删除了很多，但还是很兴奋。 通过这两周的开发使我们学习了很多新的内容，了解了相应手机app的制作。使我深深了解到我们所应用的智能手机，不是那么容易就做出来的。 我们学习了在安卓的开发环境下创建一个简单的APP，完成建立主界面、菜单、两个页面的跳转以及注册、登录等功能。我们主要应用了Android studio软件及Java语言完成了代码设计与运行。本次课程为以后工作及学习提供了宝贵的实践经验，希望能在安卓开发的道路上越走越远，为中国软件生态提供自己的绵薄之力。