Android 移动应用开发期末项目报告

闪易发票----移动端发票整理app

组号：22

小组成员:

组长：梁斐然 学号：20251280

邮箱：[20251280@bjtu.edu.cn](mailto:20251280@bjtu.edu.cn) qq:837504845

成员：20281315 华正佳

目录

[1 概述 4](#_Toc138877643)

[2 背景 4](#_Toc138877644)

[3 项目说明 5](#_Toc138877645)

[3.1 功能模块 5](#_Toc138877646)

[3.2 逻辑流程 5](#_Toc138877647)

[3.3 技术说明 6](#_Toc138877648)

[3.3.1 登录数据库实现 6](#_Toc138877649)

[3.3.2 发票数据库实现 11](#_Toc138877650)

[3.3.3 蒙特卡洛算法 13](#_Toc138877651)

[3.3.4 二维码扫描技术 16](#_Toc138877652)

[3.3.5 ListView设计 16](#_Toc138877653)

[3.3.6 webview展示页面 17](#_Toc138877654)

[3.4 代码架构 17](#_Toc138877655)

[4 应用效果 19](#_Toc138877656)

[4.1 主界面 19](#_Toc138877657)

[4.2 发票上传 20](#_Toc138877658)

[4.2.1 手动上传 20](#_Toc138877659)

[4.2.2 二维码扫描上传 21](#_Toc138877660)

[4.3 发票查看 22](#_Toc138877661)

[4.4 最优报销组合 23](#_Toc138877662)

[4.5 真伪查验 24](#_Toc138877663)

[5 开发过程中遇到的问题及解决办法 25](#_Toc138877664)

[5.1 依赖配置不成功 25](#_Toc138877665)

[5.2 数据库调用问题 25](#_Toc138877666)

[6 心得体会 26](#_Toc138877667)

[7 项目亮点 26](#_Toc138877668)

[8 分工 27](#_Toc138877669)

**图目录**

[图 1 软件架构图 5](#_Toc138877571)

[图 2 发票上传流程图 6](#_Toc138877572)

[图 3 发票快速筛选流程图 6](#_Toc138877573)

[图 4 Bomb数据库控制台 7](#_Toc138877574)

[图 5 获取Application key 7](#_Toc138877575)

[图 6 适配器单个item的ui设计 17](#_Toc138877576)

[图 7 代码架构图 19](#_Toc138877577)

[图 8 主界面 20](#_Toc138877578)

[图 9 手动上传发票界面 21](#_Toc138877579)

[图 10 手动上传发票成功界面 21](#_Toc138877580)

[图 11 二维码扫描 21](#_Toc138877581)

[图 12 扫描中 21](#_Toc138877582)

[图 13 扫描结果 21](#_Toc138877583)

[图 14 直接保存 22](#_Toc138877584)

[图 15 选择类别保存成功 22](#_Toc138877585)

[图 16 未报销发票 22](#_Toc138877586)

[图 17 已报销发票 22](#_Toc138877587)

[图 18 弹出窗口 23](#_Toc138877588)

[图 19 删除后更新的界面 23](#_Toc138877589)

[图 20 最优报销 24](#_Toc138877590)

[图 21 没有选择报销类别 24](#_Toc138877591)

[图 22 选择结果 24](#_Toc138877592)

[图 23 最优报销结果 24](#_Toc138877593)

[图 24 向已报销移动 24](#_Toc138877594)

[图 25 移动结果 24](#_Toc138877595)

[图 26 真伪查验 25](#_Toc138877596)

**表目录**

[表 1 BombUser表的结构 9](#_Toc138877624)

[表 2 User表的结构 10](#_Toc138877625)

[表 3 登录注册功能实现 10](#_Toc138877626)

[表 4 数据库函数及功能 12](#_Toc138877627)

[表 5 最优报销算法 15](#_Toc138877628)

[表 6 二维码扫描 17](#_Toc138877629)

[表 7 小组分工 28](#_Toc138877630)

1. 概述

“闪易发票”是我们设计的一款针对纸质发票进行存储和智能报销的Android应用程序。通过使用sqlite数据库存储技术以及listview,webview等控件进行布局，集成了了五种功能包括登陆注册、发票上传、发票查看、最优报销组合、真伪核验。界面美观简洁，功能强大，使用方便快捷，对于有报销需求的群体非常友好。

1. 背景

当今社会，整理归纳发票繁琐，需要工作人员手动录入发票号。工作量大且繁琐，查询困难。需要用户自己记录都有哪些纸质发票，对于发票收纳不到位的用户很难记得自己都有哪些发票。而获得纸质发票后都上传Android studio应用程序，会方便用户查看。同时电子发票也使假发票的出现更为容易，各种发票一键生成器层出不穷，据调查，有关政府机关等免税单位的财务人员通常只重视程序规范，对发票的真伪性没有防范意识，也没有上网查询比对的习惯而且发票报销通常有所上限。不仅如此需要根据上限挑选发票时，人工计算过程复杂，难以找到最优报销组合。就拿离我们最近的大创报销来说，通常是根据评级给定金额例如1000元，而我们需要找出符合要求的发票并使得发票总和最接近给定金额，这就需要我们自己组合进行计算，十分繁琐耗时。

而我们的软件就是为了解决这些痛点而生。它的特点主要在于：

1. 实用性强。用户只需要扫描发票二维码或手动输入即可上传发票信息，然后可以查看已报销和未报销的发票，以及进行快速筛选和真伪核验，方便有报销需求的用户进行使用。

2.具有创新性：软件核心创新点在于最有发票筛选，这也是目前市场上竞品很少推出的功能。用户点击快速筛选按钮进入定额报销界面，输入金额系统会自动筛选最优发票组合并且经过用户确认即可存入已报销数据库。用户就可以将选出来的纸质发票在实际生活中进行报销。

