中山大学数据科学与计算机学院

本科生项目报告

课程名称：移动应用开发　　任课教师：郑贵锋

一、 分工

组员	学号	负责内容	贡献百分比
		前端界面设计实现、美化，用户手册	25%
		总体设计、Webservice、前后端结合、视频	25%
		数据库搭建、数据库接口设计、视频	25%
		前端界面设计实现、美化，用户手册	25%

二、 个人负责部分

项目中我负责的是总体设计和提供基础接口、服务器的搭建和环境的配置、数据库和前端的对接(数据获取和提交)，以及部分前端工作和最终的整合，视频的制作。

三、 负责部分说明

1. 总体设计

我主要是进行界面的初始，设计提出界面要求，同时给出基础的模块，接口类，同时和队友根据需求设计数据库。

- **基础模块设计**
  - 时间获取模块

时间的获取使用 Calendar 获取当前手机的年、月、日、周，此时为了方便后续的使用将获取到的数据时间以 String 的方式返回，使用 calendar.get()，获取对应的年、月、日、周。

Calendar.*YEAR*	Calendar.*MONTH*	Calendar.*DATE*	Calendar.*DAY_OF_WEEK*
年	月	日	周

此处需要额外对周进行判断，根据不同的时手机的时间设定方式，需要对不同的周进行适配，存在每周第一天是周日和每周第一天是周一的两种情况所欲需要进行判断，此处展示周的获取，其他部分类似。

```
/**
 * 获取星期
 * @return "1".....".7" 显示时转换为中文
 */
public static String GetWeek(){
    calendar=Calendar.getInstance();
    boolean isFirstSunday = (calendar.getFirstDayOfWeek() == Calendar.SUNDAY);//判断第一日安
    int week=calendar.get(Calendar.DAY_OF_WEEK);
    if(isFirstSunday){
        week=week-1;
```

```
        if(week==0)week=7;
    }
    return String.valueOf(week);
}
```

- 登录信息存储模块

登录信息的存储使用 SharedPreferences 进行存储，先存储学号/教工号，过对应学号/学工号取出对应的信息。

```
/**
 * 获取存储的信息
 * @param context
 * @return 储存的学号/教工号
 */
public static String getUserId(Context context){
    String id;
    SharedPreferences preferences=context.getSharedPreferences("UserId",MODE_PRIVATE);
    id=preferences.getString("UserId",null);
    if(id!=null&&!id.equals(""))return DES.decryptDES(id);
    else return "";
}

/**
 * 储存 学号/教工号
 * @param context
 * @param id  学号/教工号
 */
public static void setUserId(Context context,String id){
    SharedPreferences preferences=context.getSharedPreferences("UserId", MODE_PRIVATE);
    SharedPreferences.Editor editor = preferences.edit();
    editor.putString("UserId", DES.encryptDES(id));
    editor.commit();
}
```

上述储存的学号/教工号，使用 DES 的方式进行加密。

通过存储的学号/教工号，调用对应的接口直接取出对应的学生 or 教师的信息，此处展示储存和取出的学生信息的接口，存取教师信息的接口类似。

```
/**
 * 设置学生信息
 * @param context
 * @param student 传入学生类
 */
public static void setStudentInfo(Context context, Student student){
    SharedPreferences preferences=context.getSharedPreferences("UserInfo", MODE_PRIVATE);
    SharedPreferences.Editor editor = preferences.edit();
    editor.putString("id",DES.encryptDES(String.valueOf(student.getId())));
    editor.putString("num", DES.encryptDES(student.getNum()));
    editor.putString("passwd", DES.encryptDES(student.getPasswd()));
    editor.putString("avatar", student.getAvatar());
    editor.putString("realname", student.getName());
    editor.putString("nickname", student.getNickname());
    editor.putString("college", student.getCollege());
    editor.putString("info", student.getInfo());
    editor.putString("sex", student.getSex());
    editor.commit();
    setUserId(context,student.getNum());
}

/**
 * 取出学生信息
 * @param context
 * @return 学生类
 */
public static Student getStudentInfo(Context context){
    String num=null;
    String id,passwd,avatar,realname,nickname,college,info,sex;
    SharedPreferences preferences=context.getSharedPreferences("UserInfo",MODE_PRIVATE);
    num=preferences.getString("num",null);
    if(num!=null&&!num.equals("")){
        num=DES.decryptDES(num);
```

```
        id=preferences.getString("id",null);
        id=DES.decryptDES(id);
        passwd=preferences.getString("passwd",null);
        passwd=DES.decryptDES(passwd);
        avatar=preferences.getString("avatar",null);
        realname=preferences.getString("realname",null);
        nickname=preferences.getString("nickname",null);
        college=preferences.getString("college",null);
        info=preferences.getString("info",null);
        sex=preferences.getString("sex",null);
        Student student =new Student(realname,nickname,passwd,avatar,sex,college,info,num);
        student.setId(Integer.valueOf(id));
        return student;
    }
    else return null;
}
```

