**河**

**海**

**大**

**学**

**移动互联网应用开发实践**

**课程设计报告**

**设计题目：家居的智能门禁控制系统**

**小组成员：陈代超1906010231**

**陈语奇1906010232**

**王迎澳1906010309**

**葛其宇1906010308**

**邢莉娟1906010202**

**帕提麦·图尔荪1906010204**

目录

**一、需求分析1**

1. 软件功能介绍**1**
2. 用户界面需求**1**
3. 运行环境说明**1**

**二、概要设计1**

1. 数据要求说明**1**
2. 系统功能结构**2**
3. 系统业务流程**1**

**三、详细设计2**

1. 目录逻辑**2**
2. 界面设计**2**
3. 模块功能设计**7**

**四、调试分析13**

1. 调试**1**
2. 问题思考**1**

**五、课程设计总结13**

1. 小组分工**13**
2. 个人总结**14**
3. **需求分析**

**1.软件功能介绍**

1. **功能需求**

本软件设计使用高频RFID实现智能门禁功能，基本功能为：

1. 使用高频RFID技术完成读卡操作(WIFI高频RFID节点)；
2. 使用直流电机的开关模拟门禁开关(WIFI风扇(直流电机)控制节点和执行节点上的风扇)；
3. 当有刷卡动作，并正确读出卡号时，门禁打开(风扇打开)，3秒之后门禁关闭(风扇关闭)；
4. 所有传感器和执行器之间采用无线方式连接。
5. **增加功能**

同时，经过讨论，为了让系统功能更加完善，符合使用者行为逻辑，我们认为除此之外，本系统还应实现相应的用户操作和管理员操作，功能如下：

1. 管理员可以修改管理员密码，增加用户，增加管理员，查询刷卡记录。
2. 用户可以修改用户密码，查询刷卡记录，刷卡。
3. **制约规则**

在实现以上功能的前提下，业务逻辑应有以下制约规则：

1. 系统设定账号为root的管理员具有最高权限，可以进行增加、删除管理员的操作。其他管理员不具备操作权限。
2. 用户只能查询自己的刷卡记录，管理员可以查看全部的刷卡记录。

**2.用户界面需求**

1. **主界面**

用于管理员和用户进行登录。

1. **管理员操作界面**
   1. 管理员导航界面

此界面中有四个操作按钮和一个退出账户按钮，可以跳转到相应功能实现或退出账户，返回登录界面。

* 1. 修改密码界面

此界面中管理员可以修改管理员密码，首先应输入原密码，原密码正确才可以成功修改密码。

* 1. 增加用户界面

此界面中管理员需输入用户姓名，账号，密码并进行记录用户卡号，方可成功增加新用户。若操作管理员为最高权限root，则其可以直接赋予新增加用户管理员权限。

* 1. 增加管理界面

此界面中root管理员只需输入用户账号即可。

* 1. 删除管理界面

此界面中root管理员只需输入用户账号即可。

* 1. 查询记录界面

此界面可根据管理员权限筛选刷卡记录进行显示，显示信息包括卡号、用户姓名、访问时间、是否开门。

1. **用户操作界面**
2. 修改密码界面

用户修改密码界面和管理员逻辑相同。

1. 查询记录界面

此界面可根据用户权限筛选刷卡记录进行显示，显示信息包括卡号、用户姓名、访问时间、是否开门。

1. **刷卡界面**

此界面中管理员或用户可以进行刷卡，并显示是否连接成功。若连接成功则会显示卡号

**3.运行环境说明**

本项目的软件开发及运行环境如下：

1. 操作系统：Windows 10
2. 开发工具：Android Studio4.2 +Android （API 25）
3. JDK环境：Java SE Development Kit（JDK）
4. 开发语言：Java、XML
5. 运行平台：AVD （虚拟机设置）
6. **概要设计**