3. 用户体验感良好。我们的应用操作简单，易于上手，通过listview，webview等组件进行布局，使UI界面美观清晰。由于数据库存储的数据量较小，运行比较稳定，运行速度较快。

4. 兼容性良好。通过在华为、小米、魅族等手机上安装app进行测试，以及在4G，5G和WIFI下分别测试，发现使用效果基本相同。

1. 项目说明
   1. 功能模块

我们的应用主要分为五个模块，分别是登陆注册，发票上传，发票查看，快速筛选报销发票以及真伪核验。其中发票上传包括手动输入和二维码扫描上传，发票查看可以查看已报销发票和未报销发票。而在发票查看中用户可以查看已报销和未报销的发票信息。在快速筛选发票中，用户先要输入金额，然后系统将挑选出的发票进行展示，然后转移到已报销。



1. 软件架构图
   1. 逻辑流程

由于发票查看和真伪检验的流程比较简单，只需点击按钮，所以下面只展示发票上传和快速筛选的流程图。

|  |  |
| --- | --- |
|  |  |
| 1. 发票上传流程图 | 1. 发票快速筛选流程图 |

* 1. 技术说明
     1. 登录数据库实现

**存放账号密码**

我们使用Bomb数据库对用户名和密码进行云端存储，使用Bomb云端存储用户名和密码有以下几个好处：

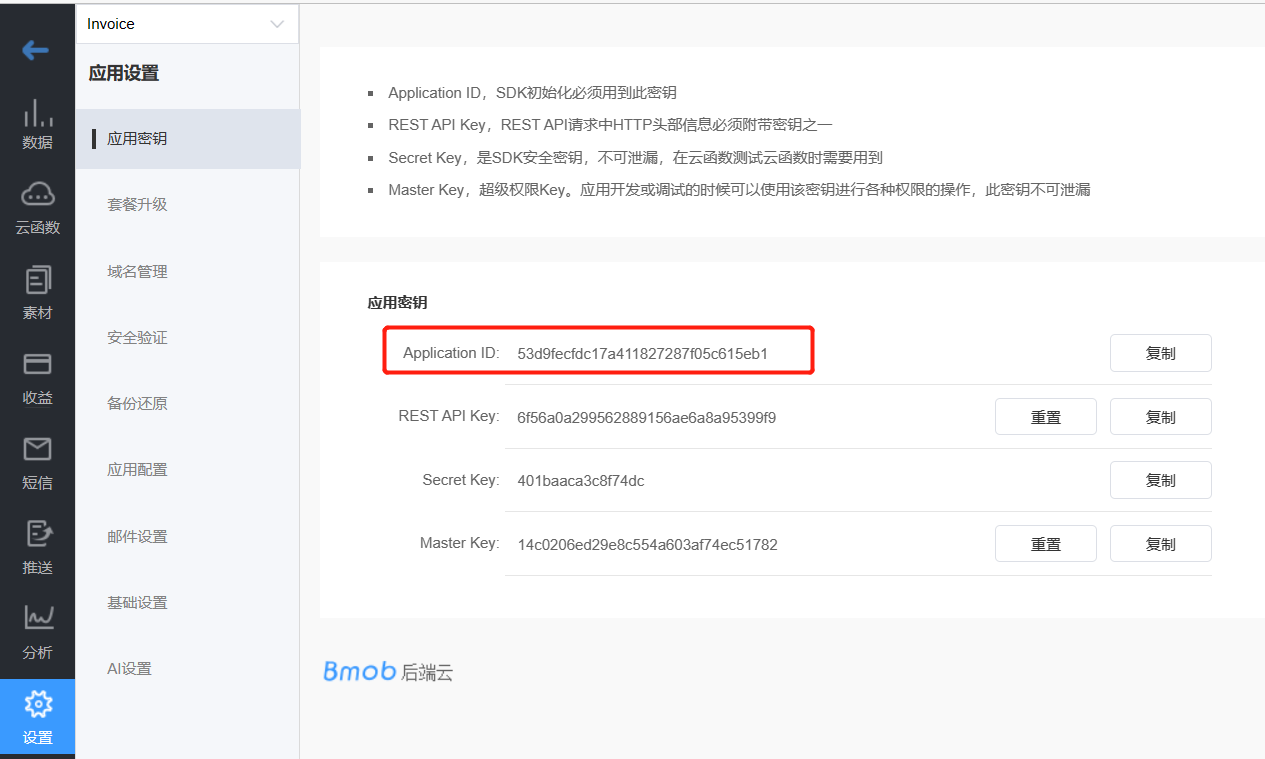
1. 数据安全性：Bomb云端存储提供了数据加密和访问控制等安全机制，可以保护用户的用户名和密码不被非法获取和篡改。
2. 数据备份和恢复：通过将用户名和密码存储在云端，即使用户的设备出现故障或丢失，用户的账号信息也可以轻松地从云端进行恢复。

3. 跨设备同步：用户可以在不同的设备上使用相同的用户名和密码，通过云端存储实现数据的同步，方便用户在不同设备上的登录和使用。

1.首先，注册Bomb账号，获取Application ID。



1. Bomb数据库控制台



1. 获取Application key

2.导入依赖

dependencies {

implementation 'io.github.bmob:android-sdk:3.8.23'

implementation 'io.reactivex.rxjava2:rxjava:2.2.8'

implementation 'io.reactivex.rxjava2:rxandroid:2.1.1'

implementation 'com.squareup.okhttp3:okhttp:4.8.1'

implementation 'com.squareup.okio:okio:2.2.2'

implementation 'com.google.code.gson:gson:2.8.5'