- 数据加密模块

数据加密使用 DES 加密，单纯加密存储的账号和密码信息，DES 加密技术较为简单，但是对于我们存储登录信息也是已经足够，因为这里我们只是需要稍微保护一下本地账户数据，当然有权限打开应用就可以获取全部数据，这里的数据在每次重新登录和修改之后都会和服务器进行同步。

```
/**
 *
 * @param encryptString 明文
 * @return 加密后的密文
 */
public static String encryptDES(String encryptString){
    try {
        //实例化 IvParameterSpec 对象，使用指定的初始化向量
        IvParameterSpec zeroIv=new IvParameterSpec(iv);
        //实例化 SecretKeySpec，根据传入的密钥获得字节数组来构造 SecretKeySpec
        SecretKeySpec key =new SecretKeySpec(encryptKey.getBytes(),"DES");
        //创建密码器
        Cipher cipher=Cipher.getInstance("DES/CBC/PKCS5Padding");
        //用密钥初始化 Cipher 对象
        cipher.init(Cipher.ENCRYPT_MODE,key,zeroIv);
        //执行加密操作
        byte[]encryptedData=cipher.doFinal(encryptString.getBytes());
        return Base64.encodeToString(encryptedData,0);
    }
}
/**
 * 解密的过程与加密的过程大致相同
 * @param decryptString 密文
 * @return  返回明文
 */
public static String decryptDES(String decryptString){
    try {
        //先使用 Base64 解密
        byte[]byteMi=Base64.decode(decryptString,0);
        //实例化 IvParameterSpec 对象使用指定的初始化向量
        IvParameterSpec zeroIv=new IvParameterSpec(iv);
        //实例化 SecretKeySpec，根据传入的密钥获得字节数组来构造 SecretKeySpec,
        SecretKeySpec key=new SecretKeySpec(encryptKey.getBytes(),"DES");
        //创建密码器
        Cipher cipher=Cipher.getInstance("DES/CBC/PKCS5Padding");
        //用密钥初始化 Cipher 对象,上面是加密，这是解密模式
        cipher.init(Cipher.DECRYPT_MODE,key,zeroIv);
        //获取解密后的数据
        byte [] decryptedData=cipher.doFinal(byteMi);
        return new String(decryptedData);
    }
}
```

- 网络判断模块

应用中我们需要进行联网获取数据库，所以需要进行网络状态的检查。这里我们使用 ConnectivityManager 和 NetworkInfo，检测网络类型，然后根据网络类型产生对应的结果返回，具体实现不在展示。

- 数据上传模块

应用中我们需要对用户的头像，评论进行上传，设计中也有作业的上传，但是因为服务器带宽不足和浏览方式问题取消。

上传功能使用 okhttp，使用 POST 的方法进行文件的上传配合服务器上的 Websevice 程序，进行接收上传图片并存储，实现部分如下：

```
public static String uploadImage(String imagename) {//将图片发送到服务器
    final String url = "https://chonor.cn/Android/Avatar/index.php";
    File file = new File( Environment.getExternalStorageDirectory(), imagename);//获取路径
    RequestBody fileBody = RequestBody.create(MediaType.parse("application/octet-stream"), file);
    RequestBody requestBody = new MultipartBody.Builder()//构建 POST
            .setType(MultipartBody.FORM)
            .addFormDataPart("image1", imagename, fileBody)
            .build();
    Request request = new Request.Builder()
            .url(url)
            .post(requestBody)
            .build();
    final okhttp3.OkHttpClient.Builder httpBuilder = new OkHttpClient.Builder();
    OkHttpClient okHttpClient = httpBuilder
            //设置超时
            .connectTimeout(10, TimeUnit.SECONDS)
            .writeTimeout(15, TimeUnit.SECONDS)
            .build();
    okHttpClient.newCall(request).enqueue(new Callback() {
        @Override
        public void onFailure(Call call, IOException e) {
            result="-1";
            Log.e("aa", "uploadMultiFile() e=" + e);
        }
        @Override
        public void onResponse(Call call, Response response) throws IOException {
            Log.i("bb", "uploadMultiFile() response=" + response.body().string());
            result="1";
        }
    });
    return result;
}
```

- **接口类设计**
  - 学生类

储存学生信息，此处的学生分数，只在进入课程信息时，通过查询获取分数，然后将分数展示，其余的都为 get 和 set 接口。

```
public class Student {
    private String name;//名字
    private String passwd;//密码
    private String nickname;//昵称
    private String avatar;  //头像
    private String sex;     //性别
    private String college;//院系
    private String info;    //简介
    private Integer id;      //sid
    private String num;     //学号
    private Integer score; //分数用于当前课程独立存储
}
```

  - 教师类

储存教师信息提供 get 和 set 接口。

```
public class Teacher {
    private String name;//名字
    private String passwd;//密码
    private String nickname;//昵称
    private String avatar;//头像地址
    private String sex;//性别
```

```
    private String college;//学院
    private String info;//简介
    private Integer id;//tid
    private String num;//教工号
    private String email;//邮箱
    private String phone;//电话
    private String office;//办公地址
    private String position;//职称
}
```

- 课程类

  课程信息，并通过查询，获取对应的教师的部分信息，提供 get 和 set 接口。

```
public class Course {
    private String name; //课程名
    private String week;//星期 1-7 使用时请转中文
    private String time;// 1-5  节数
    private String pos;//地点
    private String college;//学院
    private String info;//课程简介
    private String hour;//学时
    private String credit;//学分
    private String tname;//教师名字
    private Integer cid;//课程 id
    private Integer tid;//教师 id
    private Integer choose;//是否选课
}
```

- 通知类

  储存通知，通过查询获取的课程信息，提供 get 和 set 接口。

```
public class Notice {
    private Integer nid;//通知 id
    private Integer cid;//对应的 cid
    private String cname;//课程名称
    private String title;//标题
    private String info;//信息
    private String starttime;//通知发布时间
    private String endtime;//通知失效时间
}
```