**1. 数据要求说明**

本系统使用Android所支持的SQLite数据库对增添的用户信息、管理员信息和刷卡使用记录进行存储。

**2. 系统功能结构**

家居的智能门禁系统的功能结构图如图1.1所示。



图1.1 家居的智能门禁系统功能结构图

**3. 系统业务流程**

1. **系统业务流程图**

家居的智能门禁系统的业务流程图如图1.2所示。



图1.2 家居的智能门禁系统的业务流程图

1. **增加用户功能工作流程图**

增加用户功能是本小组设计的家居的智能门禁系统中最为重要的功能之一，其工作流程图如图1.3所示。



图1.3 增加用户工作流程图

1. **连接功能工作流程图**

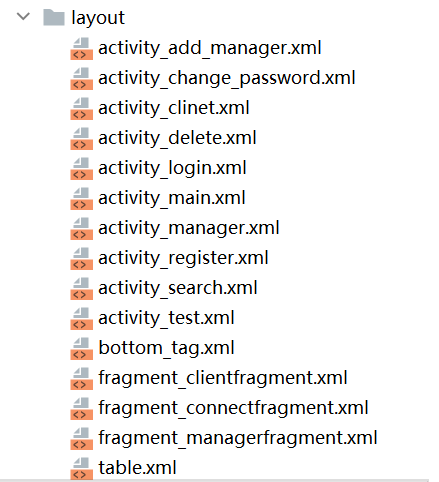
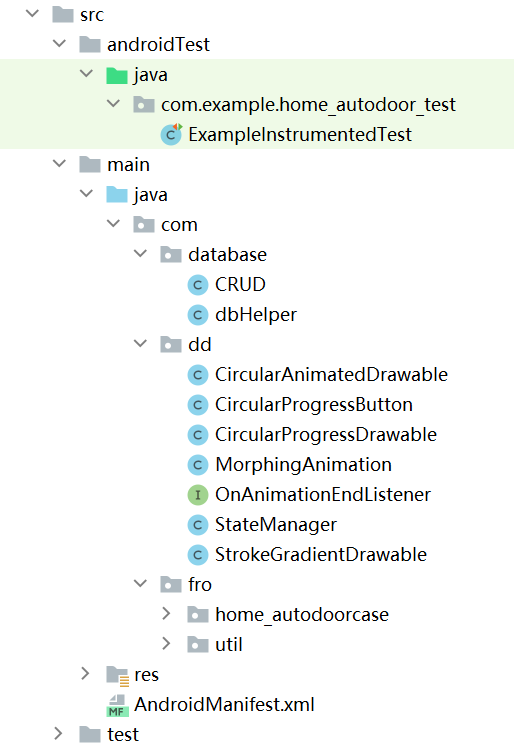
刷卡连接功能是此系统的核心功能，其工作流程图如图1.4所示。



