|  |
| --- |
| **“学校签到管理系统”详细说明书** |

队伍名称：卖女孩的小火柴

队伍成员：张艳丽 李贻慧 邹继敏 荆心科 王瑶

所属单位：曲阜师范大学软件学院

指导老师：郭吉楠

目录

[引言 1](#_Toc32448)

[1 产品概述 1](#_Toc18674)

[1．1 编写目的 1](#_Toc17)

[1．2 背景 1](#_Toc5638)

[1．3 定义 1](#_Toc27868)

[2 程序系统的结构 1](#_Toc32294)

[2．1 客户端主程序 1](#_Toc10521)

[2．2 服务器端主程序 1](#_Toc27452)

[LoginDao： 1](#_Toc32142)

[2. 3 服务器端数据库操作 3](#_Toc5977)

[3 基本操作部分 29](#_Toc1762)

[3．1 程序描述 29](#_Toc13306)

[3．2 整体结构 29](#_Toc25161)

[3．3性能 30](#_Toc31875)

[3．4输入输出项 30](#_Toc11412)

[3．5 主要类的设计 30](#_Toc5526)

[用户类： 30](#_Toc20130)

[3．6存储分配 38](#_Toc26676)

[3．7 注释 38](#_Toc10892)

[3. 8 限制条件 38](#_Toc24262)

[3. 9 测试计划 38](#_Toc5892)

[4 其它附加功能及帮助系统 38](#_Toc23419)

[4．1 程序描述 38](#_Toc17074)

[4．2 功能 42](#_Toc15016)

[4．3 性能 51](#_Toc10149)

[4.3.1 时间特性 51](#_Toc2310)

[4.3.2 安全性 51](#_Toc14087)

[4.3.3 完整性 51](#_Toc5381)

[4.3.4 可维护性 51](#_Toc26552)

[4．4 输入输出项 52](#_Toc31626)

[4．5 限制条件 52](#_Toc26238)

[5 设计特点 52](#_Toc5078)

[5．1通信便捷 52](#_Toc23781)

[5．2 开发速度快 52](#_Toc31590)

[6 项目分工 52](#_Toc11277)

引言

在科技越来越发达的今天，关于学生在校签到情况，大多数学校仍采用人工点名等签到方式，这样不仅会耗费大量的人力和时间，而且还会存在一定的出勤误差。基于改进学生签到制度，保证学生课堂出勤率以及学生信息的管理，团队开发新型签到系统，利用相应的平台，帮助学校等各个部门更加电子化、智能化的管理学生出勤的运作，从而提高学校管理的效率。本系统的开发提供了一个在线签到系统，使用实时密码、定位、指纹进行签到，以提高出勤率的正确性。

1 产品概述

1．1 编写目的

目前现有的学校签到和管理签到信息大部分是人力进行，耗时耗力，真实性也较低，本系统旨在开发一款线上签到系统，学生在指定的条件下进行签到，教师管理者也可直接导出签到的数据，方便平时出勤率的管理，提高出勤的真实性。不仅可以使用在上课的出勤率上面，还可以使用在校园活动的出勤记录，方便快捷。

1．2 背景

随着社会的发展，计算机科学技术早已应用在生活的方方面面中，给一些传统手工带来了极大的方便。计算机应用是一种新的趋势新的潮流，在一些学校，尤其是大学平时分数也是占期末总成绩一部分，而课堂出勤率又是一个重要的评分标准，为了改变以往费时费力的人工点名签到，现我团队开发了一种学校签到系统，方便教师统计信息。

1．3 定义

（1）开发（develop）：除了单纯的开发活动外，还包括维护活动。

（2）项目（project）：向顾客交付的最终产品，包括程序及各种文档，以及开发活动所需资源经费等各种信息。

（3）项目开发计划（project development plan）：把项目与过程联系起来的计划方案。

（4）产品生命周期（product life cycle）：产品从构思到不可在使用的持续时间。

2 程序系统的结构

2．1 客户端主程序

（1）工程类型：安卓项目

（2）工程名称：“学校签到管理系统”

（3）编译生成的软件：apk形式

2．2 服务器端主程序

LoginDao：

package com.example.fouthapp.dao;  
  
import com.example.fouthapp.entity.User;  
import com.example.fouthapp.util.DBUtilPre;  
  
import java.sql.ResultSet;  
import java.sql.SQLException;  
  
public class LoginDao {  
 DBUtilPre conn = new DBUtilPre();  
  
 public User checkUser(String userId,String password){  
 User user = new User();  
 String[] params = {userId,password};  
 String sql = "select \* from user where userId=? and password=?";  
 ResultSet rs = conn.executeQueryRS(sql,params);  
  
 try {  
 while(!rs.next()){  
 user.setUserId(rs.getString("userId"));  
 user.setPassword(rs.getString("password"));  
 }  
 return user;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return null;  
 }finally {  
 if (rs!=null) {  
 try {  
 rs.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
}