}

3.创建Application子类

public class BmobApp extends Application {

@Override

public void onCreate() {

super.onCreate();

Bmob.initialize(this, "你的application id");

}

}

4.配置AndroidManifest.xml

在AndroidManifest.xml文件中添加如下的应用类名、权限和ContentProvider信息：

|  |
| --- |
| <!--允许联网 -->  <uses-permission android:name="android.permission.INTERNET" />  <!--获取GSM（2g）、WCDMA（联通3g）等网络状态的信息 -->  <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  <!--获取wifi网络状态的信息 -->  <uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  <!--获取sd卡写的权限，用于文件上传和下载-->  <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  <!--允许读取手机状态 用于创建BmobInstallation-->  <uses-permission android:name="android.permission.READ\_PHONE\_STATE" />  <!—添加ContentProvider信息-->  <provider  android:name="cn.bmob.v3.util.BmobContentProvider"  android:authorities="你的应用包名.BmobContentProvider">  </provider> |

5. 用户基类

1. BombUser表的结构

|  |  |
| --- | --- |
| **属性** | **说明** |
| username | 用户名/账号/用户唯一标志 |
| password | 用户密码 |
| email | 用户邮箱 |
| emailVerified | 用户邮箱认证状态 |
| mobilePhoneNumber | 用户手机号码 |
| mobilePhoneNumberVerified | 用户手机号码认证状态 |

BmobUser继承BmobObject，有默认属性，如果用户需要其他属性，如性别、年龄、头像等，则需要继承BmobUser类进行自定义扩展。

而在本应用中，只需使用BombUser中的username和password属性，所以创建

User类继承BombUser类

1. User表的结构

|  |  |
| --- | --- |
| **属性** | **作用** |
| username | 记录用户名 |
| password | 记录用户密码 |

6. 账号密码注册及登录

分别在Login和Register活动中实现这两个功能：

1. 登录注册功能实现

|  |  |  |
| --- | --- | --- |
| **函数名** | **功能** | **代码** |
| SignUP | 调用云端数据库进行注册，并进行基本的逻辑判断，看用户输入名字和密码是否符合规范 | **private void** signUp() {  String username = **met\_user2**.getText().toString();  String pword1 = **met\_password1**.getText().toString();  String pword2 = **met\_password2**.getText().toString();  **if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***GINGERBREAD***) {  **if**(username.isEmpty())  {  Toast.*makeText*(RegisterActivity.**this**, **"用户名不能为空！"**, Toast.***LENGTH\_SHORT***).show();  **return**;  }  }  **if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***GINGERBREAD***) {  **if**(pword1.isEmpty())  {  Toast.*makeText*(RegisterActivity.**this**, **"密码不能为空！"**, Toast.***LENGTH\_SHORT***).show();  **return**;  }  }  **if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***GINGERBREAD***) {  **if**(pword2.isEmpty())  {  Toast.*makeText*(RegisterActivity.**this**, **"密码不能为空！"**, Toast.***LENGTH\_SHORT***).show();  **return**;  }  }  **if**(!pword1.equals(pword2))  {  Toast.*makeText*(RegisterActivity.**this**, **"密码两次输入不一致，请重新输入！"**, Toast.***LENGTH\_SHORT***).show();  **return**;  }  **final** User user = **new** User();  **final** Intent[] intent1 = {**null**};  user.setUsername(username);  user.setPassword(pword1);  user.signUp(**new** SaveListener<User>() {  @Override  **public void** done(User user, BmobException e) {  **if** (e == **null**) {  Toast.*makeText*(RegisterActivity.**this**,**"注册成功"**,Toast.***LENGTH\_SHORT***).show();  intent1[0] = **new** Intent(RegisterActivity.**this**, LoginActivity.**class**);  startActivity(intent1[0]);  } **else** {  Toast.*makeText*(RegisterActivity.**this**,**"注册失败"**,Toast.***LENGTH\_SHORT***).show();  Log.*d*(**"注册失败"**, **"done: "** + e.getMessage());  }  }  }); } |
| loginByAccount | 改写BmobUser中的loginByAccount方法，若登陆成功，则用Toast显示“登陆成功”并跳转到主界面，若登陆失败，则用Toast显示“登陆失败”，若输入的用户名或密码为空，也会用Toast进行提示。 | **private void** loginByAccount(**final** View view) {  **final** Intent[] intent = {**null**};  String username = **met\_user**.getText().toString();  String pword= **met\_password**.getText().toString();  **if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***GINGERBREAD***) {  **if**(username.isEmpty())  {  Toast.*makeText*(LoginActivity.**this**, **"用户名不能为空！"**, Toast.***LENGTH\_SHORT***).show();  **return**;  }  }  **if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***GINGERBREAD***) {  **if**(pword.isEmpty())  {  Toast.*makeText*(LoginActivity.**this**, **"密码不能为空！"**, Toast.***LENGTH\_SHORT***).show();  **return**;  }  }  BmobUser.loginByAccount(username, pword, **new** LogInListener<User>() {  @Override  **public void** done(User user, BmobException e) {  **if** (e == **null**) {  intent[0] = **new** Intent(LoginActivity.**this**, MainActivity.**class**);  startActivity(intent[0]);  Toast.*makeText*(LoginActivity.**this**,**"登录成功"**,Toast.***LENGTH\_SHORT***).show();  } **else** {  Toast.*makeText*(LoginActivity.**this**,**"登录失败"**,Toast.***LENGTH\_SHORT***).show();  }  }  }); } |

|  |
| --- |
|  |

* + 1. 发票数据库实现

1、先实体化一个invoicemessage类，包括发票的六个信息，分别是发票代码、发票号码、发票时间、金额、发票类别以及发票状态，我们将发票状态分为未报销和已报销两类，分别以0和1标记。