- 评论类

  储存评论，通过学号/教工号查询出评论者信息，同时在评论的赞、踩、举报表中，获取对应的信息，提供 get 和 set 接口。

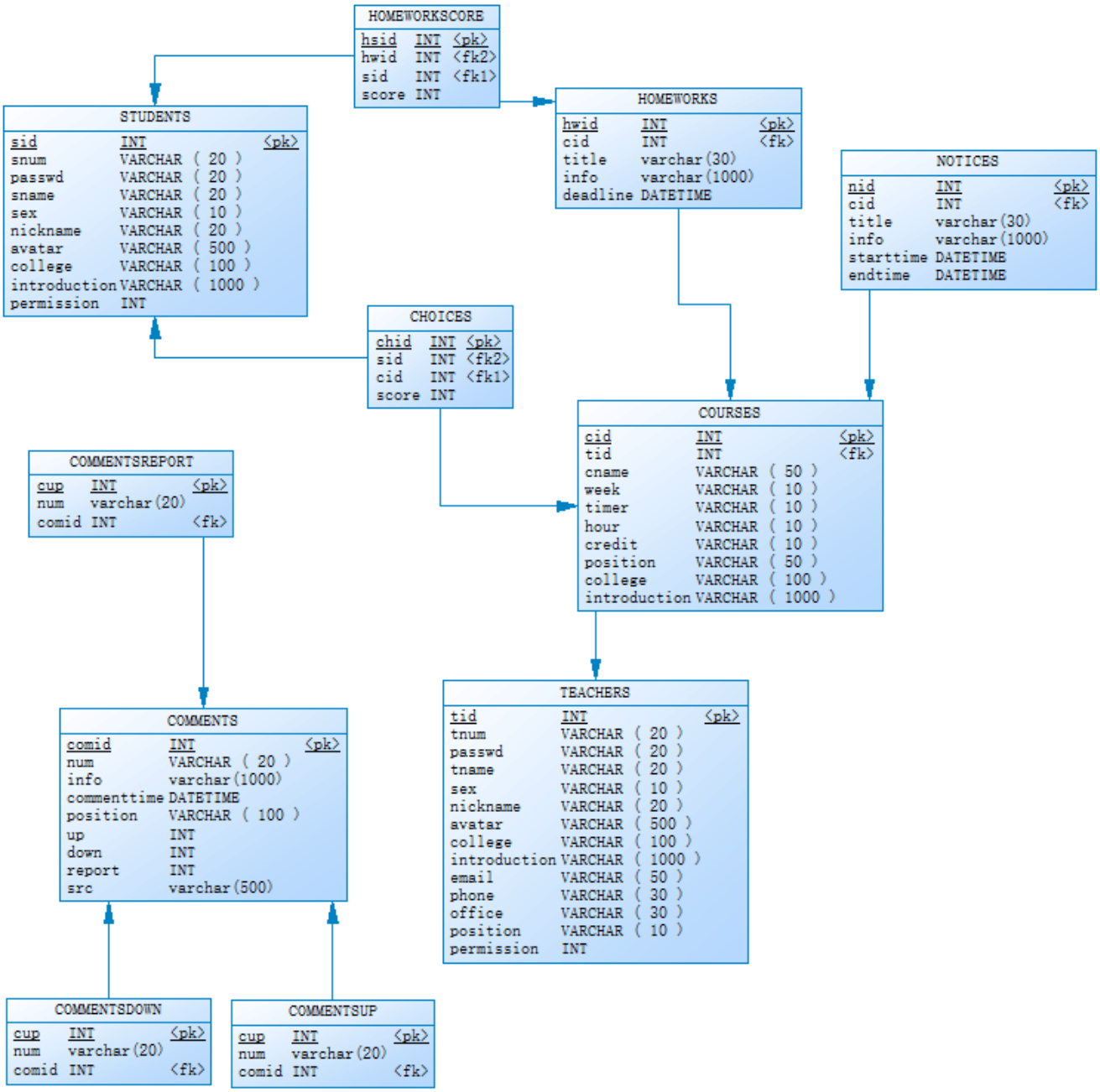
```
public class Comment {
    private Integer cid;//评论 id
    private String num;//评论者的学号/教工号
    private String info;//具体内容
    private String time;//时间  这是用排序的,  使用年/月/日 GetTime 里
    private String name;//评论者名字
    private String college;//学院
    private String pos;//这个你可以不用填
    private Integer up;//赞
    private Integer down;//踩
    private Integer report;//举报
    private String src;//图片
    private String avatar;//评论者头像
}
```

- 作业类

  储存作业信息，提供 get 和 set 接口。

```
public class Homework {
    private Integer cid;//课程 id
    private Integer hwid;//作业 id
    private String title;//标题
    private String info;//信息
    private String ddl;//ddl
    private Integer score;//分数
```

- **数据库 E-R 图**



2. 服务器的搭建、配置

- **服务端环境的选择**

  后端使用腾讯云，使用 Ubuntu，搭建 LAMP 环境(linux+apache+mysql+php)，环境配置方式此处就不说明了。

- **PHP Webservice**

  - 通过 API 访问数据库

    应用一开始考虑过使用 API 访问数据库，通过 OkHttp 使用 Http 协议 POST 发送需要查询的内容，查询结果返回 json 数据，但是因为考虑到此时密码是明文，考虑密码的加密问题最终没有使用，并将数据库连接方式改为直连。API 接口保留但不使用。此处展示查询学生信息的 PHP 代码，其余的区别不同的 POST 结构和修改 SQL 语句，代码如下：

```php
<?php
header("content-Type: text/html; charset=utf-8");//字符编码设置
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "android";
 if(!isset($_POST['sid'])){
    die('sid is not define');
}
```

```php
$sid=$_POST['sid'];
if(empty($sid)){
    die('sid is empty');
}
// 创建连接
$conn =new mysqli($servername, $username, $password, $dbname);
// 检测连接
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT * FROM STUDENTS WHERE snum='{$sid}'";
$result = $conn->query($sql);
$arr = array();
// 输出每行数据
while($row = $result->fetch_assoc()) {
    $count=count($row);//不能在循环语句中，由于每次删除 row 数组长度都减小
    for($i=0;$i<$count;$i++){
        unset($row[$i]);//删除冗余数据
    }
    array_push($arr,$row);

}
//print_r($arr);
echo json_encode($arr,JSON_UNESCAPED_UNICODE);//json 编码
$conn->close();
```

[{"sid":"1","snum":"15352008","passwd":"123","sname":"Chonor","sex":"unknow","nickname":"Chonor","avatar":"
https:\/\/chonor.cn\/Android\/Avatar\/img\/15352008.png","college":"sdcs","introduction":"root","permission":"0"}]