RegisterDao：

package com.example.fouthapp.dao;  
  
import com.example.fouthapp.util.DBUtilPre;  
  
public class RegisterDao {  
 DBUtilPre conn = new DBUtilPre();  
  
 public boolean register(String userId,String password,String name){  
 boolean result = false;  
 String[] params = {userId, password};  
 String sql = "insert into user(userId,password) values(?,?)";  
 int count = conn.executeUpdate(sql,params);  
 if (count!=0){  
 result = true;  
 }  
  
 return result;  
 }  
}

1. 3 服务器端数据库操作

package com.example.fouthapp.util;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.ResultSetMetaData;  
import java.sql.SQLException;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
   
public class DBUtilPre {  
   
   
 private static final String *DRIVER* = "com.mysql.jdbc.Driver";  
 private static final String *URLSTR* = "jdbc:mysql://123.57.200.39:3306/sign2";  
 private static final String *USERNAME* = "root";  
 private static final String *USERPASSWORD* = "root";  
 private Connection connnection = null;   
 private PreparedStatement preparedStatement = null;  
 private ResultSet resultSet = null;  
   
 static {  
 try {  
 Class.*forName*(*DRIVER*);  
 } catch (ClassNotFoundException e) {  
 System.*out*.println("驱动连接失败");  
 System.*out*.println(e.getMessage());  
 }  
 }  
  
 public Connection getConnection() {  
 try {  
 connnection = DriverManager.*getConnection*(*URLSTR*, *USERNAME*,  
 *USERPASSWORD*);  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
 return connnection;  
 }  
 public int executeDelete(String sql,Object[] params) {  
 int affectedLine = 0;  
 try {  
 connnection = this.getConnection();  
 preparedStatement = connnection.prepareStatement(sql);  
 for (int i = 0; i < 4; i++) {  
 preparedStatement.setObject(i + 1, params[i]);  
 }  
   
 affectedLine = preparedStatement.executeUpdate();  
   
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 } finally {  
 closeAll();  
 }  
 return affectedLine;  
 }  
   
 public int executeUpdate(String sql, Object[] params) {  
 int affectedLine = 0;  
 try {  
 connnection = this.getConnection();  
 preparedStatement = connnection.prepareStatement(sql);  
 for (int i = 0; i < params.length; i++) {  
 preparedStatement.setObject(i + 1, params[i]);  
 }  
   
 affectedLine = preparedStatement.executeUpdate();  
   
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 } finally {  
 closeAll();  
 }  
 return affectedLine;  
 }  
  
   
 public ResultSet executeQueryRS(String sql, Object...params) {  
 try {  
 connnection = this.getConnection();  
 preparedStatement = connnection.prepareStatement(sql);  
 for (int i = 0; i < params.length; i++) {  
 preparedStatement.setObject(i + 1, params[i]);  
 }  
   
 resultSet = preparedStatement.executeQuery();  
   
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }   
   
 return resultSet;  
 }  
   
  
 public List<Object> excuteQuery(String sql, Object[] params)  
 {   
 ResultSet rs = executeQueryRS(sql,params);  
 ResultSetMetaData rsmd = null;  
 int columnCount = 0;  
 try {  
 rsmd = rs.getMetaData();  
 columnCount = rsmd.getColumnCount();  
 } catch (SQLException e1) {  
 System.*out*.println(e1.getMessage());  
 }  
   
 List<Object> list = new ArrayList<Object>();  
   
 try {  
 while(rs.next()) {  
 Map<String, Object> map = new HashMap<String, Object>();  
 for(int i = 1; i <= columnCount; i++) {  
 map.put(rsmd.getColumnLabel(i), rs.getObject(i));   
 }  
 list.add(map);  
 }  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 } finally {  
 closeAll();  
 }  
   
 return list;  
 }  
  
 public void closeAll() {  
   
 if (resultSet != null) {  
 try {  
 resultSet.close();  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
   
 if (preparedStatement != null) {  
 try {  
 preparedStatement.close();  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
   
 if (connnection != null) {  
 try {  
 connnection.close();  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
 }  
}