图1.4 连接功能工作流程图

1. **详细设计**

**1. 目录逻辑**



**2. 界面设计**

1. **主界面**

主界面用于管理员和用户进行登录，拥有选择登录者身份，输入账号密码的功能，同时可以通过单击退出按钮结束程序。主界面效果图如图1.5所示。



图1.5 主界面效果图

1. **管理员操作界面**
   1. 管理员导航界面

管理员导航界面效果图，如图1.6所示。

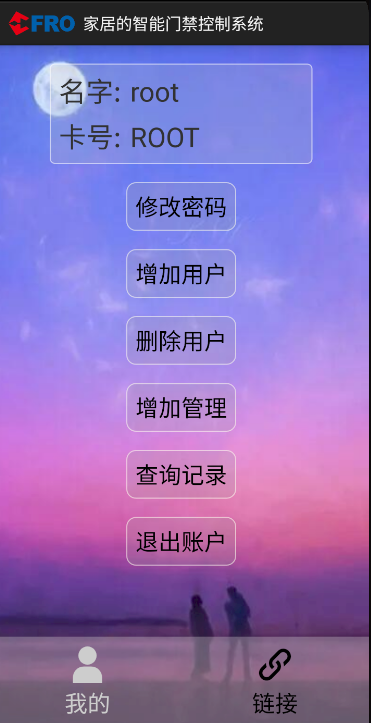


图1.6 管理员导航界面效果图

* 1. 修改密码界面



图1.7 修改密码效果图

* 1. 增加用户界面



图1.8 增加用户界面效果图

* 1. 增加/删除管理界面



图1.9 增加/删除管理界面效果图

* 1. 删除用户界面



图1.10删除用户界面效果图

* 1. 查询记录界面



图1.11 查询记录界面效果图

1. **用户操作界面**
   1. 用户导航界面

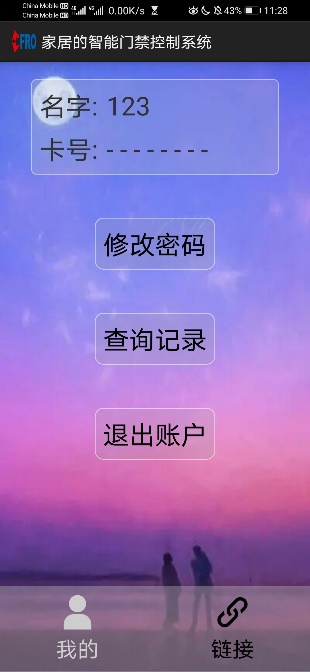


图1.12 用户导航界面效果图

* 1. 修改密码界面

同管理员修改密码界面。

* 1. 查询记录界面

同管理员查询记录界面。

1. **连接界面**

此界面中管理员或用户可以进行刷卡，并显示刷卡结果和卡号。界面效果图如图所示。



**3. 模块功能设计**

1. **数据库**
   1. 用户表

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **备注** |
| account | varchar | 用户账号 |
| name | varchar | 用户姓名 |
| password | varchar | 用户密码 |
| admin\_flag | integer | 管理员标志  （0为用户，1为管理员，2为root） |
| cardID | varchar | 用户卡号 |

* 1. 刷卡时间表

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **备注** |
| id | varchar | 刷卡卡号 |
| name | varchar | 刷卡姓名（不存在显示未知用户） |
| enter\_flag | integer | 是否成功开门（0为失败，1为成功） |
| time | varchar | 刷卡时间 |

1. **登录**
   1. 用户登录

|  |
| --- |
| int f = op.login\_check(account\_login,psd\_login,0); op.close(); if(f == 1){  Toast.*makeText*(login.this,"登陆成功",Toast.*LENGTH\_SHORT*).show();  Intent intent = new Intent(login.this,client.class);  intent.putExtra("login\_account",account\_login);  startActivity(intent);  finish(); }else if(f == 0){  Toast.*makeText*(login.this,"密码错误",Toast.*LENGTH\_SHORT*).show(); }else if(f == -1){  Toast.*makeText*(login.this,"不存在该账户",Toast.*LENGTH\_SHORT*).show(); }else{  Toast.*makeText*(login.this,"账户或密码错误",Toast.*LENGTH\_SHORT*).show(); } |

* 1. 管理员登录

|  |
| --- |
| int f = op.login\_check(account\_login,psd\_login,1); op.close(); if(f == 1){  Toast.*makeText*(login.this,"登陆成功",Toast.*LENGTH\_SHORT*).show();  Intent intent = new Intent(login.this,manager.class);  intent.putExtra("login\_account",account\_login);  startActivity(intent);  finish(); }else if(f == 0){  Toast.*makeText*(login.this,"密码错误",Toast.*LENGTH\_SHORT*).show(); }else if(f == -1){  Toast.*makeText*(login.this,"不存在该账户",Toast.*LENGTH\_SHORT*).show(); }else{  Toast.*makeText*(login.this,"账户或密码错误",Toast.*LENGTH\_SHORT*).show(); } |

1. **用户操作**
   1. 修改密码

|  |
| --- |
| public boolean updatePsd(String account, String old\_psd, String new\_psd) {  Cursor cursor = db.query(dbHelper.*USER\_TABLE*, null, "account" + "=?",  new String[]{String.*valueOf*(account)}, null, null, null, null  );  cursor.moveToFirst();  String base\_psd = cursor.getString(cursor.getColumnIndex("password"));  if (base\_psd.equals(old\_psd)) {  if(old\_psd.equals(new\_psd)){  Toast.*makeText*(context, "新密码不可以与原密码相同", Toast.*LENGTH\_SHORT*).show();  return false;  }  ContentValues contentValues = new ContentValues();  contentValues.put("password", new\_psd);  db.update(dbHelper.*USER\_TABLE*, contentValues, "account" + "=?",  new String[]{String.*valueOf*(account)});  Toast.*makeText*(context, "密码修改成功", Toast.*LENGTH\_SHORT*).show();  return true;  } else {  Toast.*makeText*(context, "原密码错误", Toast.*LENGTH\_SHORT*).show();  return false;  } } |