- 上传文件后端

同样使用 PHP 搭建，接收客户端传输上来的文件然后存储，然后根据上传结果返回 json 数组在之前的上传接口中判断是否成功，上传文件后端代码如下：

```php
 <?php
  header('Content-type: application/json;charset=utf-8');
if(empty($_FILES)) die('{"status":0,"msg":"错误提交"}');
$dirPath = './img/';//设置文件保存的目录
if(!is_dir($dirPath)){
 //目录不存在则创建目录
 @mkdir($dirPath);
}
$count = count($_FILES);//所有文件数
if($count<1) die('{"status":0,"msg":"错误提交"}');//没有提交的文件
$success = $failure = 0;
foreach($_FILES as $key => $value){
 //循环遍历数据
 $tmp = $value['name'];//获取上传文件名
 $tmpName = $value['tmp_name'];//临时文件路径
 //上传的文件会被保存到 php 临时目录，调用函数将文件复制到指定目录
 if(move_uploaded_file($tmpName,$dirPath.$tmp)){
   $success++;
 }else{
   $failure++;
 }
}
$arr['status'] = 1;
$arr['msg']   = '提交成功';
$arr['success'] = $success;
$arr['failure'] = $failure;
echo json_encode($arr);
 ?>
```

3. 前端后端的对接

对接工作主要就是将两位队友写的界面模块，填充数据，此处依靠后端队友写的数据库接口和前端预留的点击事件接口，进行数据的填充和数据库的更新操作，以及对于前端的页面转跳，作出对应的数据传递和获取处理。

主要就是使用子线程获取 or 更新数据库信息，然后进行界面的更新，此处介绍基础增删改查的使用，其他的使用方式类似。

- **登录信息的获取**
  - 判断是否登录，如果登录判断是学生还是教师，然后根据不同的登录进行相应的课程表和闲聊区信息获取。

```
/**
 * add by chonor
 * 判断是否登录
 */
private void GetLesson(){//获取登录信息和课程
    usernum= SaveUser.getUserId(Main_Tabhost.this);
    if(usernum.equals("")){//未登录
        logined=false;
        Button button = (Button) findViewById(R.id.sign_out);//设置按钮
        button.setText("点击登录");
        findViewById(R.id.add_course).setVisibility(View.GONE);
        GetAllCourse(-1,-1);//获取全部课程
    }
    else {
        logined = true;
        if(usernum.length()==6){//教师
            isTeacher=true;
            teacher=SaveUser.getTeacherInfo(Main_Tabhost.this);
            TextView textView=(TextView)findViewById(R.id.view_my_notice_text);
            textView.setText("查看我发布的通知");
            findViewById(R.id.add_course).setVisibility(View.VISIBLE);
            GetAllCourse(-1,teacher.getId());//获取全部课程
            GetClassTable(-1,teacher.getId());//获取课程表
        }else {//学生
            isTeacher=false;
            student = SaveUser.getStudentInfo(Main_Tabhost.this);
            TextView textView=(TextView)findViewById(R.id.view_my_notice_text);
            textView.setText("查看我的通知");
            //Button addCourse=(Button)findViewById(R.id.add_course);
            findViewById(R.id.add_course).setVisibility(View.GONE);
            GetAllCourse(student.getId(),-1);//获取全部课程
            GetClassTable(student.getId(),-1);//获取课程表
        }
        Button button = (Button) findViewById(R.id.sign_out);
        button.setText("退出登录");
    }
    GetComments(true);//获取闲聊
}
```

未登录　　登录　　登录保持

- 根据登录的信息获获取课程表和所有课程列表，此处初始获取所有课程限定获取在 10 个，保证不会因为数据过多导致传输过慢出现加载延时，余下的部分在下拉时检测是否下拉到底部然后进行回去，课程表在选课，或者退课之后会重新获取；当身份为教师时，添加课程也会重新获取课程表，此部分使用 onActivityResult 实现调用，课表加载如下：

```
/**
 * 获取课程列表
 * @param sid 学号
 * @param tid 教工号
 *            -1 为无
 */
private void GetAllCourse(final int sid,final int tid){
    new Thread(new Runnable() {
        @Override
        public void run() {
            Coursedb coursedb=new Coursedb();
            Message msg = Message.obtain();
            msg.what=0;
            try{
                AllClass=coursedb.selectAll(0,10,sid,tid); //限定 10 个
                msg.what=2;
                coursedb.close();
            } catch (Exception e){
            }
            Handler_Course.sendMessage(msg);
        }
    }).start();
}
/**
 * 获取课表
 * @param sid 学号
 * @param tid 教工号
 */
private void GetClassTable(final int sid,final  int tid){
    new Thread(new Runnable() {
        @Override
        public void run() {
            Coursedb coursedb=new Coursedb();
            Message msg = Message.obtain();
            Noticedb noticedb=new Noticedb();
            Homeworkdb homeworkdb=new Homeworkdb();
            msg.what=0;
            try{
```

```
                if(sid!=-1){//学生 额外获取通知和作业情况
                    String date=GetTime.GetYeat()+"/"+GetTime.GetMonth()+"/"+GetTime.GetDay();//今日日期
                    ClassTable=coursedb.selectBysid(sid);
                    CntHW=homeworkdb.CountToday(sid,date);
                    CntNot=noticedb.CountToday(sid,date);
                }
                else ClassTable=coursedb.selectBytid(tid);
                msg.what=1;
                coursedb.close();
            } catch (Exception e){
            }
            Handler_Course.sendMessage(msg);
        }
    }).start();
}
/**
 * 下拉时和上拉时加载课程
 * @param sizes
 */
private void ReloadCourse(final int sizes){
    new Thread(new Runnable() {
        @Override
        public void run() {
            Coursedb coursedb = new Coursedb();
            Message msg = Message.obtain();
            msg.what = 0;
            try {
                if(sizes==-1) { //下拉加载全部
                    if (logined) {
                        if (isTeacher)
                            AllClass = coursedb.selectAll(0, 10, -1, teacher.getId());
                        else AllClass = coursedb.selectAll(0, 10, student.getId(), -1);
                    } else
                        AllClass = coursedb.selectAll(0, 10, -1, -1);
                    msg.what = 2;
                }else { //上拉加载后续 10 条
                    if (logined) {
                        if (isTeacher)
                            AllClass = coursedb.selectAll(sizes, 10, -1, teacher.getId());
                        else
                            AllClass = coursedb.selectAll(sizes, 10, student.getId(), -1);
                    } else {
                        AllClass = coursedb.selectAll(sizes, 10, -1, -1);
                    }
                    msg.what = 3;
                }
                coursedb.close();
            } catch (Exception e) {
            }
            Handler_Course.sendMessage(msg);
        }
    }).start();
}
```