Acache:

package com.example.fouthapp.util.cache;  
  
import android.content.Context;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.graphics.Canvas;  
import android.graphics.PixelFormat;  
import android.graphics.drawable.BitmapDrawable;  
import android.graphics.drawable.Drawable;  
  
import org.json.JSONArray;  
import org.json.JSONObject;  
  
import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.ByteArrayInputStream;  
import java.io.ByteArrayOutputStream;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
import java.io.OutputStream;  
import java.io.RandomAccessFile;  
import java.io.Serializable;  
import java.util.Collections;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Map.Entry;  
import java.util.Set;  
import java.util.concurrent.atomic.AtomicInteger;  
import java.util.concurrent.atomic.AtomicLong;  
  
public class ACache {  
 public static final int *TIME\_HOUR* = 60 \* 60;  
 public static final int *TIME\_DAY* = *TIME\_HOUR* \* 24;  
 private static final int *MAX\_SIZE* = 1000 \* 1000 \* 50; // 50 mb  
 private static final int *MAX\_COUNT* = Integer.*MAX\_VALUE*; // 不限制存放数据的数量  
 private static Map<String, ACache> *mInstanceMap* = new HashMap<String, ACache>();  
 private ACacheManager mCache;  
  
 public static ACache get(Context ctx) {  
 return *get*(ctx, "ACache");  
 }  
  
 public static ACache get(Context ctx, String cacheName) {  
 File f = new File(ctx.getCacheDir(), cacheName);  
 return *get*(f, *MAX\_SIZE*, *MAX\_COUNT*);  
 }  
  
 public static ACache get(File cacheDir) {  
 return *get*(cacheDir, *MAX\_SIZE*, *MAX\_COUNT*);  
 }  
  
 public static ACache get(Context ctx, long max\_zise, int max\_count) {  
 File f = new File(ctx.getCacheDir(), "ACache");  
 return *get*(f, max\_zise, max\_count);  
 }  
  
 public static ACache get(File cacheDir, long max\_zise, int max\_count) {  
 ACache manager = *mInstanceMap*.get(cacheDir.getAbsoluteFile() + *myPid*());  
 if (manager == null) {  
 manager = new ACache(cacheDir, max\_zise, max\_count);  
 *mInstanceMap*.put(cacheDir.getAbsolutePath() + *myPid*(), manager);  
 }  
 return manager;  
 }  
  
 private static String myPid() {  
 return "\_" + android.os.Process.*myPid*();  
 }  
  
 private ACache(File cacheDir, long max\_size, int max\_count) {  
 if (!cacheDir.exists() && !cacheDir.mkdirs()) {  
 throw new RuntimeException("can't make dirs in " + cacheDir.getAbsolutePath());  
 }  
 mCache = new ACacheManager(cacheDir, max\_size, max\_count);  
 }  
  
 */\*\*  
 \* Provides a means to save a cached file before the data are available.  
 \* Since writing about the file is complete, and its close method is called,  
 \* its contents will be registered in the cache. Example of use:  
 \*   
 \* ACache cache = new ACache(this) try { OutputStream stream =  
 \* cache.put("myFileName") stream.write("some bytes".getBytes()); // now  
 \* update cache! stream.close(); } catch(FileNotFoundException e){  
 \* e.printStackTrace() }  
 \*/* class xFileOutputStream extends FileOutputStream {  
 File file;  
  
 public xFileOutputStream(File file) throws FileNotFoundException {  
 super(file);  
 this.file = file;  
 }  
  
 public void close() throws IOException {  
 super.close();  
 mCache.put(file);  
 }  
 }  
  
 // =======================================  
 // ============ String数据 读写 ==============  
 // =======================================  
 */\*\*  
 \* 保存 String数据 到 缓存中  
 \*   
 \** ***@param*** *key  
 \* 保存的key  
 \** ***@param*** *value  
 \* 保存的String数据  
 \*/* public void put(String key, String value) {  
 File file = mCache.newFile(key);  
 BufferedWriter out = null;  
 try {  
 out = new BufferedWriter(new FileWriter(file), 1024);  
 out.write(value);  
 } catch (IOException e) {  
 e.printStackTrace();  
 } finally {  
 if (out != null) {  
 try {  
 out.flush();  
 out.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 mCache.put(file);  
 }  
 }  
  
 */\*\*  
 \* 保存 String数据 到 缓存中  
 \*   
 \** ***@param*** *key  
 \* 保存的key  
 \** ***@param*** *value  
 \* 保存的String数据  
 \** ***@param*** *saveTime  
 \* 保存的时间，单位：秒  
 \*/* public void put(String key, String value, int saveTime) {  
 put(key, Utils.*newStringWithDateInfo*(saveTime, value));  
 }  
  
 */\*\*  
 \* 读取 String数据  
 \*   
 \** ***@param*** *key  
 \** ***@return*** *String 数据  
 \*/* public String getAsString(String key) {  
 File file = mCache.get(key);  
 if (!file.exists())  
 return null;  
 boolean removeFile = false;  
 BufferedReader in = null;  
 try {  
 in = new BufferedReader(new FileReader(file));  
 String readString = "";  
 String currentLine;  
 while ((currentLine = in.readLine()) != null) {  
 readString += currentLine;  
 }  
 if (!Utils.*isDue*(readString)) {  
 return Utils.*clearDateInfo*(readString);  
 } else {  
 removeFile = true;  
 return null;  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 return null;  
 } finally {  
 if (in != null) {  
 try {  
 in.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 if (removeFile)  
 remove(key);  
 }  
 }  
  