* 1. 查询记录

|  |
| --- |
| public ArrayList<String[]> getTimeRecord(String account){  Pair<String,String> pair = findInfoByAccount(account);  String cardID = pair.second;  Cursor cursor = db.query(dbHelper.*TIME\_TABLE*, null, "id" + "=?",  new String[]{String.*valueOf*(cardID)}, null, null, null, null  );  ArrayList<String[]> arrayList = new ArrayList<>();  if(cursor.moveToFirst()){  do{  String name = cursor.getString(cursor.getColumnIndex("name"));  int enter\_flag = cursor.getInt(cursor.getColumnIndex("enter\_flag"));  String time = cursor.getString(cursor.getColumnIndex("time"));  String newCardID = cursor.getString(cursor.getColumnIndex("id"));  arrayList.add(new String[]{newCardID,name,String.*valueOf*(enter\_flag),time});  }while(cursor.moveToNext());  }  return arrayList; }  public ArrayList<String[]> getAllTimeRecord(String account){  Pair<String,String> pair = findInfoByAccount(account);  String cardID = pair.second;  Cursor cursor = db.query(dbHelper.*TIME\_TABLE*, null, null,  null, null, null, null, null  );  ArrayList<String[]> arrayList = new ArrayList<>();  if(cursor.moveToFirst()){  do{  String name = cursor.getString(cursor.getColumnIndex("name"));  int enter\_flag = cursor.getInt(cursor.getColumnIndex("enter\_flag"));  String time = cursor.getString(cursor.getColumnIndex("time"));  String newCardID = cursor.getString(cursor.getColumnIndex("id"));  arrayList.add(new String[]{newCardID,name,String.*valueOf*(enter\_flag),time});  }while(cursor.moveToNext());  }  return arrayList; }  public void dropList(){  db.execSQL("DELETE FROM " + dbHelper.*TIME\_TABLE*); } |

1. **管理员操作**
   1. 增加用户

|  |
| --- |
| public void addUser(String account, String password, String name, String cardID, int flag) {  if (exist(account)) {  Toast.*makeText*(context, "账号已存在", Toast.*LENGTH\_SHORT*).show();  } else {  ContentValues contentValues = new ContentValues();  contentValues.put("account", account);  contentValues.put("password", password);  contentValues.put("name", name);  contentValues.put("admin\_flag", flag);  contentValues.put("cardID", cardID);  db.insert("UserTable", null, contentValues);  Toast.*makeText*(context, "账号添加成功", Toast.*LENGTH\_SHORT*).show();  } } |

* 1. 增加管理

|  |
| --- |
| package com.fro.home\_autodoorcase;  import android.app.Activity; import android.os.Bundle; import android.text.Editable; import android.text.TextWatcher; import android.view.View; import android.widget.Button; import android.widget.EditText; import android.widget.ImageButton; import android.widget.Toast;  import com.database.CRUD;  public class AddManager extends Activity {   EditText account\_et;  Button btn;  ImageButton back\_btn;   private String cur\_account;   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_add\_manager*);  back\_btn = findViewById(R.id.*add\_manager\_back\_btn*);  back\_btn.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  finish();  }  });  account\_et = findViewById(R.id.*add\_manager\_et*);  cur\_account = getIntent().getStringExtra("client\_account");  btn = findViewById(R.id.*add\_manager\_btn*);  btn.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  String account\_input = account\_et.getText().toString().trim();  if(account\_input.equals("")){  Toast.*makeText*(AddManager.this,"请输入账号",Toast.*LENGTH\_SHORT*).show();  return;  }  CRUD op = new CRUD(AddManager.this);  op.open();  int power = op.getPowerByAccount(account\_input);  if(power == -1){  Toast.*makeText*(AddManager.this,"未找到该账户",Toast.*LENGTH\_SHORT*).show();  }else if(power == 0){  String name = op.getNameByAccount(account\_input);  op.updateAdmin(account\_input,1);  Toast.*makeText*(AddManager.this,"用户"+name+"已更改为管理员",Toast.*LENGTH\_SHORT*).show();  }else if(power == 1){  if(op.getPowerByAccount(cur\_account) == 2){  op.updateAdmin(account\_input,0);  String name = op.getNameByAccount(account\_input);  Toast.*makeText*(AddManager.this,"用户"+name+"已去除管理员",Toast.*LENGTH\_SHORT*).show();  }else{  Toast.*makeText*(AddManager.this,"您没有权限",Toast.*LENGTH\_SHORT*).show();  }  }else{  Toast.*makeText*(AddManager.this,"您没有权限",Toast.*LENGTH\_SHORT*).show();  }  op.close();  account\_et.setText("");  }  });   } } |

