目录

1. 系统分析（同学 2 撰写）

1.1 系统需求

进入新世纪以来，由于社会发展和经济变化太快，城市化和互联网化带来的生活压力，抑郁症等新型心理问题成为流行的“时代病”。因此心理健康也越来越受到人们的关注。但由于人们对心理问题的了解和认识的滞后，以及传统的心理咨询方式会受到时间和地点的限制，和自身工作繁忙，难以抽出时间进行而被迫继续忍耐，许多隐患不能及早地发现和治疗，心理问题不能得到正确的对待和解决。面对线上心理咨询和心理治疗的缺少和拥有心理隐患的人们对线上的咨询和治疗的渴望处在矛盾境地，有些人愿意咨询心理咨询师但不方便或者是不想与心理咨询师线下交流。即本系统是可线上的心理问题咨询治疗的系统。

1.2 系统设计资源

主要是运用 Android studio64x，LitePal 数据库进行开发。

1.3 系统可行性

在目前的网络大时代，本系统的开发以及投入市场在目前社会生活中具有巨大的潜在市场。系统开发的可行性是基本能实现的，而且操作的相对简单，在系统投入市场后投入成本在可控范围内，而且从长期看收益是可观的，由此可见，本系统的开发以及投入使用是可行的。而本次设计主要是设计关于本系统中心理咨询师的一些功能。

2. 系统设计（王东启撰写）

2.1 系统目标

根据注册心理师对心理系统软件的要求，制定目标如下：

（1） 操作简单方便，界面大气舒适。

（2） 方便注册心理师进行上传报告，上传问卷，查看订单等操作。

（3） 确保个人信息（注册心理师）安全性，使用设置密码，验证的方法。

（4） 系统运行稳定，安全可靠，舒适。

2.2 系统功能结构

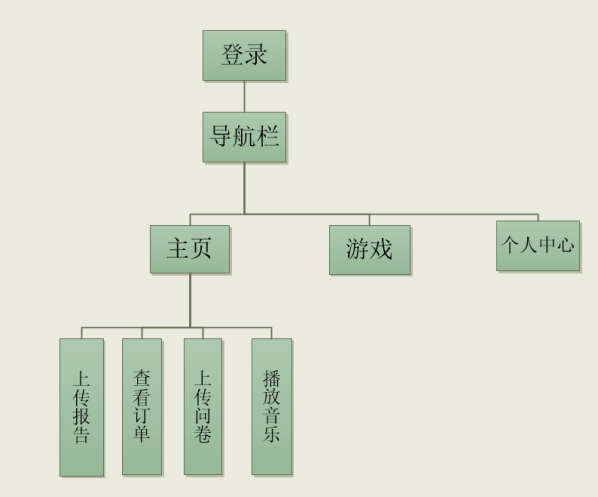


图.2.2 系统功能结构图

2.3 系统业务流程图

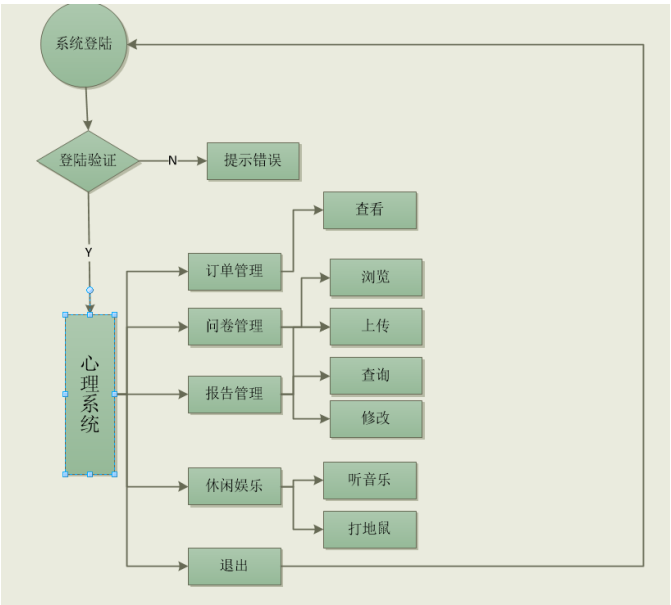


图 2.3.系统业务流程图

2.4 系统编码规范

2.4.1数据库命名规范

（1）数据库：数据库以英文单词 Psychology 的缩写命名。Psychology 的意思是心理学；心理状态

表 2.4.1（1）.数据库描述表

数据库名称	描述
Psy.db	心理系统数据库

（2）数据表：数据表以 Psy 开头，外加相关表的英文释义，表的英文释义首字母大写。

表 2.4.1（2）.数据表描述表

数据表名称	描述
PsyOrder	订单表
PsyUser	用户表

（3）字段

每个表中的字段都以表的英文单词的首字母开头，后面跟随字段的英文单词或者英文

单词的缩写。注：每张表中都有唯一 id，就起为 id。

表 2.4.1（3）.字段描述表

字段名称	描述
PsyOrder	
id	编号（订单编号）
uid	用户编号
oprice	心理系统数据库
otime	订单生成时间
opay	订单支付状态
PsyUser	
id	编号（用户编号）
uname	用户昵称
upass	登录密码
uidentity	用户身份
usex	性别

2.4.2程序代码命名规范

（1）包分类规范

在 psy 下主要分为以下六个包：

（1.broadcast 包下有实现广播功能的类；

（2.entity 包下放 dao 层；

（3.implementation 的中文释义是实现，下面分三个包，分别是实现订单功能的包，实现问卷功能的包，实现报告功能的包；

（4.music 包实现音乐播放的功能

（5.ui 包下放了三个导航栏包，三个包下放着各自的界面。

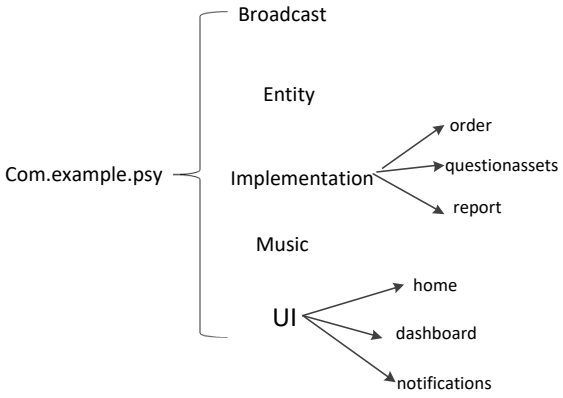


图 2.4.2(1).包分类图

（2）类文件命名规则

（1）动词加名词组合，动词首字母大写，名词首字母大写。

（2）形容词加名词组和，形容词首字母大写，名词首字母大写。

（3）单独动词，动词首字母大写。

表 2.4.2（1）.类文件命名规范表

v+n	Add Order	Find Activity	Upload Questionnaire	Up report			
adj+n	SMS Receiver	Psy Order	Psy User	File Adapter	Music Activity	Init Database	Sum Login
v	Register						