2、数据库继承自SQLiteOpenHelper，实现文件命名为userdata。

3、根据不同功能所需，在userdata.java中实现不同函数，以下为各个函数代码及功能介绍：

1. 数据库函数及功能

|  |  |  |
| --- | --- | --- |
| **函数名** | **功能** | **代码** |
| onCreate | 建数据库表 | public void onCreate(SQLiteDatabase db) {  String sql="CREATE TABLE IF NOT EXISTS "+*table\_name*+"(" +  "\_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,"+  "code VARCHAR NOT NULL," +  "number VARCHAR NOT NULL," +  "time VARCHAR NOT NULL," +  "category VARCHAR NOT NULL," +  "price FLOAT NOT NULL," +  "state INT NOT NULL);";  db.execSQL(sql); } |
| save | 保存一条发票信息，返回值大于0则保存成功 | **public long** save(invoicemessage invoice){  ContentValues cv =**new** ContentValues();  cv.put(**"code"**,invoice.**code**);  cv.put(**"number"**,invoice.**number**);  cv.put(**"time"**,invoice.**time**);  cv.put(**"category"**,invoice.**category**);  cv.put(**"price"**,invoice.**price**);  cv.put(**"state"**,invoice.**state**);  **return wRDB**.insert(***table\_name***,**null**,cv); } |
| delectbycode | 根据发票代码删除发票信息，返回值大于0则删除成功 | **public long** delectbycode(String code, **int** state){  String[] whereArgs = {code, String.*valueOf*(state)};  **return wRDB**.delete(***table\_name***, **"code=? AND state=?"**, whereArgs); } |
| selectbystate | 根据发票状态（已报销或未报销）选择发票，返回值为存储了所有查询出的发票信息的列表 | **public** List<invoicemessage> selectbystate(**int** state){  List<invoicemessage> list=**new** ArrayList<>();  String[] selectionArgs = { String.*valueOf*(state) };  Cursor cursor = **mRDB**.rawQuery(**"SELECT \* FROM massage1 WHERE state = "**+ state, **null**);  **while** (cursor.moveToNext()){  invoicemessage invoice=**new** invoicemessage();  invoice.**code**=cursor.getString(1);  invoice.**number**=cursor.getString(2);  invoice.**time**=cursor.getString(3);  invoice.**category**=cursor.getString(4);  invoice.**price**=cursor.getFloat(5);  invoice.**state**=cursor.getInt(6);  list.add(invoice);  }  **return** list; } |
| pricebystate | 根据发票状态计算这一类发票的总价格，返回值为总价格 | **public float** pricebystate(**int** state){  **float** price=0;  String[] selectionArgs = { String.*valueOf*(state) };  Cursor cursor = **mRDB**.rawQuery(**"SELECT \* FROM massage1 WHERE state = "**+ state, **null**);  **while** (cursor.moveToNext()){  price=price+cursor.getFloat(5);  }  **return** price; } |
| selectallstate | 根据发票类别和发票状态选择对应的发票，返回值为存储了所有查询出的发票的id和价格信息的列表 | **public** List<MyElement> selectallstate(**int** state,String category){  ArrayList<MyElement> list = **new** ArrayList<>();  String state1=String.*valueOf*(state);  String selection = **"state = ? AND category = ?"**;  String[] selectionArgs = { state1, category };  Cursor cursor = **mRDB**.query(***table\_name***,**null**,selection,selectionArgs,**null**,**null**,**null**);  **if** (cursor != **null** && cursor.moveToFirst()) {  **do**{  MyElement element=**new** MyElement();  element.**price**=cursor.getFloat(5);  element.**id**=cursor.getInt(0);  list.add(element);  }**while** (cursor.moveToNext());  }  **return** list; } |
| updatebyid | 根据发票存储在数据库的id值更新发票的状态，返回值大于0则更新成功 | **public long** updatebyid(**int** id){  ContentValues values = **new** ContentValues();  values.put(**"state"**, 1);  String[] selectionArgs = { String.*valueOf*(id) };  **return mRDB**.update(**"massage1"**, values, **"\_id=?"**, selectionArgs); } |
| selectbyid | 根据发票存储在数据库id值挑选对应的发票信息，返回值为存储了单个查询出的发票信息 | **public** invoicemessage selectbyid(**int** id){  Cursor cursor = **mRDB**.rawQuery(**"SELECT \* FROM massage1 WHERE \_id = "**+ id, **null**);  invoicemessage invoice = **new** invoicemessage();  **if** (cursor != **null** && cursor.moveToFirst()) {  **do**{  invoice.**code** = cursor.getString(1);  invoice.**number** = cursor.getString(2);  invoice.**time** = cursor.getString(3);  invoice.**category** = cursor.getString(4);  invoice.**price** = cursor.getFloat(5);  invoice.**state** = cursor.getInt(6);  }**while** (cursor.moveToNext());  }  **return** invoice; } |