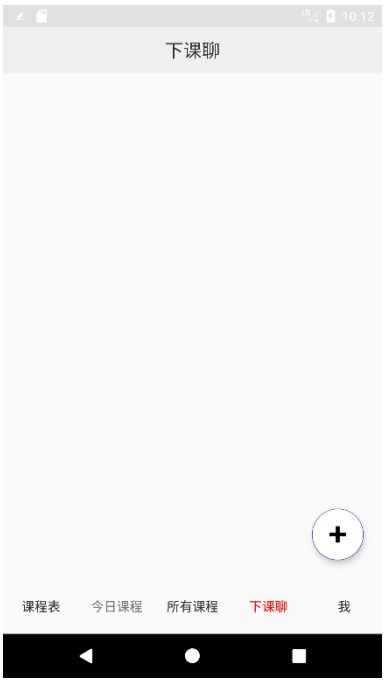
- 使用 Handler 更新数据列表，根据子线程传递的不同信息，作出对应的处理，然后调用队友写的界面更新函数进行界面更新。

```
/**
 * 加载课程表
 */
private android.os.Handler Handler_Course = new Handler() {
    public void handleMessage(Message msg) {
        if (msg.what == 0) {
            Toast.makeText(Main_Tabhost.this, "获取课表失败请检查网络", Toast.LENGTH_SHORT).show();
        } else if (msg.what == 1) { //登录后加载
            CheckChoose();
            if(logins&&!isTeacher){
                broad_init();
                logins=false;
            }
            Filling_ListClassToday();
```

```
            Update_ViewTodayClass();
            Update_Class_Table();
        }
        else if(msg.what==2) {//下拉刷新课表
            if(!isTeacher&&logined)CheckChoose(); //学生额外检测是否选课
            Update_ViewClassList();
        }
        else if(msg.what==3){//加载课表
            for(int i=0;i<Classtmp.size();i++)
                AllClass.add(Classtmp.get(i));
            if(!isTeacher&&logined)CheckChoose();//学生额外检测是否选课
            Update_ViewClassList();
        }
        else if(msg.what==4){//搜索
            issearch=true;
            if(!isTeacher&&logined)CheckChoose();
            Update_ViewClassList();
        }
        loadFlag=false; //避免重复加载标志
    }
};
/**
 * 对照课表查询是否选课
 */
private void CheckChoose(){
    for(int i=0;i<AllClass.size();i++){
        for(int j=0;j<ClassTable.size();j++){
            if(AllClass.get(i).getCid()==ClassTable.get(j).getCid())
                AllClass.get(i).setChoose(1);
        }
    }
}
```

- 效果展示

所有课程加载



闲聊加载中

闲聊加载后

- **注册和编辑个人信息的修改数据库**
  - 注册时插入数据库，将队友界面传入的判断好的信息，进行加载到对应类中的操作，如果出现头像，进行头像的上传，然后更新头像信息，并插入数据库。此处展示注册学生的代码，教师的注册类似不在展示。

```
/**
 * add by chonor
 * to init
 */
if(choose_type==0 &&sid.length()==8) {//判断注册的账号类型
```

```
    final Student student =new Student();//新建学生类
    if(photo!=null) {//先上传照片
        uploadPic(sid);
        student.setAvatar("https://chonor.cn/Android/Avatar/img/" + sid + ".png");
    }//设置信息
    student.setName(real_name);
    student.setNum(sid);
    student.setNickname(name);
    student.setPasswd(first_pass);
    if (sex == 1) student.setSex("male");
    else if (sex == 2) student.setSex("female");
    if (department.length() != 0)
        student.setCollege(department);
    new Thread(new Runnable() {
        @Override
        public void run() { //插入数据库
            Studentdb studentdb = new Studentdb();
            Message msg = Message.obtain();
            msg.what = 0;
            try {
                ResultSet rs = studentdb.queryByNum(student.getNum());
                if (rs.next()) {//学号已存在
                    msg.what = 2;
                } else {
                    try {//插入
                        studentdb.insert(student);
                        msg.what = 1;
                    } catch (Exception e) {
                    }
                }
                rs.close();
                studentdb.close();
            } catch (Exception e) {
            }
            Handler.sendMessage(msg);
        }
    }).start();
    //返回登录界面
}
```

- 通过 Handler 判断是否成功插入，并做出相应提示，成功插入后返回主界面。

```
/**
 * add by chonor
 * to get info
 */
private Handler Handler = new Handler() {
    public void handleMessage(Message msg) {
        if (msg.what == 1) {
            Toast.makeText(RegisterActivity.this, "注册成功,请登录", Toast.LENGTH_SHORT).show();
            setResult(100);
            finish();
        }
        else if(msg.what==2){
            Toast.makeText(RegisterActivity.this, "学号/教工号已被注册", Toast.LENGTH_SHORT).show();
        }
        else {
            Toast.makeText(RegisterActivity.this, "连接服务器出错", Toast.LENGTH_SHORT).show();
        }
    }
};
```