 // =======================================  
 // ============= JSONObject 数据 读写 ==============  
 // =======================================  
 */\*\*  
 \* 保存 JSONObject数据 到 缓存中  
 \*   
 \** ***@param*** *key  
 \* 保存的key  
 \** ***@param*** *value  
 \* 保存的JSON数据  
 \*/* public void put(String key, JSONObject value) {  
 put(key, value.toString());  
 }  
  
 */\*\*  
 \* 保存 JSONObject数据 到 缓存中  
 \*   
 \** ***@param*** *key  
 \* 保存的key  
 \** ***@param*** *value  
 \* 保存的JSONObject数据  
 \** ***@param*** *saveTime  
 \* 保存的时间，单位：秒  
 \*/* public void put(String key, JSONObject value, int saveTime) {  
 put(key, value.toString(), saveTime);  
 }  
  
 */\*\*  
 \* 读取JSONObject数据  
 \*   
 \** ***@param*** *key  
 \** ***@return*** *JSONObject数据  
 \*/* public JSONObject getAsJSONObject(String key) {  
 String JSONString = getAsString(key);  
 try {  
 JSONObject obj = new JSONObject(JSONString);  
 return obj;  
 } catch (Exception e) {  
 e.printStackTrace();  
 return null;  
 }  
 }  
  
 // =======================================  
 // ============ JSONArray 数据 读写 =============  
 // =======================================  
 */\*\*  
 \* 保存 JSONArray数据 到 缓存中  
 \*   
 \** ***@param*** *key  
 \* 保存的key  
 \** ***@param*** *value  
 \* 保存的JSONArray数据  
 \*/* public void put(String key, JSONArray value) {  
 put(key, value.toString());  
 }  
  
 */\*\*  
 \* 保存 JSONArray数据 到 缓存中  
 \*   
 \** ***@param*** *key  
 \* 保存的key  
 \** ***@param*** *value  
 \* 保存的JSONArray数据  
 \** ***@param*** *saveTime  
 \* 保存的时间，单位：秒  
 \*/* public void put(String key, JSONArray value, int saveTime) {  
 put(key, value.toString(), saveTime);  
 }  
  
 */\*\*  
 \* 读取JSONArray数据  
 \*   
 \** ***@param*** *key  
 \** ***@return*** *JSONArray数据  
 \*/* public JSONArray getAsJSONArray(String key) {  
 String JSONString = getAsString(key);  
 try {  
 JSONArray obj = new JSONArray(JSONString);  
 return obj;  
 } catch (Exception e) {  
 e.printStackTrace();  
 return null;  
 }  
 }  
  
 // =======================================  
 // ============== byte 数据 读写 =============  
 // =======================================  
 */\*\*  
 \* 保存 byte数据 到 缓存中  
 \*   
 \** ***@param*** *key  
 \* 保存的key  
 \** ***@param*** *value  
 \* 保存的数据  
 \*/* public void put(String key, byte[] value) {  
 File file = mCache.newFile(key);  
 FileOutputStream out = null;  
 try {  
 out = new FileOutputStream(file);  
 out.write(value);  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 if (out != null) {  
 try {  
 out.flush();  
 out.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 mCache.put(file);  
 }  
 }  
  
 */\*\*  
 \* Cache for a stream  
 \*   
 \** ***@param*** *key  
 \* the file name.  
 \** ***@return*** *OutputStream stream for writing data.  
 \** ***@throws*** *FileNotFoundException  
 \* if the file can not be created.  
 \*/* public OutputStream put(String key) throws FileNotFoundException {  
 return new xFileOutputStream(mCache.newFile(key));  
 }  
  
 */\*\*  
 \*   
 \** ***@param*** *key  
 \* the file name.  
 \** ***@return*** *(InputStream or null) stream previously saved in cache.  
 \** ***@throws*** *FileNotFoundException  
 \* if the file can not be opened  
 \*/* public InputStream get(String key) throws FileNotFoundException {  
 File file = mCache.get(key);  
 if (!file.exists())  
 return null;  
 return new FileInputStream(file);  
 }  
  
 */\*\*  
 \* 保存 byte数据 到 缓存中  
 \*   
 \** ***@param*** *key  
 \* 保存的key  
 \** ***@param*** *value  
 \* 保存的数据  
 \** ***@param*** *saveTime  
 \* 保存的时间，单位：秒  
 \*/* public void put(String key, byte[] value, int saveTime) {  
 put(key, Utils.*newByteArrayWithDateInfo*(saveTime, value));  
 }  
  