* + 1. 蒙特卡洛算法

**算法原理**

蒙特卡洛算法的核心原理是利用随机数和概率统计方法来模拟问题，通过大量随机样本的采样，得到问题的概率分布或期望值。

**算法实现**

1. 数据库实现

为了能够实现蒙特卡洛算法，需要一个能够记录发票存储的主键id值以及price的list，新实现了一个类myelement，包含了一个float型和一个int型数据，分别记录了金额以及发票存储的id。

同时在userdata数据库中实现了一个靠id值查找发票信息的函数，返回值为invoicemessage。以及靠id值改变对应发票状态函数，还有点击ListView删除对应发票的函数，函数类型为long。

2.算法设计

在本应用中，为了得到n张所有发票金额的最优组合， 在n张发票中随机选择k张，将这k张发票总和设为sumt，与用户给定金额suma相比较，若suma小于sumt，则不符合要求重新选择k张发票；若suma大于等于sumt，则计算二者差额，差额小于10-7可直接定为最优报销组合。如此随机选择2x104次，直到选出最小差额，它所对应的组合即为最优报销组合。

代码及各函数功能如下：

1. 最优报销算法

|  |  |  |
| --- | --- | --- |
| **函数名** | **功能** | **代码** |
| MyElement | 实现一个实体类，存放即将进行最优报销算法的发票\_id值及对应的发票价格 | **public class** MyElement {  **public float price**;  **public int id**;  **public** MyElement(){  }  **public** MyElement(**float** price, **int** id) {  **this**.**price** = price;  **this**.**id** = id;  }  **public float** getPrice() {  **return this**.**price**;  }  **public int** getId() {  **return this**.**id**;  }  **public** String toString() {  **return "MyElement{"** +  **"price='"** + **price** + **'\''** +  **", id='"** + **id** + **'\''** +  **'}'**;  } } |
| excel\_MC\_combination | 最优报销算法，根据蒙特卡洛算法进行发票筛选，并将对应信息传递给下一个页面进行展示 | **public void** excel\_MC\_combination(**float** suma) {  List<MyElement> data = **new** ArrayList<>();  **for**(String item:**selectedOptions**){  data = *musedata*.selectallstate(0,item);  }  ArrayList<Float> priceList = **new** ArrayList<>();  ArrayList<Integer> idList = **new** ArrayList<>();  ArrayList<Integer> idlist1 = **new** ArrayList<>();  **for** (MyElement element : data) {  priceList.add(element.getPrice());  idList.add(element.getId());  }  **int** lendata = priceList.size();  List<Float> resdata = **new** ArrayList<>();  **double** resSt = 0.0; *// 组合加和最大值* **double** resErr = 10000.0;  **int** flag = 0; *// 用来判断是否有可以报销的发票* Random rand = **new** Random();  ArrayList<invoicemessage> list1 = **new** ArrayList<>();  **for** (**int** i = 0; i < 20000; i++) { *// 把0到19999依次赋给i，即重复20000次* List<Float> data1 = **new** ArrayList<>(priceList);  List<Float> data2 = **new** ArrayList<>();  **int** k = rand.nextInt(lendata) + 1; *// 从给定的范围返回随机项（1，lendata+1)  // // 从list中随机获取k个元素，作为一个片断返回赋给data2* **for** (**int** j = 0; j < k; j++) {  **int** randomIndex = rand.nextInt(data1.size());  data2.add(data1.get(randomIndex));  data1.remove(randomIndex);  }  **float** sumt = data2.stream().reduce(Float::*sum*).orElse((**float**) 0);  **double** ds = suma - sumt; *// ds为给定值和组合最大值的差* **if** (ds < 0) {  **continue**;  } **else** {  flag = 1;  **if** (ds < resErr) {  resErr = ds;  resdata = data2;  resSt = sumt;  **if** (ds < 0.0000001) {  **break**;  }  }  }  }  **int** index = 0;  **int** len = resdata.size();  **for** (**int** m = 0; m < len; m++) {  Float pri = resdata.get(m);  **int** k1=0;  **for**(k1=0;k1<lendata;k1++)  {  **if**(priceList.get(k1)==pri)  {  index = k1;  **break**;}  }  idlist1.add(idList.get(index).intValue());  invoicemessage invoice = *musedata*.selectbyid(idList.get(index).intValue());  list1.add(invoice);  }  String result;  **if** (flag == 0) {  result = **"根据给定金额，暂无可以报销的发票！"**;  **int** num = 0;  **tv**.setText(result);  } **else** {  result = **"共筛选出"** + resdata.size() + **"张发票，总金额为："** + Math.*round*(resSt \* 100.0) / 100.0 + **"，总金额与给定值相差："** + Math.*round*(resErr);  Intent intent = **new** Intent(quick.**this**, showinvoice2.**class**);  intent.putExtra(**"list"**, (Serializable) list1);  intent.putExtra(**"result"**, result);  intent.putIntegerArrayListExtra(**"id"**, idlist1);  startActivity(intent);  }  } } |

* + 1. 二维码扫描技术

Zxing库是一个能够在Android设备上实现二维码扫描（QR代码等）所需的库。调用zxing库，打开相机权限，利用zxing库的IntentIntegrator类打开二维码扫描，使用Zxing库提供的IntentIntegrator类parseActivityResult方法获取扫码的结果。