（3）组件命名规则

表 2.4.1（2）.组件命名规范表

控件	起名形式
Fragment	nav_host_fragment
Button	播放：openAssetMusic
	暂停：pause
	返回：back
Textview	注册心理专家登录界面：tx_psylogin
	用户名：tx_psyname
	密码：tx_psypass
EditText	请输入注册心理师用户名：ed_psyname
	请输入注册心理师密码：ed_psypass
RadioGroup	rg_registor
RadioButton	rg_reguser
FrameLayout	frameLayout1
ImageView	imageView1
ListView	lv_order
RecyclerView	question_rv

3. 数据库与数据表的创建与实现（同学 1 撰写）

3.1 数据库与数据表的创建

心理驿站 app 使用的是 Psy 数据库，根据小组现在的需求一共创建了两张表分别是 psyuser 和 psyorder。psyuser 中的属性有 id（用户编号），uname（用户名），upass（登录密码），uidentity（用户身份），usex（用户性别），psyorder 中的属性有 id（订单编号），uid（用户编号），oprice（价格），otime（订单创建时间），opay（是否已支付）。

3.2 数据库与数据表的实现

3.2.1数据库的实现

在 java 下创建 assets 包，新建 litepal.xml 文件，文件中设置数据库名 Psy，设置数据库把版本号以及数据库中的表 psyuser 和 psyorder，如图 3.2.1（1）litepal 设置所示。

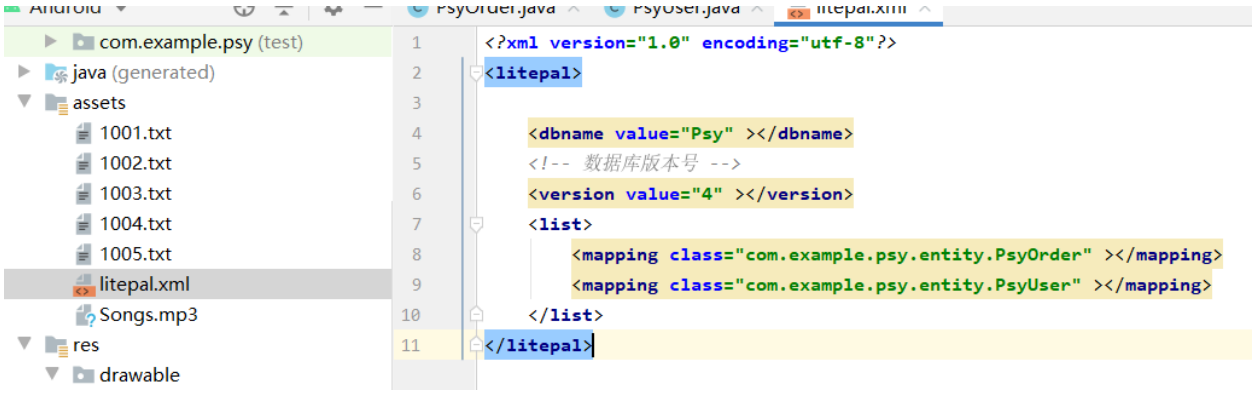


图 3.2.1（1）litepal 设置

3.2.2 数据表的实现

在 java 下 com.example.psy 中创建 entity 包存放数据表的类，新建 PsyUser 和 PsyOrder 的 java 文件，文件中设置个数据表的属性以及各属性的 getter 和 setter 方法，如图 3.2.2（1）PsyUser 设置和图 3.2.2（1）PsyUser 设置所示。