* 1. 删除用户/管理

|  |
| --- |
| if(power == -1){  Toast.*makeText*(Delete.this,"未找到该账户",Toast.*LENGTH\_SHORT*).show(); }else if(power == 0){  String name = op.getNameByAccount(account\_input);  op.deleteUser(account\_input);  Toast.*makeText*(Delete.this,"用户"+name+"已删除",Toast.*LENGTH\_SHORT*).show(); }else if(power == 1){  if(op.getPowerByAccount(cur\_account) == 2){  String name = op.getNameByAccount(account\_input);  op.deleteUser(account\_input);  Toast.*makeText*(Delete.this,"管理员"+name+"已删除",Toast.*LENGTH\_SHORT*).show();  }else{  Toast.*makeText*(Delete.this,"您没有权限",Toast.*LENGTH\_SHORT*).show();  } }else{  Toast.*makeText*(Delete.this,"您没有权限",Toast.*LENGTH\_SHORT*).show(); } |

1. **刷卡连接**

|  |
| --- |
| package com.fro.home\_autodoorcase;  import android.animation.ValueAnimator; import android.content.Context; import android.os.AsyncTask; import android.os.Handler; import android.os.Looper; import android.util.Log; import android.view.animation.AccelerateDecelerateInterpolator; import android.widget.TextView; import android.widget.Toast;  import java.io.IOException; import java.net.InetSocketAddress; import java.net.Socket;  import com.dd.CircularProgressButton; import com.fro.util.FRORfid; import com.fro.util.StreamUtil;  */\*\*  \* Created by Jorble on 2016/3/4.  \*/* public class MyConnectTask extends AsyncTask<Void, Void, Void> {   private Context context;  TextView readResult\_tv;  TextView cardId\_tv1;  CircularProgressButton connectCircularButton;   private String cardId1;  private byte[] read\_buff;   private Socket rfidSocket;  private Socket fanSocket;   private boolean CIRCLE = false;   public MyConnectTask(Context context, TextView cardId\_tv,  CircularProgressButton connectCircularButton) {  this.context = context;  this.cardId\_tv1 = cardId\_tv;  this.connectCircularButton = connectCircularButton;  }   */\*\*  \* 更新界面  \*/* @Override  protected void onProgressUpdate(Void... values) {  if (rfidSocket != null && fanSocket != null) {   *// 设置当前进度值，100代表完成* connectCircularButton.setProgress(100);   } else {  Log.*i*(Const.*TAG*, "连接失败!");  *// 设置当前进度，-1代表失败* connectCircularButton.setProgress(-1);  }   *// 显示数据* if (Const.*CARD\_ID* != null) {  cardId\_tv1.setText(Const.*CARD\_ID*);  connectCircularButton.setText("确认卡号");  }   }   */\*\*  \* 子线程任务  \*  \* @param params  \* @return  \*/* @Override  protected Void doInBackground(Void... params) {  *// 连接* rfidSocket = getSocket(Const.*RFID\_IP*, Const.*RFID\_PORT*);  fanSocket = getSocket(Const.*FAN\_IP*, Const.*FAN\_PORT*);   *// 循环读取数据* while (CIRCLE) {  try {  *// 如果全部连接成功  // if (rfidSocket != null ) {* if (rfidSocket != null && fanSocket != null) {   *// 寻卡* StreamUtil.*writeCommand*(rfidSocket.getOutputStream(), Const.*RFID\_FIND*);  read\_buff = StreamUtil.*readData*(rfidSocket.getInputStream());*//寻卡* boolean hasRfid = FRORfid.*isFound*(read\_buff);*//寻卡* if (hasRfid) {  *// 寻卡成功后才读卡号* StreamUtil.*writeCommand*(rfidSocket.getOutputStream(), Const.*RFID\_READ*);  Thread.*sleep*(Const.*time* / 2);  read\_buff = StreamUtil.*readData*(rfidSocket.getInputStream());  cardId1 = FRORfid.*getCardId*(read\_buff);*//读卡号* if (cardId1 != null) { *// new Thread(){ // @Override // public void run() { // // TODO Auto-generated method stub // showToastByRunnable(context, "", 3000); // } // // }.start();* Const.*CARD\_ID* = cardId1;  *// 如果能读出卡号验证成功则开门* Log.*i*(Const.*TAG*, "Const.linkage=" + Const.*linkage*);  Log.*i*(Const.*TAG*, "Const.rfid=" + Const.*CARD\_ID*); *// if (Const.linkage) { // // 门禁电机 // if (!Const.isFanOn) { // StreamUtil.writeCommand(fanSocket.getOutputStream(), Const.FAN\_ON); // Thread.sleep(3000); // StreamUtil.writeCommand(fanSocket.getOutputStream(), Const.FAN\_OFF); // Thread.sleep(200); // } // }* }  }  }  *// 更新界面* publishProgress();  Thread.*sleep*(200);    } catch (IOException | InterruptedException e) {  e.printStackTrace();  }  *// 最后关闭蜂鸣器，关闭风扇 // try { // Const.isFanOn = false; // StreamUtil.writeCommand(fanSocket.getOutputStream(), Const.FAN\_OFF); // Thread.sleep(200); // } catch (IOException | InterruptedException e1) { // e1.printStackTrace(); // }* }  return null;  } *// private void showToastByRunnable(final Context context, final CharSequence text, final int duration) { // Handler handler = new Handler(Looper.getMainLooper()); // handler.post(new Runnable() { // @Override // public void run() { // } // }); // }  /\*\*  \* 建立连接并返回socket，若连接失败返回null  \*  \* @param ip  \* @param port  \* @return  \*/* private Socket getSocket(String ip, int port) {  Socket mSocket = new Socket();  InetSocketAddress mSocketAddress = new InetSocketAddress(ip, port);  *// socket连接* try {  *// 设置连接超时时间为3秒* mSocket.connect(mSocketAddress, 3000);  } catch (IOException e) {  e.printStackTrace();  }  *// 检查是否连接成功* if (mSocket.isConnected()) {  Log.*i*(Const.*TAG*, ip + "连接成功！");  return mSocket;  } else {  Log.*i*(Const.*TAG*, ip + "连接失败！");  return null;  }  }   public void setCIRCLE(boolean cIRCLE) {  CIRCLE = cIRCLE;  }   @Override  protected void onCancelled() {  connectCircularButton.setProgress(0);  }   */\*\*  \* 关闭socket  \*/* void closeSocket() {  try {  if (rfidSocket != null) {  rfidSocket.close();  }  } catch (IOException e) {  e.printStackTrace();  }  } } |