- 个人信息的编辑，此处展示更新教师信息。先是是获取前端的修改数据，然后更新整个教师类，并同步到数据库，同时更新本地缓存的教师信息。

```
if(isTeacher){//更新教师个人信息
    if(mail.length()==0){//判断邮箱
        TextInputLayout ed_tea_email=(TextInputLayout)findViewById(R.id.ed_teacher_email);
        ed_tea_email.setErrorEnabled(true);
        ed_tea_email.setError("必须填写邮箱，请补充");
    }else {
        if (flag) { // 是否更新过头像
```

```
            uploadPic(teacher.getNum());//上传头像
            teacher.setAvatar("https://chonor.cn/Android/Avatar/img/" + teacher.getNum() + ".png");
        }//更新教师类
        teacher.setNickname(new_name);
        teacher.setCollege(new_department);
        teacher.setInfo(new_introduction);
        teacher.setEmail(mail);
        teacher.setOffice(offic);
        teacher.setPhone(phone);
        teacher.setPosition(pos);
        if (edit_male.getId() == new_sex) teacher.setSex("male");
        else if (edit_female.getId() == new_sex) teacher.setSex("famale");
        else if (edit_secret.getId() == new_sex) teacher.setSex("unknow");
        SaveUser.setTeacherInfo(Edit_information.this, teacher);//更新本地信息
        new Thread(new Runnable() {
            @Override
            public void run() {//更新数据库
                Teacherdb teacherdb = new Teacherdb();
                Message msg = Message.obtain();
                msg.what = 0;
                try {
                    teacherdb.update(teacher);
                    msg.what = 1;
                } catch (Exception e) {
                }

                Handler.sendMessage(msg);
            }
        }).start();
    }
}
```

时候 Handler 对修改结果进行判断并进行提示，如修改成功则返回原主界面。

```
private android.os.Handler Handler = new Handler() {
    public void handleMessage(Message msg) {
        if(msg.what==0){
            Toast.makeText(Edit_information.this, "修改失败请检查网络", Toast.LENGTH_SHORT).show();
        }
        else if(msg.what==1){
            Toast.makeText(Edit_information.this, "修改成功", Toast.LENGTH_SHORT).show();
            setResult(RESULT_OK);
            finish();
        }
    }
};
```

- 效果展示

修改信息效果

信息修改后重新加载

注册效果

- **其他接口实现方式类似，此处列表说明**

接口名称	接口说明
闲聊发布	向数据库插入评论，使用前端提供的数据新建一个评论类，然后调用数据库接口插入，通过 Handler 提示是否成功。
闲聊区加载	初次加载和下拉刷新时(flag=true)：从数据库中获取最近的 5 条评论，在 Handler 调用界面更新函数。 上拉加载时(flag=false)：重数据库获取接下来的 5 条评论，在 Handler 调用界面更新函数。
闲聊区点赞	先查询当前登录者的学号/教工号是否已经存在于评论点赞表，不存在则将其插入评论点赞表。点赞已存在时 Handler 提示不能重复，否则使用 Handler 更新先点赞数。
闲聊区点踩	先查询当前登录者的学号/教工号是否已经存在于评论点踩表，不存在则将其插入评论点踩表。点踩已存在时 Handler 提示不能重复，否则使用 Handler 更新先点踩数。
闲聊区举报	先查询当前登录者的学号/教工号是否已经存在于评论举报表，不存在则将其插入评论举报表。同时判断举报数是否为 20，如果为 20 则从评论表中移除这条评论。举报已存在时 Handler 提示不能重复。
教师课程查询	根据前教师 tid 查询其所有任教课程，在 Handler 中进行课程更新
学生课程查询	根据学生 sid 查询其所有选择课程，在 Handler 中进行课程更新
课程搜索	根据输入的关键信息，调用后台的搜索模块然后在 Handler 中更新界面
课程添加	从前端控件获取信息，然后新建一个课程类，将前端通知信息填入，同时从本地数据中取出教师信息填入课程类，调用后端接口插入数据库，插入成功后在 Handler 中实现返回
通知信息获取	判断当前登录账号类型，如果为学生，调用数据库中接口查询当前所有通知，传入当前学生 sid 和当前日期。
查询所有通知	通过登录账号类型，根据 sid or tid 分别查询所有通知
查询今日 ddl	通过 sid 和当前日期查询所有今日的 ddl，并在 handler 中调用广播
查询今日通知	通过 sid 和当前日期查询所有今日的新通知，并在 handler 中调用广播
通知发布	从前端控件获取信息，然后新建一个通知类，将前端通知信息填入，调用后端接口插入数据库，插入成功后在 Handler 中实现返回
作业发布	从前端控件获取信息，然后新建一个作业类，将前端作业信息填入，调用后端接口插入数据库，插入成功后在 Handler 中实现返回
学生选课	向数据库中插入学生 sid 和对应的课程 cid
学生退课	从数据库中移除学生 sid 和对应的课程 cid
教师给分	给分调用数据库接口更新选课表中的对应的 sid 和 cid 的条目的 score

4. 部分前端工作

- **图片裁剪**

  图片裁剪使用 UCrop，由 Yalantis 公司设计的开源库 UCrop，主要功能是：裁剪框不动，图片动，进行裁剪；图片可以旋转，缩放；支持各种比例裁剪框，这个裁剪好处是我们可以固定大小进行裁剪，从而直接裁剪出符合要求的图片，避免后续对图片进行再次操作的必要。

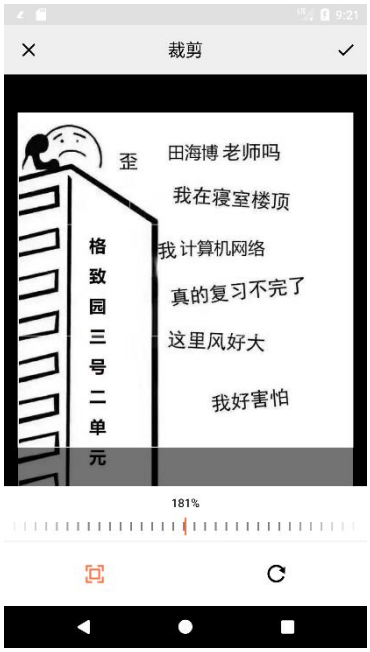
  - 首先需要 AndroidManifest 中注册 UCropActivity 和在 gradle 中加入 Ucrop

```
compile 'com.github.yalantis:ucrop:2.2.1'

<activity
    android:name="com.yalantis.ucrop.UCropActivity"
    android:screenOrientation="portrait"
    android:theme="@style/Theme.AppCompat.Light.NoActionBar"/>
```