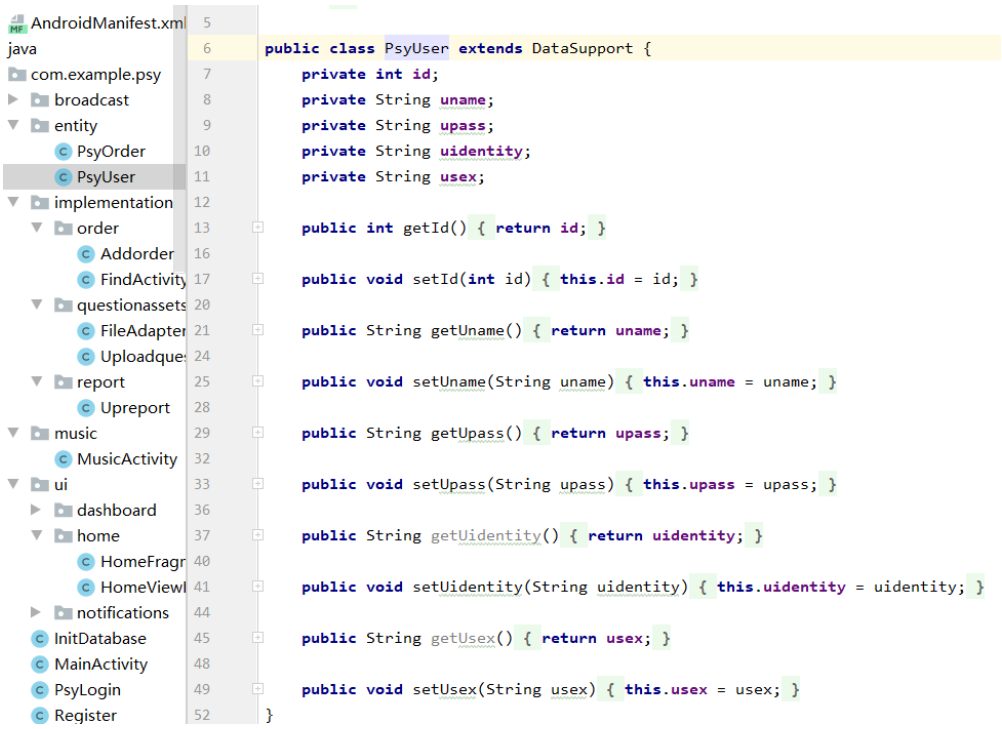


图 3.2.2（1）PsyUser 设置

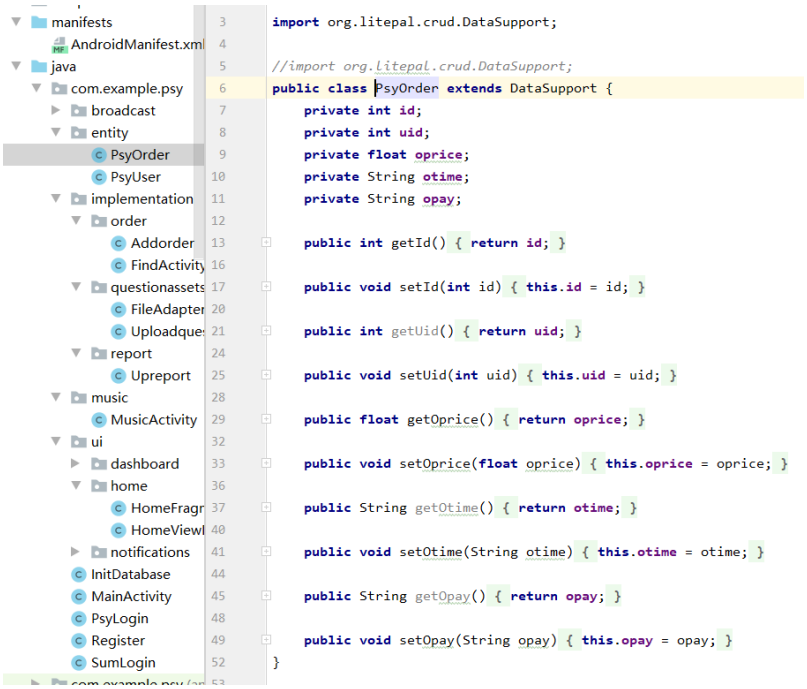


图 3.2.2（2）PsyOrder 设置

3.2.3 数据库的初始化

在 java 下创建 SumLogin 中通过按钮实现数据库的初始化如图 3.2.3（1）初始化数据库所示，并且保证在每次运行该 app 时只运行一次。同时通过添加 InitDatabase 中的 init

方法往表格中添加一些数据如图 3.2.3（2）（3）所示。代码执行结果通过 Android Studio 自带的 Database Inspector 展示，如图 3.2.3（4）（5）用户和订单信息表所示。

```
case R.id.button5:
    LitePal.initialize( context: this);
    Connector.getDatabase();
    new InitDatabase().init();
    break;
```