1. **调试分析**

**1. 调试**

关于界面，在手机上安装软件之后，我们对按钮点击，页面跳转，输入输出，文本和按钮的显示进行了测试并没有发现任何问题。

关于用户/管理员操作，我们添加了三种身份对操作进行测试，分别为普通用户，普通管理员，最高权限管理员，进行了增加管理员，删除管理员，查询记录的操作，验证了本软件确实实现了进行需求分析中所列出的制约条件和功能需求。

关于连接，我们分别测试了已添加的用户卡和未添加的用户卡，结果符合预期。同时，在添加用户时进行连接工作，高频RFID节点的工作状态也没有发现异常。

**2. 问题思考**

在调试的过程中，我们也遇到了很多问题，但最后都完美解决了。

关于代码整合问题，因为在开发过程中，我们并没有对开发环境和某些接口进行统一，所以在首次整合调试的过程中出现了一些麻烦，但是好在我们及时讨论，对代码中涉及到的所有接口，变量，方法进行了格式统一，也加深了我们对于规范化程序编写的理解。

关于界面显示，我们在不同型号的虚拟机上对本软件进行了测试，一些按钮因为无法适配当前屏幕尺寸出现了无法显示的问题。

关于功能操作，在首次调试过程中，我们在测试增加用户的过程中发现在未连接高频节点的wifi前提下，就算连接失败，但是依然可以成功添加用户信息，只是无法显示卡号，同时在进行刷卡时也无法成功。后来我们又在不同API和尺寸的虚拟机和另外几位同学的手机上进行了测试，并未发现相同问题。我们又对相关代码进行了检查，推测是个例，可能是手机在连接高频节点的过程中出现了信号异常。最后重新卸载安装程序，再次对此功能进行测试，并未出现相同问题。

1. **课程设计总结**

**1. 小组分工**

1. **编程**

我们首先对功能模块进行了划分，由。。完成。。。

1. **调试**

测试用例由本组成员共同讨论确立，尽量包含所有可能出现的情况。主要测试操作由邢莉娟和帕提麦·图尔荪完成，发现问题及时反给相应负责人进行更改。

1. **文档**

文档主要由邢莉娟和帕提麦·图尔荪负责撰写。帕提麦·图尔荪负责文档中的所有图片制作以及调试分析；邢莉娟负责其余部分和文档整；其余组员完成个人总结部分，并对文档进行审核。

**2. 个人总结**

1. 陈代超
2. 陈语奇
3. 葛其宇
4. 王迎澳
5. 邢莉娟
6. 帕提麦·图尔荪