 */\*\*  
 \* 获取 byte 数据  
 \*   
 \** ***@param*** *key  
 \** ***@return*** *byte 数据  
 \*/* public byte[] getAsBinary(String key) {  
 RandomAccessFile RAFile = null;  
 boolean removeFile = false;  
 try {  
 File file = mCache.get(key);  
 if (!file.exists())  
 return null;  
 RAFile = new RandomAccessFile(file, "r");  
 byte[] byteArray = new byte[(int) RAFile.length()];  
 RAFile.read(byteArray);  
 if (!Utils.*isDue*(byteArray)) {  
 return Utils.*clearDateInfo*(byteArray);  
 } else {  
 removeFile = true;  
 return null;  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 return null;  
 } finally {  
 if (RAFile != null) {  
 try {  
 RAFile.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 if (removeFile)  
 remove(key);  
 }  
 }  
  
 // =======================================  
 // ============= 序列化 数据 读写 ===============  
 // =======================================  
 */\*\*  
 \* 保存 Serializable数据 到 缓存中  
 \*   
 \** ***@param*** *key  
 \* 保存的key  
 \** ***@param*** *value  
 \* 保存的value  
 \*/* public void put(String key, Serializable value) {  
 put(key, value, -1);  
 }  
  
 */\*\*  
 \* 保存 Serializable数据到 缓存中  
 \*   
 \** ***@param*** *key  
 \* 保存的key  
 \** ***@param*** *value  
 \* 保存的value  
 \** ***@param*** *saveTime  
 \* 保存的时间，单位：秒  
 \*/* public void put(String key, Serializable value, int saveTime) {  
 ByteArrayOutputStream baos = null;  
 ObjectOutputStream oos = null;  
 try {  
 baos = new ByteArrayOutputStream();  
 oos = new ObjectOutputStream(baos);  
 oos.writeObject(value);  
 byte[] data = baos.toByteArray();  
 if (saveTime != -1) {  
 put(key, data, saveTime);  
 } else {  
 put(key, data);  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 oos.close();  
 } catch (IOException e) {  
 }  
 }  
 }  
  
 */\*\*  
 \* 读取 Serializable数据  
 \*   
 \** ***@param*** *key  
 \** ***@return*** *Serializable 数据  
 \*/* public Object getAsObject(String key) {  
 byte[] data = getAsBinary(key);  
 if (data != null) {  
 ByteArrayInputStream bais = null;  
 ObjectInputStream ois = null;  
 try {  
 bais = new ByteArrayInputStream(data);  
 ois = new ObjectInputStream(bais);  
 Object reObject = ois.readObject();  
 return reObject;  
 } catch (Exception e) {  
 e.printStackTrace();  
 return null;  
 } finally {  
 try {  
 if (bais != null)  
 bais.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 try {  
 if (ois != null)  
 ois.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 return null;  
  
 }  
  
 // =======================================  
 // ============== bitmap 数据 读写 =============  
 // =======================================  
 */\*\*  
 \* 保存 bitmap 到 缓存中  
 \*   
 \** ***@param*** *key  
 \* 保存的key  
 \** ***@param*** *value  
 \* 保存的bitmap数据  
 \*/* public void put(String key, Bitmap value) {  
 put(key, Utils.*Bitmap2Bytes*(value));  
 }  
  
 */\*\*  
 \* 保存 bitmap 到 缓存中  
 \*   
 \** ***@param*** *key  
 \* 保存的key  
 \** ***@param*** *value  
 \* 保存的 bitmap 数据  
 \** ***@param*** *saveTime  
 \* 保存的时间，单位：秒  
 \*/* public void put(String key, Bitmap value, int saveTime) {  
 put(key, Utils.*Bitmap2Bytes*(value), saveTime);  
 }  
  
 */\*\*  
 \* 读取 bitmap 数据  
 \*   
 \** ***@param*** *key  
 \** ***@return*** *bitmap 数据  
 \*/* public Bitmap getAsBitmap(String key) {  
 if (getAsBinary(key) == null) {  
 return null;  
 }  
 return Utils.*Bytes2Bimap*(getAsBinary(key));  
 }  
  
 // =======================================  
 // ============= drawable 数据 读写 =============  
 // =======================================  
 */\*\*  
 \* 保存 drawable 到 缓存中  
 \*   
 \** ***@param*** *key  
 \* 保存的key  
 \** ***@param*** *value  
 \* 保存的drawable数据  
 \*/* public void put(String key, Drawable value) {  
 put(key, Utils.*drawable2Bitmap*(value));  
 }  
  