图 3.2.3（1）初始化数据库

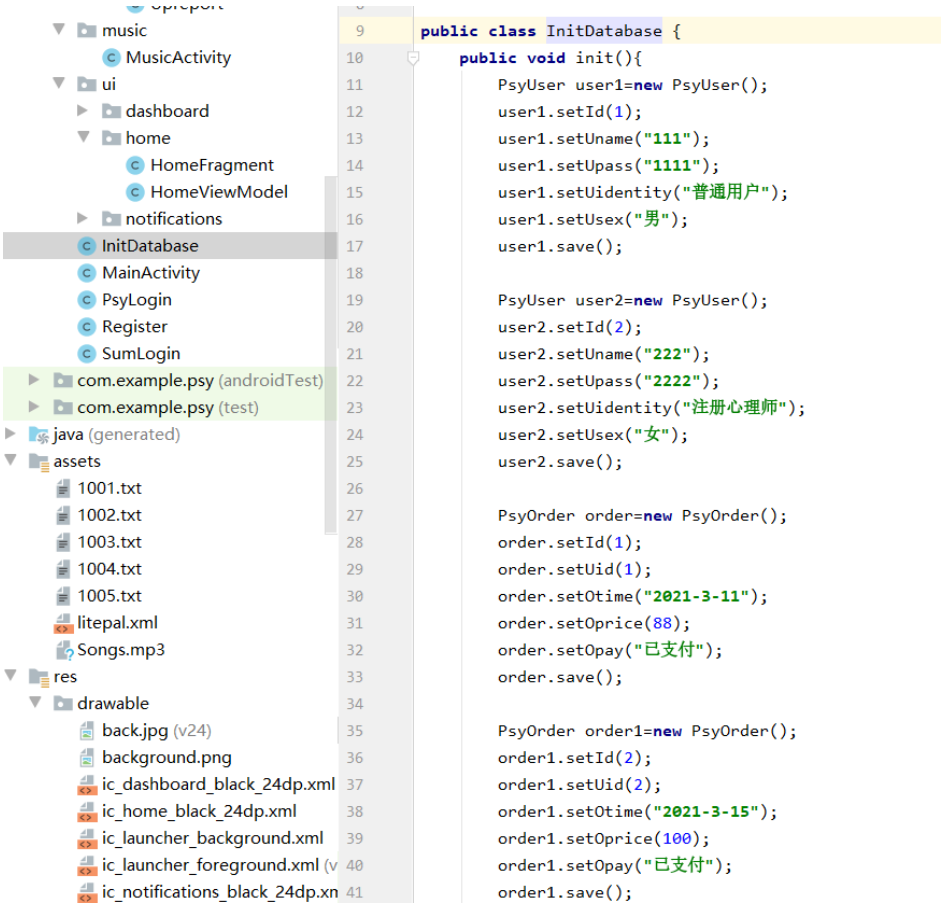


图 3.2.3（2）添加用户信息

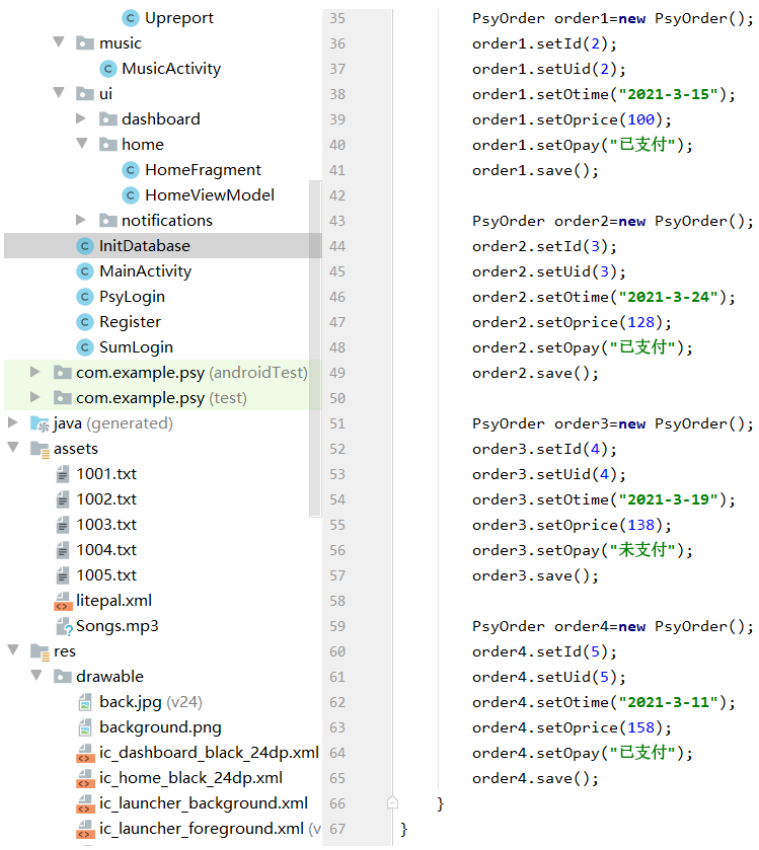


图 3.2.3（3）添加订单信息

图 3.2.3（4）用户信息表

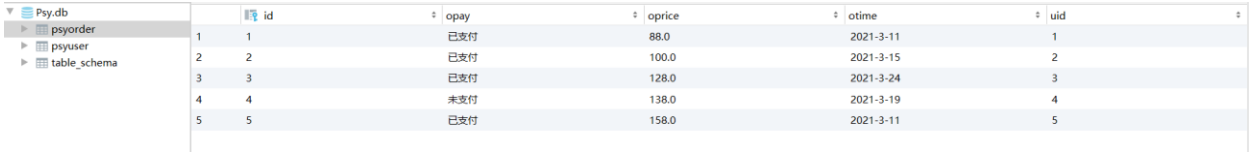


图 3.2.3（5）订单信息表

4. 程序窗体布局设计（同学 1 撰写）

4.1 登录及注册窗体设计

心理驿站 app 登录界面以总分总的形式展现，小组目标是通过一个总的登录界面，让用户选择自己的登录渠道实现不同身份用户的登录，分为客服、普通用户、注册心理师、专家，通过点击不同的按钮进入自己所需的界面。但由于时间原因并没有完善所有的 app 功能，仅实现了注册心理师专家方面的部分，所以各个按钮通向的登录界面为统一登录界面，并且由于专家和客服相当于平台内部人员，因此在注册界面中，没有该身份的选择。

总体登录界面由一个背景，一个 TextView 和五个 Button 组成，其中前四个按钮点击通往同一登录界面，最后一个为建立数据库，实现数据库数据表的初始化。总体界面如图 4.1（1）总体登录界面所示。

注册心理专家登录界面是由一个 ImageView、三个 TextView、两个 EditText 和两个 Button 组成。如图 4.1（2）登录界面所示，出于界面的总体颜色搭配，采用统一的薄荷绿颜色，实现界面的美观。

用户注册界面是由一个 ImageView、六个 TextView、三个个 EditText、两个 RadioGroup 和三个 Button 组成。如图 4.1（3）用户注册界面所示，出于界面的总体颜色搭配，采用统一的淡紫色颜色，实现界面的美观。

图 4.1（1）总体登录界面

图 4.1（2）心理师登录界面

图 4.1（3）用户注册界面

4.2 主界面布局

心理驿站 app 主界面是以导航的形式展现，主页是由四个 ImageButton、六个 TextView、一个 EditText、一个 Button 和一个 ListView 组成，如图 4.2（1）主页界面所示。其中 ListView 所展示的内容随着改变搜索框内的内容而改变。游戏界面是由两个 ImageView 组成，通过改变 Image 的位置实现游戏功能，如图 4.2（2）游戏界面所示。个人中心界面是由六个 TextView、两个 button 和两个 Button，如图 4.2（3）个人中心界面所示。

图 4.2（1）主页界面

图 4.2（2）游戏界面

图 4.2（3）个人中心界面

4.3 主要功能界面布局

心理驿站 app 的主要功能界面有上传报告、查看订单，问卷上传以及音乐组成。上传报告是由一个 TextView、两个 EditText 和三个 Button 组成，如图 4.3（1）上传报告界面所示。查看订单界面一个 TextView、三个 Button 和一个 ListView 组成，如图 4.3（2）查看订单界面所示。问卷上传界面由三个 TextView、一个 RecyclerView 和两个 button 组成，如图 4.3（3）问卷上传界面所示。音乐界面由三个 Button 组成，如图 4.3（4）音乐界面所示。

图 4.3（1）上传报告界面

图 4.3（2）查看订单界面

图 4.3（3）问卷上传界面

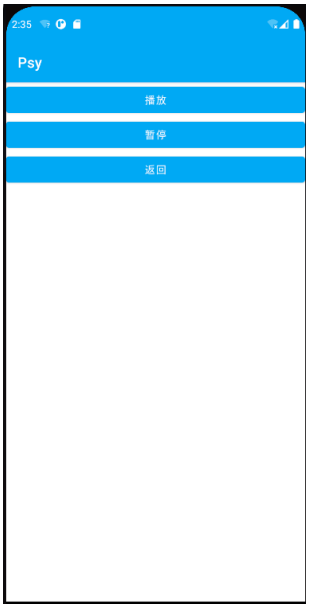


图 4.3（4）音乐界面

5. 主要功能介绍

5.1 登录及注册功能（同学 1 撰写）

在总的登录界面中点击客服登录、用户登录、注册心理师登录、专家登录任意按钮跳转到登录界面，在登录界面中的两个 EditText 中输入相应内容即可实现登录功能，若是输入的内容与在数据库中用户名或密码不相符则登录失败，弹出消息框，密码清空，如图 5.1（1）登录失败所示。所示若是相同则登录成功，弹出登录成功的消息框并完成主页的跳转，如图 5.1（2）登录成功所示。

注册功能则需要完成用户名、两次密码的填写以及身份和性别的选择，若是这些内容不完整则注册失败，弹出内容缺失的消息框，如图 5.1（3）注册内容缺失所示。若是两次密码填写不相符也会弹出两次密码不同的消息框，如图 5.1（4）注册两次密码不相符所示。当这些内容都更正后，则会弹出注册成功的提示框，如图 5.1（5）注册成功所示，并且数据库也会做出相应改变，添加相应数据，如图 5.1（6）数据库的变化所示。