1. 二维码扫描

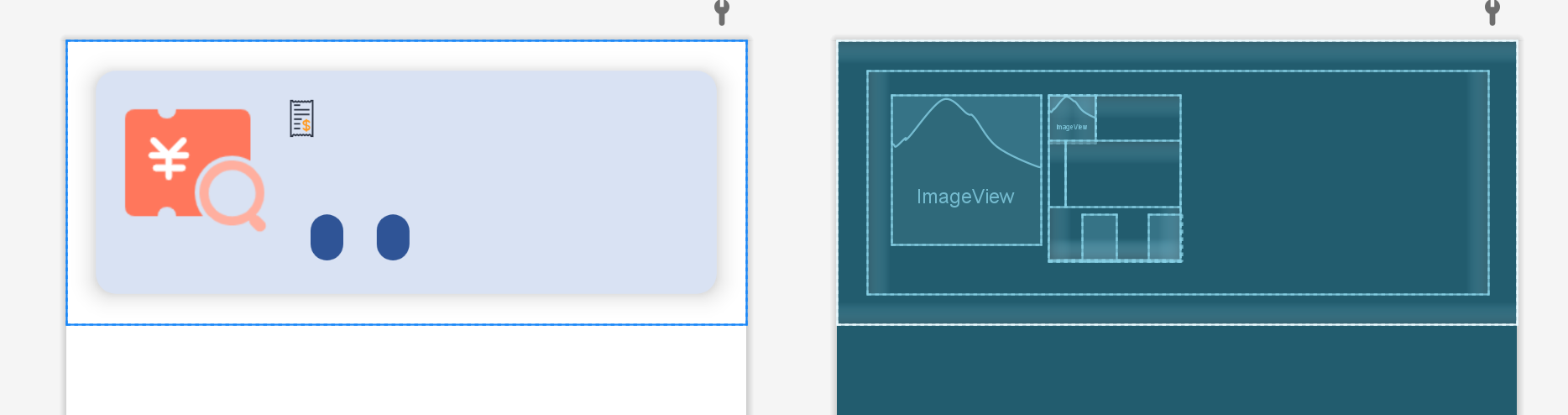
|  |  |
| --- | --- |
| **功能** | **代码实现** |
| 点击按钮，打开二维码扫描界面，通过调用zxing库的IntentIntegrator类进行扫描 | **public void** onClick(View view) {  IntentIntegrator intentIntegrator = **new** IntentIntegrator(  scan.**this** );  intentIntegrator.setPrompt(**"For flash use volume up key"**);  intentIntegrator.setBeepEnabled(**true**);  intentIntegrator.setOrientationLocked(**true**);  intentIntegrator.setCaptureActivity(Capture.**class**);  intentIntegrator.initiateScan();  } |
| 根据扫描结果，决定响应功能 | **protected void** onActivityResult(**int** requestCode, **int** resultCode, @Nullable Intent data) {  **super**.onActivityResult(requestCode, resultCode, data);  IntentResult intentResult=IntentIntegrator.*parseActivityResult*(  requestCode,resultCode,data  );  **if** (intentResult.getContents() !=**null**){  AlertDialog.Builder builder=**new** AlertDialog.Builder(  scan.**this** );  builder.setTitle(**"Result"**);  String str=intentResult.getContents().toString();  Intent intent = **new** Intent(**this**, showinvoice.**class**);  intent.putExtra(**"str"**, str);  startActivity(intent);  }**else** {  Toast.*makeText*(getApplicationContext(),**"没有扫描任何东西"**,Toast.***LENGTH\_SHORT***).show();  } } |

* + 1. ListView设计

为了将查询到的发票信息进行展示，需要利用ListView控件进行展示。

在前端方面，为了美观，将ListView固定大小为600dp，并设置android:scrollbars=“vertical”，使其可纵向滑动。在后端实现中，设计实现适配器，将数据与视图绑定，并使用ViewHolder对适配器进行优化，有效地提高列表的滚动性能，从而提升用户体验。