 */\*\*  
 \* 保存 drawable 到 缓存中  
 \*   
 \** ***@param*** *key  
 \* 保存的key  
 \** ***@param*** *value  
 \* 保存的 drawable 数据  
 \** ***@param*** *saveTime  
 \* 保存的时间，单位：秒  
 \*/* public void put(String key, Drawable value, int saveTime) {  
 put(key, Utils.*drawable2Bitmap*(value), saveTime);  
 }  
  
 */\*\*  
 \* 读取 Drawable 数据  
 \*   
 \** ***@param*** *key  
 \** ***@return*** *Drawable 数据  
 \*/* public Drawable getAsDrawable(String key) {  
 if (getAsBinary(key) == null) {  
 return null;  
 }  
 return Utils.*bitmap2Drawable*(Utils.*Bytes2Bimap*(getAsBinary(key)));  
 }  
  
 */\*\*  
 \* 获取缓存文件  
 \*   
 \** ***@param*** *key  
 \** ***@return*** *value 缓存的文件  
 \*/* public File file(String key) {  
 File f = mCache.newFile(key);  
 if (f.exists())  
 return f;  
 return null;  
 }  
  
 */\*\*  
 \* 移除某个key  
 \*   
 \** ***@param*** *key  
 \** ***@return*** *是否移除成功  
 \*/* public boolean remove(String key) {  
 return mCache.remove(key);  
 }  
  
 */\*\*  
 \* 清除所有数据  
 \*/* public void clear() {  
 mCache.clear();  
 }  
  
 */\*\*  
 \** ***@title*** *缓存管理器  
 \** ***@version*** *1.0  
 \*/* public class ACacheManager {  
 private final AtomicLong cacheSize;  
 private final AtomicInteger cacheCount;  
 private final long sizeLimit;  
 private final int countLimit;  
 private final Map<File, Long> lastUsageDates = Collections.*synchronizedMap*(new HashMap<File, Long>());  
 protected File cacheDir;  
  
 private ACacheManager(File cacheDir, long sizeLimit, int countLimit) {  
 this.cacheDir = cacheDir;  
 this.sizeLimit = sizeLimit;  
 this.countLimit = countLimit;  
 cacheSize = new AtomicLong();  
 cacheCount = new AtomicInteger();  
 calculateCacheSizeAndCacheCount();  
 }  
  
 */\*\*  
 \* 计算 cacheSize和cacheCount  
 \*/* private void calculateCacheSizeAndCacheCount() {  
 new Thread(new Runnable() {  
 @Override  
 public void run() {  
 int size = 0;  
 int count = 0;  
 File[] cachedFiles = cacheDir.listFiles();  
 if (cachedFiles != null) {  
 for (File cachedFile : cachedFiles) {  
 size += calculateSize(cachedFile);  
 count += 1;  
 lastUsageDates.put(cachedFile, cachedFile.lastModified());  
 }  
 cacheSize.set(size);  
 cacheCount.set(count);  
 }  
 }  
 }).start();  
 }  
  
 private void put(File file) {  
 int curCacheCount = cacheCount.get();  
 while (curCacheCount + 1 > countLimit) {  
 long freedSize = removeNext();  
 cacheSize.addAndGet(-freedSize);  
  
 curCacheCount = cacheCount.addAndGet(-1);  
 }  
 cacheCount.addAndGet(1);  
  
 long valueSize = calculateSize(file);  
 long curCacheSize = cacheSize.get();  
 while (curCacheSize + valueSize > sizeLimit) {  
 long freedSize = removeNext();  
 curCacheSize = cacheSize.addAndGet(-freedSize);  
 }  
 cacheSize.addAndGet(valueSize);  
  
 Long currentTime = System.*currentTimeMillis*();  
 file.setLastModified(currentTime);  
 lastUsageDates.put(file, currentTime);  
 }  
  
 private File get(String key) {  
 File file = newFile(key);  
 Long currentTime = System.*currentTimeMillis*();  
 file.setLastModified(currentTime);  
 lastUsageDates.put(file, currentTime);  
  
 return file;  
 }  
  
 private File newFile(String key) {  
 return new File(cacheDir, key.hashCode() + "");  
 }  
  
 private boolean remove(String key) {  
 File image = get(key);  
 return image.delete();  
 }  
  
 private void clear() {  
 lastUsageDates.clear();  
 cacheSize.set(0);  
 File[] files = cacheDir.listFiles();  
 if (files != null) {  
 for (File f : files) {  
 f.delete();  
 }  
 }  
 }  
  
 */\*\*  
 \* 移除旧的文件  
 \*   
 \** ***@return*** *\*/* private long removeNext() {  
 if (lastUsageDates.isEmpty()) {  
 return 0;  
 }  
  