图 5.1（1）登录失败

图 5.1（2）登录成功

图 5.1（3）注册内容缺失

图 5.1（4）注册两次密码不相符

图 5.1（5）注册成功

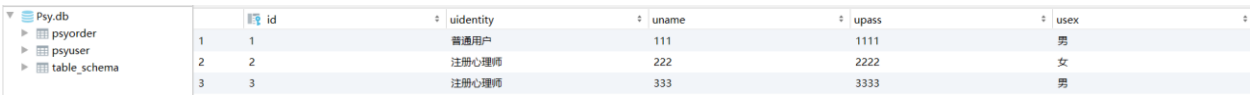


图 5.1（6）数据库的变化

5.2 搜索功能（同学 1 撰写）

在 APP 主页中有一个 ListView 用来显示订单的部分信息，当“搜索”后面的 EditText 中输入的内容与订单的某些信息相符，点击搜索按钮，ListView 中则会显示相关的订单信息，如图 5.2（1）相关订单信息所示。当 EditText 中没有内容（文本为空），点击搜索

按钮，ListView 中则会显示全部的订单信息，如图 5.2（1）全部订单信息所示。

图 5.2（1）相关订单信息

图 5.2（1）全部订单信息

5.3 导航功能（同学 1 撰写）

在主页下方有一个导航栏，由 3 个 item 组成分别是 home、dashboard 和 notifications，点击相应按钮即可完成相应界面的跳转，如图 5.3（1）（2）（3）所示。与传统页面跳转不同的是导航是固定的。导航功能实现本质上是用一个页面布局中加载了一个类似于 Container 的容器，三个界面的内容在这个容器中改变，而容器之外的导航栏不变。

图 5.3（1）home 界面

图 5.3（2）dashboard 界面

图 5.3（1）notifications 界面

5.4 上传报告（王东启撰写）

上下两个 EditText。上面输入报告名，下面输入报告内容，点击上传，即可上传。上传成功后会有上传成功消息显示，并会将两个 EditText 里的内容设置为空。在上面的 EditText 里输入报告名，点击查看，即可查看报告内容。上传成功以后，输入报告名点击查看 ，还可以对查找出来的报告进行修改。

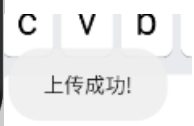


图 5.4（1）.上传报告图　　图 5.4（2）.上传成功显示图

5.5 查看订单（王东启撰写）

从主页里点击查看订单即可查看全部订单，并且可以查看每个订单里的全部内容。

图 5.5（1）.查看订单图

5.6 打地鼠小游戏（王东启撰写）

此 app 中两个休闲娱乐功能中的一个，打地鼠 。地鼠会来回移动，“打到”（点击到）地鼠最下方会显示你共打到了几只地鼠。

图 5.6.打地鼠小游戏图

5.7 音乐播放（同学 2 撰写）

主要通过布局中添加三个按钮，分别是播放、暂停和返回按钮；通过点击播放即可存储在 assets 文件夹下的音乐资源播放音乐，点击暂停即可暂停音乐，如图 5.7 所示。

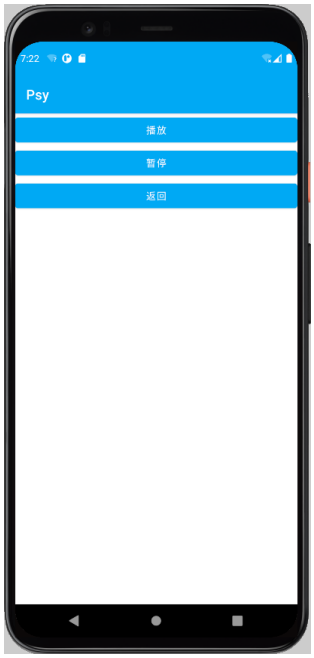


图 5.7 音乐播放图

5.8 问卷上传（同学 2 撰写）

在问卷上传界面中，点击上传按钮，会依次调取 assets 中的问卷，并且每点击一个已经上传问卷，则会显示问卷中的内容。效果如下图 5.8。

图 5.8 问卷上传图

5.9 Service 应用（同学 2 撰写）

在个人中心的界面中添加两个按钮，分别是启动 service 和停止 service 两个按钮，单击启动 service，会在 logcat 显示相应方法调用，如下图 5.9（2）所示；点击停止按

钮，也会在 logcat 日志中显示相关方法被调用，如下图 5.9（3）所示。

图 5.9（1）service 应用图

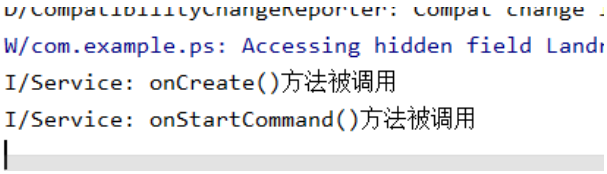


图 5.9（2）启动 service 应用图

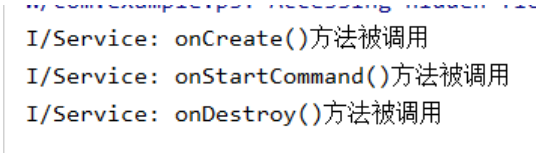


图 5.9（3）停止 service 应用图

5.10 广播应用（同学 2 撰写）

模拟实现当收到信息时给出提示信息的功能。如图 5.10 所示。

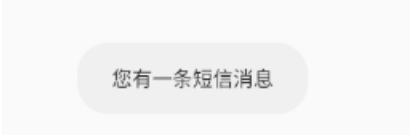


图 5.10 广播应用图

6. 普通功能实现

6.1 登录及注册功能实现（同学 1 撰写）

想要完成登录功能和注册功能首先需要完成相应组件的初始化，如图 6.1（1）（2）登录和注册组件初始化中所示。然后就是组件实现的功能的编码，如图 6.1（3）（4）（5）中所示，登录界面中登录按钮实现与数据库中相应内容的匹配，成功发出信息跳转，失败提示清空密码框，注册按钮实现向注册界面的跳转。注册界面中，注册完成内容完整性的检验与数据库中相应内容的匹配，成功则发出信息，失败提示错误所在，重置按钮完成所有内容的清空，返回按钮则完成向登录界面的跳转。

```
public class PsyLogin extends AppCompatActivity {

    private EditText ed_psyname;
    private EditText ed_psypass;
    private Button btn_psylogin;
    private Button btn_psyreg;


    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_psy_login);

        ed_psyname = (EditText) findViewById(R.id.ed_psyname);
        ed_psypass = (EditText) findViewById(R.id.ed_psypass);
        btn_psylogin = (Button) findViewById(R.id.btn_psylogin);
        btn_psyreg = (Button) findViewById(R.id.btn_psyreg);
```