为了界面更加美观，多次调整item布局，加入图片、图标、阴影、圆角等元素，并调整字体风格和margin、padding等值，使界面尽可能美观。



1. 适配器单个item的ui设计
   * 1. webview展示页面

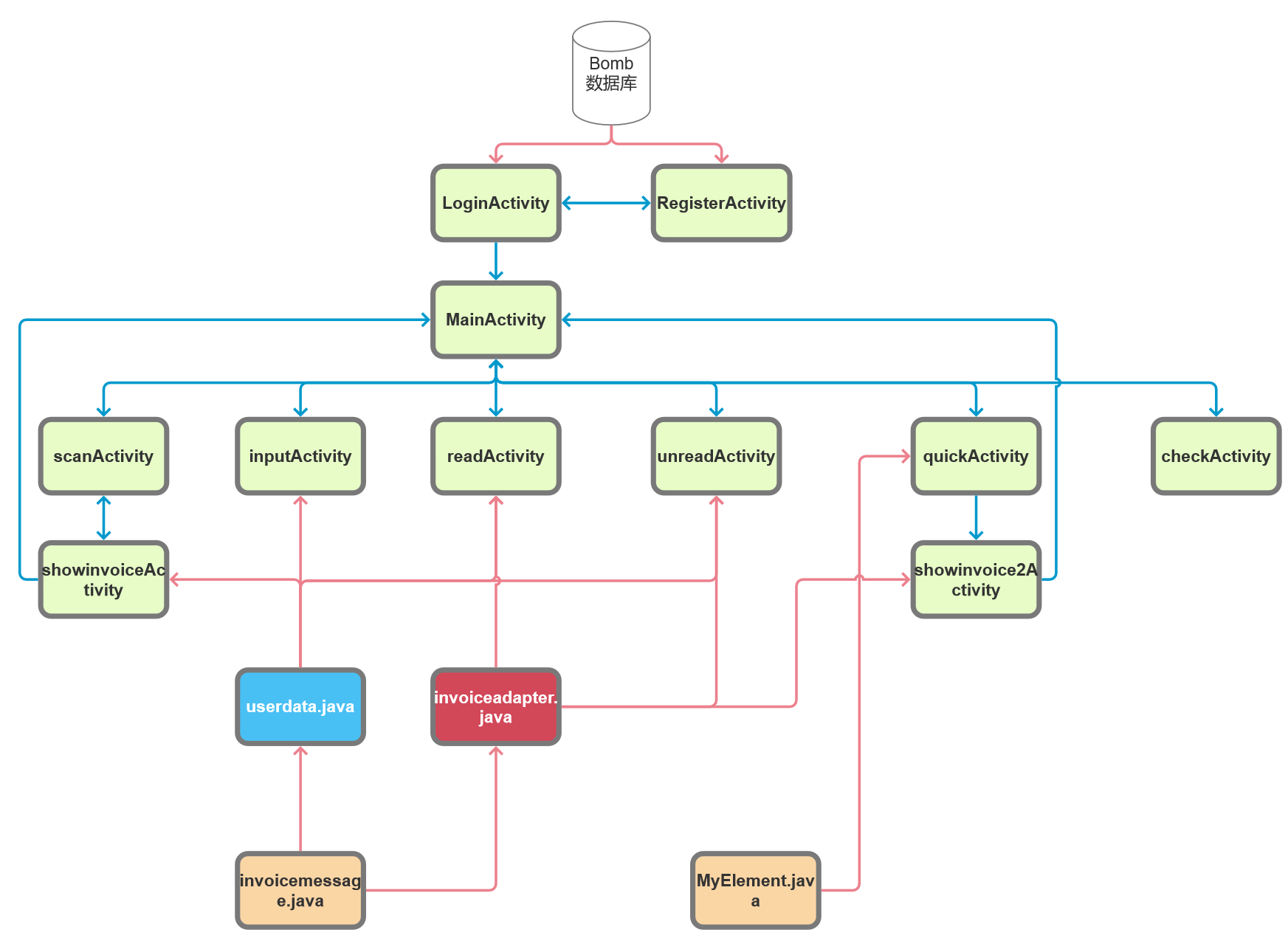
Webview主要用于手机系统来展示html界面，在本应用中为了方便用户核验发票真伪，设计一键跳转到国家税务平台，用户可输入发票信息直接进行查询。在编写代码过程中需要打开互联网权限，由于链接不安全，还要加入忽略证书错误的部分。

WebView webView = (WebView) findViewById(R.id.***web\_view***);  
webView.getSettings().setJavaScriptEnabled(**true**);*//允许Javascript*webView.setLayerType(View.***LAYER\_TYPE\_SOFTWARE***, **null**);  
webView.setWebViewClient(**new** WebViewClient(){  
 @Override  
 **public void** onReceivedSslError(WebView view, SslErrorHandler handler, SslError error) {  
 *// 忽略证书错误* handler.proceed();  
 }  
});  
webView.loadUrl(**"inv-veri.chinatax.gov.cn"**);

* 1. 代码架构

|  |  |  |
| --- | --- | --- |
| **模块名称** | **文件名** | **功能** |
| 登录 | LoginActivity.java | 实现用户登录 |
| 注册 | RegisterActivity.java | 实现用户注册 |
| 发票扫码上传 | scan.java | 调用zxing库实现二维码扫描 |
| 发票扫码上传 | showinvoice.java | 用户可查看扫描出发票的具体信息，并决定将扫描结果存入数据库 |
| 发票手动上传 | input.java | 手动输入发票代码等信息，并存入数据库 |
| 发票存储 | invoicemessage.java | 定义invoicemessage类存放发票的具体信息 |
| 发票存储 | userdata.java | 将invoicemessag类对象存放的发票信息插入SQLite数据库进行增删改查 |
| 发票查看 | InvoiceAdapter | 连接发票信息数据与ListView，使数据填充到布局 |
| 发票查看 | unread.java | 查看未报销的发票具体信息 |
| 发票查看 | read.java | 查看已报销的发票具体信息并可以选择是否要删除发票 |
| 快速筛选 | MyElement.java | 将发票信息中的id和金额提取出来作为一个新的类以便进行后续的筛选 |
| 快速筛选 | quick.java | 用户给定金额和发票类别，使用蒙特卡洛算法筛选出最优报销组合 |
| 快速筛选 | show2invoice.java | 用户可查看快速筛选出的发票的具体信息，并决定是否存入已报销 |
| 真伪查验 | check.java | 使用webview调出国家税务平台的页面以供用户查验发票真伪 |

画出代码架构图，其中浅黄绿色矩形代表各个活动，蓝色矩形代表自己实现的SQLite数据库文件，红色代表自己实现的适配器文件，橙色矩形代表自行实现的两个实体类。蓝色线代表活动间跳转，粉色线代表被调用。



1. 代码架构图
2. 应用效果
   1. 主界面

|  |  |
| --- | --- |
| 可选择的功能有：   * **发票上传**   用户可根据自己习惯选择二维码扫描或手动输入。   * **发票查看**   点击可查看已经报销和尚未报销的发票**。**   * **最优报销**   进入发票最优筛选界面，输入金额。   * **真伪查验**   跳转至国家税务平台 | 1. 主界面 |