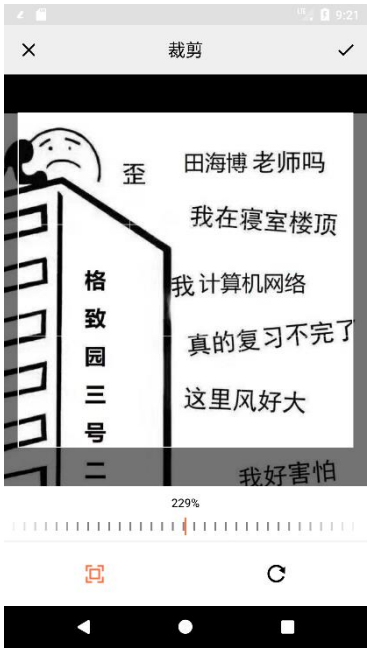
- 通过建造者模式来创建一个 uCrop 对象,并且可以通过 UCrop.Options 来设置一些个性化的参数，这里根据不同设置不同的裁剪比例和大小，以头像为例我们需要裁剪出一个正方形并限制大小为 150*150，配置 uCrop 参数即可实现。

```
/**
 * 裁剪图片方法实现
 * @param uri
 */
protected void startPhotoZoom(Uri uri) {
    //设置储存目录
    Uri mDestinationUri = Uri.fromFile(new File(getCacheDir(), "tmps.jpg"));
    UCrop.Options options = new UCrop.Options();//裁剪界面设置
    options.setToolbarColor(ActivityCompat.getColor(RegisterActivity.this, R.color.colorPrimary1));
    options.setStatusBarColor(ActivityCompat.getColor(RegisterActivity.this,
R.color.colorPrimaryDark));
    options.setToolbarWidgetColor(Color.BLACK);
    UCrop.of(uri, mDestinationUri)//设置裁剪大小比例
            .withAspectRatio(9, 9)
            .withMaxResultSize(150, 150)
            .withOptions(options)
            .start(RegisterActivity.this);
}
```

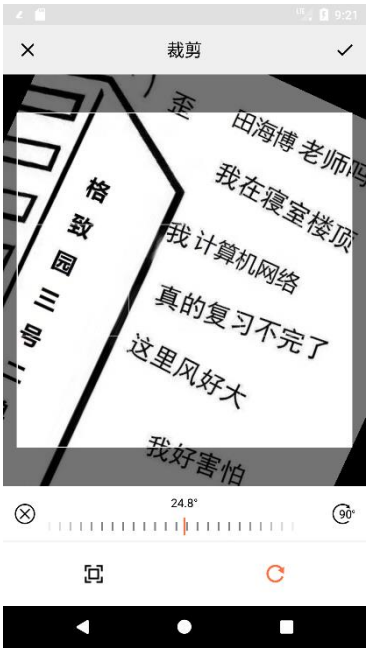
- 效果展示



裁剪



放大



旋转

- **广播**

  广播直接在每次进入到应用，时如果身份为学生，就调用广播发送今日新的通知和今天截止的 ddl，使用静态广播方式实现通知。

- 静态广播的注册

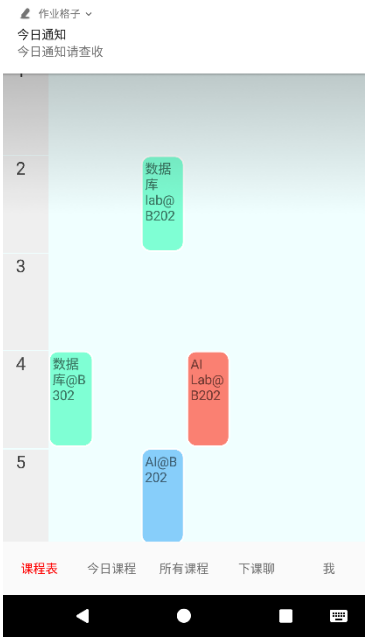
```
<receiver android:name=".Main_Tabhost$StaticReceiver" android:enabled="true">
    <intent-filter>
        <action android:name="cn.chonor.final_pro.staticreceiver" />
    </intent-filter>
</receiver>
```

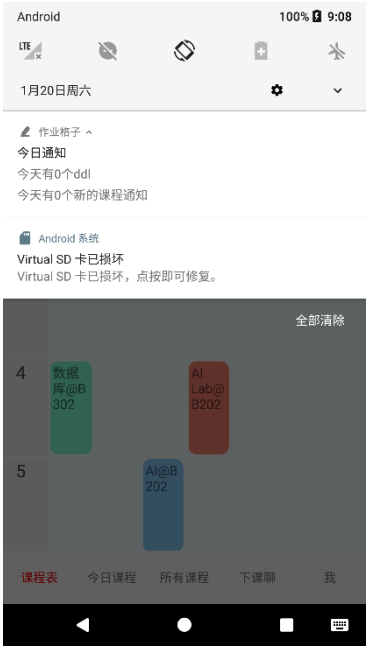
- 广播函数和通知栏函数的配置，此处因为课程信息都是分布的，无法同时跳转到多个界面，所以只显示数量，作为提醒，代码如下：

```
public static class StaticReceiver extends BroadcastReceiver {
    public StaticReceiver(){}
    @Override     //静态广播接收器执行的方法
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals(STATICACTION)) {
            Bundle extras = intent.getBundleExtra("mainActivity");
            int hw=extras.getInt("hw");
            int not=extras.getInt("not");
```

```
            NotificationManager notifyManager = (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);

            //实例化 NotificationCompat.Builde 并设置相关属性
            Notification.InboxStyle inboxStyle = new Notification.InboxStyle()
                    .setBigContentTitle("今日通知")
                    .addLine("今天有"+hw+"个 ddl")
                    .addLine("今天有"+not+"个新的课程通知");

            Notification.Builder builder = new Notification.Builder(context)
                    //设置小图标
                    //设置通知标题
                    .setSmallIcon(R.mipmap.ic_launcher)

                    .setTicker("TickerText:" + "您有新短消息，请注意查收！")
                    .setContentTitle("今日通知")
                    //设置通知内容
                    .setContentText("今日通知请查收")
                    .setStyle(inboxStyle)
                    .setAutoCancel(true)
                    .setDefaults(Notification.DEFAULT_SOUND | Notification.DEFAULT_VIBRATE)
                    .setWhen(System.currentTimeMillis())
                    .setPriority(Notification.PRIORITY_HIGH);
            notifyManager.notify(0, builder.build());
        }
    }
}
```