图 6.1（1）登录组件初始化

```
private EditText ed_regname;
private EditText ed_psypass;
private EditText ed_psytpass;
private RadioGroup rg_registor;
private RadioGroup rg_regsex;
private Button btn_register;
private Button btn_regreset;
private Button btn_back;

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_register);

    ed_regname = (EditText)findViewById(R.id.ed_regname);
    ed_psypass = (EditText)findViewById(R.id.ed_psypass);
    ed_psytpass = (EditText)findViewById(R.id.ed_psytpass);
    rg_registor = (RadioGroup)findViewById(R.id.rg_registor);
    rg_regsex = (RadioGroup)findViewById(R.id.rg_regsex);
    btn_register = (Button)findViewById(R.id.btn_register);
    btn_regreset = (Button)findViewById(R.id.btn_regreset);
    btn_back = (Button) findViewById(R.id.btn_back);
```

图 6.1（2）注册组件初始化

```
btn_psyreg.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent=new Intent( packageContext: PsyLogin.this,Register.class);
        startActivity(intent);
    }
});

btn_psylogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        boolean flag=false;
        String uname=ed_psyname.getText().toString().trim();
        String upass=ed_psypass.getText().toString().trim();
        List<PsyUser> list= DataSupport.findAll(PsyUser.class);
        //Iterator<User> it=list.iterator();
        for(PsyUser user:list){
            if(user.getUname().equals(uname) && user.getUpass().equals(upass)){
                flag=true;
            }
            if(flag){
                Intent intent=new Intent( packageContext: PsyLogin.this,MainActivity.class);
                startActivity(intent);
                Toast.makeText(getApplicationContext (), text: "登录成功!",Toast.LENGTH_SHORT).show();
            }else{
                Toast.makeText(getApplicationContext (), text: "登录失败，用户名或密码错误！",Toast.LENGTH_SHORT).show();
                ed_psypass.setText("");
            }
        }
    }
});
```

图 6.1（3）登录组件代码

```
btn_register.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(!(TextUtils.isEmpty(ed_regname.getText())) && !(TextUtils.isEmpty(ed_psypass.getText())) && !(TextUtils
            if(ed_psypass.getText().toString().equals(ed_psytpass.getText().toString())){
                RadioButton r1=findViewById(rg_registor.getCheckedRadioButtonId());
                RadioButton r2=findViewById(rg_regsex.getCheckedRadioButtonId());
                PsyUser user=new PsyUser();
                user.setUname(ed_regname.getText().toString());
                user.setUpass(ed_psypass.getText().toString());
                user.setUidentity(r1.getText().toString());
                user.setUsex(r2.getText().toString());
                user.save();
                Toast.makeText(getApplicationContext (), text: "注册成功！！！",Toast.LENGTH_SHORT).show();
            }else{
                Toast.makeText(getApplicationContext (), text: "两次密码输入不一致！！！",Toast.LENGTH_SHORT).show();
            }
        }else{
            Toast.makeText(getApplicationContext (), text: "请输入完整！！！",Toast.LENGTH_SHORT).show();
        }

    }
```

图 6.1（4）注册按钮代码

```
        btn_regreset.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ed_regname.setText("");
                ed_psypass.setText("");
                ed_psytpass.setText("");
                rg_registor.clearCheck();
                rg_regsex.clearCheck();
            }
        });

        btn_back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent=new Intent( packageContext: Register.this,PsyLogin.class);
                startActivity(intent);
            }
        });

    }
```

图 6.1（5）注册剩余组件代码

6.2 搜索功能的实现（同学 1 撰写）

搜索功能的实现首先需要判断输入框是否为空，为空就从数据库中查找出全部内容，

不为空则从数据库中实现 where 的条件搜索，把搜索到的内容放入 List 中，由于搜索到的内容是完整的所有信息，因此需要通过 HashMap 遍历出所有需要展示的内容放入到 Map 中，最后通过 Adapter 放入到 ListView 中，如图 6.2 搜索功能代码所示。

```
case R.id.btn_search:
    if(TextUtils.isEmpty(et_search.getText().toString().trim())){
        List<PsyOrder> orders = DataSupport.findAll(PsyOrder.class);
        List<Map<String, Object>> listItems = new ArrayList<Map<String, Object>>(); // 创建一个List集合
        for (PsyOrder order : orders) {
            Map<String, Object> map = new HashMap<~>();
            map.put("id", order.getId());
            map.put("oprice", order.getOprice());
            listItems.add(map); // 将map对象添加到List集合中
        }
        simpleAdapter = new SimpleAdapter(getActivity(), listItems, R.layout.home_listv, new String[] { "id", "oprice"},
                new int[] { R.id.id, R.id.price }); // 创建SimpleAdapter
        lv_order.setAdapter(simpleAdapter);
    }else{
        List<PsyOrder> orders = DataSupport.where( ...conditions: "id=? or oprice=?",et_search.getText().toString(),et_search
        List<Map<String, Object>> listItems = new ArrayList<~>(); // 创建一个List集合
        for (PsyOrder order : orders) {
            Map<String, Object> map = new HashMap<~>();
            map.put("id", order.getId());
            map.put("oprice", order.getOprice());
            listItems.add(map); // 将map对象添加到List集合中
        }
        simpleAdapter= new SimpleAdapter(getActivity(), listItems, R.layout.home_listv, new String[] { "id", "oprice"},
                new int[] { R.id.id, R.id.price }); // 创建SimpleAdapter
        lv_order.setAdapter(simpleAdapter);
    }
    break;
```

图 6.2 搜索功能代码

6.3 上传报告（王东启撰写）

（1）使用技术：IO 流

（2）界面布局：垂直线性布局，主要使用组件如下图：

```
<EditText
    android:id="@+id/edFilename"
    android:layout_width="match_parent"
    android:layout_height="107dp" />

<EditText
    android:id="@+id/edFilecontent"
    android:layout_width="match_parent"
    android:layout_height="103dp" />
```

图 6.3（1）.上传报告界面代码图 1

```
<Button
    android:id="@+id/upload_report"
    android:layout_width="wrap_content"
    android:layout_height="60dp"
    android:layout_weight="0.5"
    android:text="上传"
    android:textSize="30dp"
    />
<Button
    android:id="@+id/select_report"
    android:layout_width="wrap_content"
    android:layout_height="60dp"
    android:layout_weight="0.5"
    android:text="查看"
    android:textSize="30dp"
     />
```

图 6.3（2）.上传报告界面代码图 2

（3）代码实现功能：设立三个监听事件，分别是上传报告，查看报告，返回主界面。设置两个 EditText.

```
upload_report=(Button)this.findViewById(R.id.upload_report);
select_report=(Button)this.findViewById(R.id.select_report);
btn_back=(Button)this.findViewById(R.id.btn_back);

edFilename=(EditText)this.findViewById(R.id.edFilename);
edFilecontent=(EditText)this.findViewById(R.id.edFilecontent);
```