* 1. 发票上传
     1. 手动上传

界面如图8所示，其中发票类别为单选框，设置了选择其中一个选项其他选项会被取消选中。尝试手动一张发票，上传成功页面如图9所示：

|  |  |
| --- | --- |
|  |  |
| 1. 手动上传发票界面 | 1. 手动上传发票成功界面 |
|  |  |

* + 1. 二维码扫描上传

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1. 二维码扫描 | 1. 扫描中 | 1. 扫描结果 |

|  |  |
| --- | --- |
|  |  |
| 1. 直接保存 | 1. 选择类别保存成功 |

* 1. 发票查看

从主界面进行点击，可以查看已存储上传的所有发票：

|  |  |
| --- | --- |
|  |  |
| 1. 未报销发票 | 1. 已报销发票 |

点击展示出来的单个发票信息，可以删除指定发票，并且页面会自动更新为删除后的内容，如下图所示：

|  |  |
| --- | --- |
|  |  |
| 1. 弹出窗口 | 1. 删除后更新的界面 |

* 1. 最优报销组合

通过主界面进入最优报销功能页面，第一次没有选择报销类别，会报出警告。第二次输入报销金额300，以及选择报销发票类别为交通和耗材，点击确认，程序给出结果，并向下一页跳转：

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1. 最优报销 | 1. 没有选择报销类别 | 1. 选择结果 |

查看程序筛选出来的发票，可以手动选择是否将其转移至已报销类别，若选择转移，我们可以到已报销发票界面查看，发现结果已更新：

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1. 最优报销结果 | 1. 向已报销移动 | 1. 移动结果 |

* 1. 真伪查验

点击真伪查验，即可跳转国家增值税发票查验平台进行查验。



1. 真伪查验
2. 开发过程中遇到的问题及解决办法
   1. 依赖配置不成功

在二维码扫描库的导入中，由于版本问题，一直无法正确导入。最后通过查找csdn，最终在一篇博客的评论区中找到了解决方法。

在云端数据库Bomb的api接口配置时，也是遇到了这样的问题，最后通过查看Bomb开发文档，逐个细节比对，最终连接成功。

* 1. 数据库调用问题

在我们应用的各个功能，基本都需要使用到数据库，但不同功能需要的数据库功能也不尽相同，就需要在数据库实现的Java文件中实现多个函数接口，在实现这些函数接口时，调试往往会遇到很多问题，比如：增删改查的各个命令实现格式不相同、传入的数据类型和数据库查找语句要求格式不匹配、数据库越界访问导致应用停止运行等等。

针对这些问题，我们也通过调试和查阅资料解决了。

增删改查命令格式：在csdn查找资料，查看规范化的命令格式如何写，再实现自己的数据库根据不同传入参数进行增删改查。

传入的数据类型和数据库查找语句要求格式不匹配：在传入时，统一把参数转换为String格式，并存入String格式的数组，作为查找条件，规范化操作就不会出错。

数据库越界访问：这个问题主要是查找时需要用到游标，但是一开始时由于代码写作不规范，对于游标的移动的循环条件只设置了：while (cursor.moveToNext())。这是不够的，没有判断游标是否为空，后面改成：

**if** (cursor != **null** && cursor.moveToFirst()) {  
 **do**{  
 ……..  
 }**while** (cursor.moveToNext());  
}

对游标进行判断，若为非空则移向第一行，然后每次取出指定数据后移向下一项。

1. 心得体会

我们花了大概一个月的时间来完成这次安卓开发大作业，从前期的构思选定主题，对市面上的发票软件进行调研，到真正进行代码结构设计，前后端设计，并反复调试修改优化，最终实现了所有预期功能，感觉在这个过程中收获颇丰。其中最主要的要属界面设计和功能设计。在界面设计中，需要考虑用户的使用习惯和界面的美观度，以提供良好的用户体验。在功能设计中，需要明确项目的需求，并将其分解为可实现的模块。数据库设计是项目中一个非常重要的部分，我们需要设计一个合理的数据库结构来存储和管理发票数据。在设计数据库时，考虑到查询和更新的效率，我们使用本地SQLite存储发票信息，能快速地对发票进行增删改查，提高用户的体验感，同时考虑到安全性和方便性，我们使用Bomb云端数据库存储用户的账户密码，用户在任何设备都可以登录。

通过完成这个大作业，我们不仅学到了安卓开发的基本知识和技能，还学会了如何解决问题和克服困难。我们意识到在开发过程中，遇到问题是正常的，重要的是要保持积极的态度和持续的学习精神。相信这些经验和技能将对未来发展有很大的帮助。通过这次课程和大作业，不仅提高了我们的编程能力和创新思维，也增强了我的自信心和团结协作的能力。我们觉得这是一门非常有意义且有趣的课程，让我们对安卓开发有了更深入的了解，也激发了我们深入学习安卓开发相关知识的兴趣。

1. 项目亮点

1.功能全面

我们的应用融合了发票存储，发票查看，发票定额筛选，发票验真等一条龙服务，功能强大，使用方便快捷，对于有报销需求的群体非常友好

2.界面美观

我们的应用使用listview、webview，合理布局各个控件，方便用户使用，使界面清晰美观，重点突出。

3.代码规范

在调用时多使用静态全局变量，保证了代码的健壮性，应用了课上所学的界面跳转，数据库存储，web网页展示，UI优化等知识，将理论转化为实践

1. 分工
2. 小组分工

|  |  |  |  |
| --- | --- | --- | --- |
| 组员 | 学号 | 负责内容 | 贡献百分比 |
| 梁斐然 | 20251280 | 本地数据库实现以及前端搭建 | 50% |
| 华正佳 | 20281315 | 云端数据库实现、蒙特卡洛算法实现以及真伪查验功能实现 | 50% |