 Long oldestUsage = null;  
 File mostLongUsedFile = null;  
 Set<Entry<File, Long>> entries = lastUsageDates.entrySet();  
 synchronized (lastUsageDates) {  
 for (Entry<File, Long> entry : entries) {  
 if (mostLongUsedFile == null) {  
 mostLongUsedFile = entry.getKey();  
 oldestUsage = entry.getValue();  
 } else {  
 Long lastValueUsage = entry.getValue();  
 if (lastValueUsage < oldestUsage) {  
 oldestUsage = lastValueUsage;  
 mostLongUsedFile = entry.getKey();  
 }  
 }  
 }  
 }  
  
 long fileSize = calculateSize(mostLongUsedFile);  
 if (mostLongUsedFile.delete()) {  
 lastUsageDates.remove(mostLongUsedFile);  
 }  
 return fileSize;  
 }  
  
 private long calculateSize(File file) {  
 return file.length();  
 }  
 }  
  
 */\*\*  
 \** ***@title*** *时间计算工具类  
 \** ***@author*** *杨福海（michael） www.yangfuhai.com  
 \** ***@version*** *1.0  
 \*/* private static class Utils {  
  
 */\*\*  
 \* 判断缓存的String数据是否到期  
 \*   
 \** ***@param*** *str  
 \** ***@return*** *true：到期了 false：还没有到期  
 \*/* private static boolean isDue(String str) {  
 return *isDue*(str.getBytes());  
 }  
  
 */\*\*  
 \* 判断缓存的byte数据是否到期  
 \*   
 \** ***@param*** *data  
 \** ***@return*** *true：到期了 false：还没有到期  
 \*/* private static boolean isDue(byte[] data) {  
 String[] strs = *getDateInfoFromDate*(data);  
 if (strs != null && strs.length == 2) {  
 String saveTimeStr = strs[0];  
 while (saveTimeStr.startsWith("0")) {  
 saveTimeStr = saveTimeStr.substring(1, saveTimeStr.length());  
 }  
 long saveTime = Long.*valueOf*(saveTimeStr);  
 long deleteAfter = Long.*valueOf*(strs[1]);  
 if (System.*currentTimeMillis*() > saveTime + deleteAfter \* 1000) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 private static String newStringWithDateInfo(int second, String strInfo) {  
 return *createDateInfo*(second) + strInfo;  
 }  
  
 private static byte[] newByteArrayWithDateInfo(int second, byte[] data2) {  
 byte[] data1 = *createDateInfo*(second).getBytes();  
 byte[] retdata = new byte[data1.length + data2.length];  
 System.*arraycopy*(data1, 0, retdata, 0, data1.length);  
 System.*arraycopy*(data2, 0, retdata, data1.length, data2.length);  
 return retdata;  
 }  
  
 private static String clearDateInfo(String strInfo) {  
 if (strInfo != null && *hasDateInfo*(strInfo.getBytes())) {  
 strInfo = strInfo.substring(strInfo.indexOf(*mSeparator*) + 1, strInfo.length());  
 }  
 return strInfo;  
 }  
  
 private static byte[] clearDateInfo(byte[] data) {  
 if (*hasDateInfo*(data)) {  
 return *copyOfRange*(data, *indexOf*(data, *mSeparator*) + 1, data.length);  
 }  
 return data;  
 }  
  
 private static boolean hasDateInfo(byte[] data) {  
 return data != null && data.length > 15 && data[13] == '-' && *indexOf*(data, *mSeparator*) > 14;  
 }  
  
 private static String[] getDateInfoFromDate(byte[] data) {  
 if (*hasDateInfo*(data)) {  
 String saveDate = new String(*copyOfRange*(data, 0, 13));  
 String deleteAfter = new String(*copyOfRange*(data, 14, *indexOf*(data, *mSeparator*)));  
 return new String[] { saveDate, deleteAfter };  
 }  
 return null;  
 }  
  
 private static int indexOf(byte[] data, char c) {  
 for (int i = 0; i < data.length; i++) {  
 if (data[i] == c) {  
 return i;  
 }  
 }  
 return -1;  
 }  
  
 private static byte[] copyOfRange(byte[] original, int from, int to) {  
 int newLength = to - from;  
 if (newLength < 0)  
 throw new IllegalArgumentException(from + " > " + to);  
 byte[] copy = new byte[newLength];  
 System.*arraycopy*(original, from, copy, 0, Math.*min*(original.length - from, newLength));  
 return copy;  
 }  
  
 private static final char *mSeparator* = ' ';  
  
 private static String createDateInfo(int second) {  
 String currentTime = System.*currentTimeMillis*() + "";  
 while (currentTime.length() < 13) {  
 currentTime = "0" + currentTime;  
 }  
 return currentTime + "-" + second + *mSeparator*;  
 }  
  