图 6.3（1）.上传报告代码图 1

①上传报告使用输出流,从编辑框里获取报告名,报告内容。将报告内容写入,显示上传成功消息。上传成功后将两个 EditText 置空。

```
String filename=edFilename.getText().toString();//从文本编辑框里获取文件名
String filecontent=edFilecontent.getText().toString();//从文本编辑框里获取文本内容
FileOutputStream out=null;//打开输出流
try{
    out=context.openFileOutput(filename, Context.MODE_PRIVATE);//私有覆盖模式，只能被当前应
    out.write(filecontent.getBytes( charsetName: "UTF_8"));//将报告内容写入
    Toast.makeText(getApplicationContext (), text: "上传成功!",Toast.LENGTH_SHORT).show();
    edFilename.setText("");
    edFilecontent.setText("");
```

图 6.3（2）.上传报告代码图 2

②查看报告使用输入流，获取文件输入流中的数据，通过 while 循环将指定报告的内容显示出

```
String filename=edFilename.getText().toString();
FileInputStream in=null;//打开输入流
ByteArrayOutputStream bout=null;//捕获内存缓存中的数据，转换成字节数组
byte[]buf=new byte[1024];//定义保存数据的数组
bout=new ByteArrayOutputStream();
int length=0;
try{
    in=context.openFileInput(filename);//获得文件输入流
    while((length=in.read(buf))!=-1){//如果从输入流中读取的数据
        bout.write(buf, off: 0,length);//将指定字节数组从偏移量处(off)开始的len字节写入此字节数组输出流。
    }
    byte[] content=bout.toByteArray();//获取内存缓冲中的数据
    edFilecontent.setText(new String(content, charsetName: "UTF_8"));//显示报告内容
}catch(Exception e){
    e.printStackTrace();
}
edFilecontent.invalidate();
try{
    in.close();//关闭输入流
    bout.close();
}catch(Exception e){}
```

图 6.3（3）.上传报告代码图 3

6.4 查看订单（王东启撰写）

（1）使用技术：适配器，litepal 数据库

（2）界面布局：垂直线性布局，主要使用组件如下图：

```
<ListView
    android:id="@+id/lvinaccountinfo"
    android:scrollbarAlwaysDrawVerticalTrack="true"
    android:layout_width="wrap_content"
    android:layout_height="400dp"
    android:scrollbarSize="10dp" />
```

图 6.4（1）.查看订单界面图 1

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
<TextView
    android:id="@+id/t1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="28dip"
    android:text="订单编号："
    android:textColor="@color/black"
    />
<TextView
    android:id="@+id/id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="28dip"
    android:textColor="@color/black"/>
</LinearLayout>
```

图 6.4（2）.查看订单界面图 2

（3）代码实现功能：查询全部订单，创建一个 list 集合。

```
List<PsyOrder> orders = DataSupport.findAll(PsyOrder.class);
myListView = (ListView) findViewById(R.id.lvinaccountinfo);
List<Map<String, Object>> listItems = new ArrayList<Map<String, Object>>(); // 创建一个List集合
```

图 6.4（3）.查看订单代码图 1

使用 for 循环，map 对象获取订单表中的值，将 map 对象添加到 list 集合中

```
for (PsyOrder order : orders) {
    Map<String, Object> map = new HashMap<String, Object>();
    map.put("id", order.getId());
    map.put("uid", order.getUid());
    map.put("otime", order.getOtime());
    map.put("oprice", order.getOprice());
    map.put("opay", order.getOpay());
    listItems.add(map); // 将map对象添加到List集合中
}
```

图 6.4（4）.查看订单代码图 2

使用适配器将每个订单的全部内容进行循环显示

```
SimpleAdapter adapter = new SimpleAdapter( context: this, listItems, R.layout.list_entry, new String[] { "id", "uid","oprice","otime","opay"},
        new int[] { R.id.id,R.id.uid,R.id.oprice,R.id.otime, R.id.opay }); // 创建SimpleAdapter
myListView.setAdapter(adapter);
```

图 6.4（5）.查看订单代码图 3

6.5 打地鼠（王东启撰写）

此部分功能作为特色将放在特色功能里详细说明

图 6.5.打地鼠小游戏图

6.6 Service 应用实现（同学 2 撰写）

建立监听事件，设置单击事件，调用 service 类中的 oncreate()方法和 ondestroy()方法以及 onStartCommand()方法，调用成功返回，如下图 6.6(1)(2)所示。

```
final Intent intent=new Intent( packageContext: MainActivity.
        this,MyService.class);
btn_start.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startService(intent);
```

图 6.6（1）部分代码

```
public void onCreate() {
    Log.i( tag: "Service", msg: "onCreate()方法被调用");
    super.onCreate();
}
public void onDestroy() {
    Log.i( tag: "Service", msg: "onDestroy()方法被调用");
    super.onDestroy();
}
public int onStartCommand(Intent intent, int flags, int startId) {
    Log.i( tag: "Service", msg: "onStartCommand()方法被调用");
    return super.onStartCommand(intent, flags, startId);
}
```

图 6.6（2）部分代码

6.7 广播应用实现（同学 2 撰写）

在聊天界面中模拟实现当收到信息时给出提示信息的功能，创建一个 SMSReceiver 类，接口为 BroadcastReceiver 类，定义 onReceive（）方法，其次创建对话框来相应该方法的实现，最后要在 AndroidManfist 中注册。部分代码如下图 6.7.

```
public class SMSReceiver extends BroadcastReceiver {
    private static final String action = "android.provider.Telephony.SMS_RECEIVED";
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals(action)) {
            Toast.makeText(context.getApplicationContext(),
                    text: "您有一条短信消息", Toast.LENGTH_LONG).show();
        }
```

图 6.7 广播应用部分代码

7. 特色功能实现

7.1 导航功能的实现（同学 1 撰写）

导航功能的实现首先需要初始化你所需要的的 fragment 和 bottomNaviationView ，如图 7.1（1）Main 中代码所示。然后在建立 ui 文件包，如图 7.1（2）ui 中代码所示。下面存放 home,dashboard 以及 notification 的 model 以及 fragment 的文件，model 负责界面布局方面的相关设置，如图 7.1（3）model 代码中所示。fragment 负责组件功能，如图 7.1（4）fragment 代码中所示。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    BottomNavigationView navView = findViewById(R.id.nav_view);
    // Passing each menu ID as a set of Ids because each
    // menu should be considered as top level destinations.
    AppBarConfiguration appBarConfiguration = new AppBarConfiguration.Builder(
            R.id.navigation_home, R.id.navigation_dashboard, R.id.navigation_notifications)
            .build();
    NavController navController = Navigation.findNavController( activity: this, R.id.nav_host_fragment);
    NavigationUI.setupActionBarWithNavController( activity: this, navController, appBarConfiguration);
    NavigationUI.setupWithNavController(navView, navController);
```