- 效果展示，展开后会有具体通知。





四、 实验中遇到的困难

1. 数据库连接问题

前期准备的时候，在选定 webservice 的的时候考虑了两种方式，一开始最先考虑的是直连数据库，然后在实践中发现 mysql 提供的官方 jar 包不能实现连接，检测了网络之后还是无法连接，于是准备了使用 HTTP 中的 POST，然后通过 php 进行数据库的访问，然后将数据使用 json 返回，但是后来考虑到加密的问题。还是选择了直连的方式，然后通过去寻找数据库对应版本的驱动，然后重新使用了 mysql-connector-java-3.0.17-ga-bin.jar，而不是之前官方的 jar 包进行访问。

2. 裁剪问题

裁剪部分开始并不是我在实现，然后队友的前端发给我之后我发现队友的裁剪，只有在特定版本中才能使用，再 8.0 中无法使用，然后为了选择一个更通用的裁剪方式选择了 Ucrop 进行裁

剪，Ucrop 的裁剪功能还能对图片进行放缩。

3. 文件上传问题

上传问题是我在测试上传的基础功能的时候发现的，服务端如果直接使用网页，是可以正常上传的，但是在 android 端上传的时候都显示上传成功，但是服务器端没有收到，后来通过 debug 才发现，只有当路径改为内置 SD 卡上时才能正常上传。

4. 数据同步问题

在使用对用给的接口的时候发现，在添加完课程时全部课程中不会立即显示，只有在整个应用移除之后在此安装才会显示，下课聊中也是这样，之后在 debug 的时候代码在 SQL 代码的 limit 在安卓端和网页端上有点区别，然后修改了队友代码，解决了这个问题。

五、 项目总结

这个项目一开始，我们在考虑功能的时候考虑了很多方面，比如作业上传、直接获取学校的课程信息，在一开始设计的时候也有考虑到这个问题，但是实际中发现，学校的课程信息还是没有公开获取方式的，所以我们改变了策略使用教师、学生分开，教师添加课程学生选课的方式进处理。至于上传作业，在实际测试了服务器的负载能力之后不得不放弃，因为服务器并没有负载这个的能力。

项目初衷是要提醒我们的 ddl 临近，主要是受到这个学期 ddl 太多，很容易忘记，然后没交扣分，于是想实现这样一个平台，然后我们考虑到只是单纯的提醒，那和备忘效果差不多，还是很麻烦，所以我们的目标转到了教务系统这种类型的开发，通过添加课程，通过教师端发布通知与作业，学生端直接接收这些信息，就不需要我们手动添加，直接打开软件就可以获取到这些消息，然后我们考虑到竟然做了这做了选课的功能，那就结合课程表提示一下要上的课，同时还增加了互动区。

项目的实现部分主要分为前端后端，我主要是后端，同时还加做了一些前端的工作，做这种项目其实实现并不困难，都是一些控件方法的使用，主要就是通过查的方式了解控价如何运作，并不会出现很多困难，主要的困难就是配合，这么大的项目配合还是很重要的，尽管我提供了一些基础接口，但是界面部分很多还是基于队友实现的，不同的代码风格对于我之后的后端结合压力真的很大，尽管已经要求队友提供动作时间的接口，但是实际中还是很麻烦，这主要还是协调问题，前端的队友并没有真正的使用过 github 只是把它当成网盘使用，这个我吐槽很久，他们也没有使用过 github 合作，整个项目的开始就是在交队友使用 github，然后对我在前后端结合中发现的 bug 让他们直接去调整，这个不同的代码风格对于配合真的致命，需要一点时间才能读懂他们的代码目的，其次前端队友对于接口也不是很熟悉，这导致了我这边工作量的加大，只是给我一个函数告诉我会调用，但是没有给我需要更改的数据，这使我还要继续找他的数据存放。

这个项目对我来说是对于这个学期学的东西的复习和检验，同时让我体会到了项目合作需要协调，对于这种大项目，队友之间的项目协调是非常重要的，协调到位还是能事半功倍的，也让我体会到了需要一个项目经理。对于项目的不足吧，主要还是碍于服务器问题，没有实现作业的上传功能，只实现了发布，这也是一点遗憾吧，还有就是美工还是不是很到位。

六、 课程总结

Android 这门课还是比较实用的一门课，这是这学期少数几门实践大于理论的课程，也是让我觉得学到的东西较多的一门课，对于 Android 的开发也有了认识，一开始以为需要重新学习 java，但是在课程中发现其实 Android 的开发并不是需要很多的算法，主要在于功能的设计和选取合适的控件进行功能的实现，这门课我觉得主要还是查表，上完这门课我觉得我得到的就是我对于要实现什么功能，需要点什么方法让我有一个印象，并不需要记住具体的使用方式，当我们实际需要使用时只要知道需要的控件再去查就好了，这就好像建立了一个索引，之后在去通过索引去查询如何使用。