图 7.1（1）Main 中代码

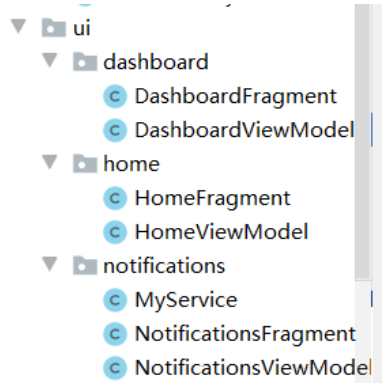


图 7.1（2）ui

```
public View onCreateView(@NonNull LayoutInflater inflater,
                         ViewGroup container, Bundle savedInstanceState) {
    homeViewModel =
            new ViewModelProvider( owner: this).get(HomeViewModel.class);
    View root = inflater.inflate(R.layout.fragment_home, container, attachToRoot: false);
    final TextView textView = root.findViewById(R.id.text_home);
    homeViewModel.getText().observe(getViewLifecycleOwner(), new Observer<String>() {
        @Override
        public void onChanged(@Nullable String s) { textView.setText(s); }
    });
```

图 7.1（3）model 代码

```
public class DashboardViewModel extends ViewModel {

    private MutableLiveData<String> mText;

    public DashboardViewModel() {
        mText = new MutableLiveData<>();
        //mText.setValue("This is dashboard fragment");
    }

    public LiveData<String> getText() { return mText; }
}
```

图 7.1（4）model 代码

7.2 打地鼠（王东启撰写）

（1）作为特色设计的目的：帮助注册心理师使用 app 时放松身心。

（2）使用技术：线程，随机数

（3）界面布局：帧布局，主要使用组件如下图：

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/f1"
    android:background="@drawable/background"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="143dp"
        android:layout_height="304dp"
        android:src="@drawable/mouse" />

</FrameLayout>
```

图 7.2（1）.打地鼠界面图

（4）代码实现功能：

创建随机数组，设置地鼠在指定位置显示。

```
public int[][] position=new int[][]{
        {95,134},{369,177},{231,99},{240,119},{323,93},{350,151},{181,146},{520,11}};//声明一个表示地鼠位置的数组
```

图 7.2（2）.打地鼠代码图 1

```
handler=new Handler(){
    public void handleMessage(Message msg){
        int index=0;
        if(msg.what==0x101){
            index=msg.arg1;
            mouse.setX(position[index][0]);
            mouse.setY(position[index][1]);
            mouse.setVisibility(View.VISIBLE);
        }
        super.handleMessage(msg);
    }
};
```

图 7.2（3）.打地鼠代码图 2

获取 ImageView 组件，为该组件添加触摸监听器，通过 toast 显示打到几只地鼠。

```
mouse=root.findViewById(R.id.imageView1);
mouse.setOnTouchListener(new View.OnTouchListener(){
    public boolean onTouch(View v, MotionEvent event){
        v.setVisibility(View.INVISIBLE);
        i++;
        Toast.makeText(getContext(), text: "打到["+i+"]只地鼠！",Toast.LENGTH_SHORT).show();
        return false;
    }
});
```

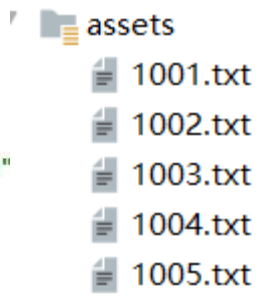
图 7.2（4）.打地鼠代码图 3

7.3 问卷上传实现（同学 2 撰写）

定义上传问卷相关界面的组件，如 Recyclerview、TextView 等；如 7.3（1）所示；调用父类的构建方法，读取 assets 文件夹的文件，如 7.3（2）所示；构建文件适配器，建立响应事件，如图 7.3（3）所示；通过 IO 流读取数据，读取成功后返回，如图 7.3（4）所示。

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/question_rv"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toTopOf="@+id/file_title"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

7.3（1）问卷上传界面部分代码



7.3（2）assets 目录图

```
findViewById(R.id.btn_commit).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (currentFilePos > 5) {
            Toast.makeText(getApplicationContext(),
                    text: "已经没有文件可以添加啦", Toast.LENGTH_SHORT).show();
        } else {
            fileList.add(filePath[currentFilePos]);
            fileAdapter.setData(fileList);
            String content = readAssets(filePath[currentFilePos]);
            fileContent[currentFilePos] = content;
            currentFilePos++;
```

7.3（3）相适应事件部分代码

```
private String readAssets(String fileName) {
    AssetManager assetManager = this.getApplicationContext().getAssets();
    InputStream inputStream = null;
    InputStreamReader isr = null;
    BufferedReader br = null;
    StringBuffer sb = new StringBuffer();
    try {
        inputStream = assetManager.open(fileName);
        isr = new InputStreamReader(inputStream);
        br = new BufferedReader(isr);
        sb.append(br.readLine());
        String line = null;
        while ((line = br.readLine()) != null) {
            sb.append("\n").append(line);
        }
        br.close();
        isr.close();
        inputStream.close();
```

7.3（4）IO 流部分代码

7.4 音乐播放实现（同学 2 撰写）

定义音乐播放的相关组件，定义初始化方法，调用父类的构造方法，创建 mediaplayer 对象；依次调用 setDataSource()、prepare()、start()方法，如下图 7.4（1）（2）所示

```
public class MainActivity extends AppCompatActivity
        implements View.OnClickListener {
    private static final String TAG = "MainActivity";
    private Button openAssetMusic;
    private MediaPlayer mediaPlayer;
    private Button pause;
```

7.4（1）音乐播放部分代码图

```
private void openAssetMusics() {
    try {
        AssetFileDescriptor fd = getAssets().openFd( fileName: "1.mp3");
        mediaPlayer = new MediaPlayer();
        mediaPlayer.setDataSource(fd.getFileDescriptor(),
                fd.getStartOffset(), fd.getLength());
        mediaPlayer.prepare();
        mediaPlayer.start();
    } catch (IOException e) {
        e.printStackTrace();
    }
```

7.4（2）音乐播放部分代码图