 /\*  
 \* Bitmap → byte[]  
 \*/  
 private static byte[] Bitmap2Bytes(Bitmap bm) {  
 if (bm == null) {  
 return null;  
 }  
 ByteArrayOutputStream baos = new ByteArrayOutputStream();  
 bm.compress(Bitmap.CompressFormat.*PNG*, 100, baos);  
 return baos.toByteArray();  
 }  
  
 /\*  
 \* byte[] → Bitmap  
 \*/  
 private static Bitmap Bytes2Bimap(byte[] b) {  
 if (b.length == 0) {  
 return null;  
 }  
 return BitmapFactory.*decodeByteArray*(b, 0, b.length);  
 }  
  
 /\*  
 \* Drawable → Bitmap  
 \*/  
 private static Bitmap drawable2Bitmap(Drawable drawable) {  
 if (drawable == null) {  
 return null;  
 }  
 // 取 drawable 的长宽  
 int w = drawable.getIntrinsicWidth();  
 int h = drawable.getIntrinsicHeight();  
 // 取 drawable 的颜色格式  
 Bitmap.Config config = drawable.getOpacity() != PixelFormat.*OPAQUE* ? Bitmap.Config.*ARGB\_8888* : Bitmap.Config.*RGB\_565*;  
 // 建立对应 bitmap  
 Bitmap bitmap = Bitmap.*createBitmap*(w, h, config);  
 // 建立对应 bitmap 的画布  
 Canvas canvas = new Canvas(bitmap);  
 drawable.setBounds(0, 0, w, h);  
 // 把 drawable 内容画到画布中  
 drawable.draw(canvas);  
 return bitmap;  
 }  
  
 /\*  
 \* Bitmap → Drawable  
 \*/  
 @SuppressWarnings("deprecation")  
 private static Drawable bitmap2Drawable(Bitmap bm) {  
 if (bm == null) {  
 return null;  
 }  
 BitmapDrawable bd=new BitmapDrawable(bm);  
 bd.setTargetDensity(bm.getDensity());  
 return new BitmapDrawable(bm);  
 }  
 }  
  
}

3 基本操作部分

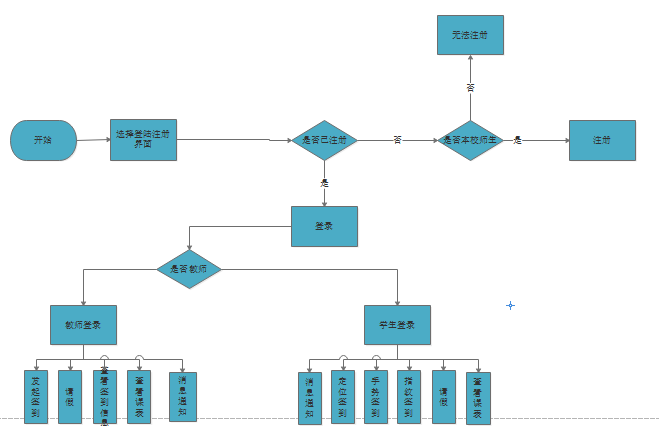
3．1 程序描述

（1） 客户端窗体 大气简约尽量美观的设计，让用户清晰明了的使用软件，简单易懂。

（2）服务端设计 尽量符合人们的习惯，并在非法操作的情况下有相应的提示。

3．2 整体结构

如下为系统流程图



3．3性能

灵活性：窗口响应绝大部分的快捷菜单和控制面板操作。

时间特性：响应鼠标单击的时间在2~3秒之间。

3．4输入输出项

输入的数据是用户执行的各种操作，包括触屏、键盘等操作。

输出数据是从数据库中读取的数据，包括错误警告信息。

3．5 主要类的设计

用户类：

package com.example.fouthapp.entity;  
  
  
import org.litepal.crud.LitePalSupport;  
  
  
public class User extends LitePalSupport {  
 private String userId;  
 private String password;  
  
 public User() {  
 }  
  
 public User(String userId, String password) {  
 this.userId = userId;  
 this.password = password;  
 }  
  
 public String getUserId() {  
 return userId;  
 }  
  
 public void setUserId(String userId) {  
 this.userId = userId;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
  
}

课程信息类

package com.example.fouthapp.entity;  
  
import org.litepal.crud.LitePalSupport;  
  
public class Cources extends LitePalSupport {  
 private String Cno;  
 private String Cname;  
 private String Classes;  
 private String Tno;  
  
 public String getCno() {  
 return Cno;  
 }  
  
 public void setCno(String cno) {  
 Cno = cno;  
 }  
  
 public String getCname() {  
 return Cname;  
 }  
  
 public void setCname(String cname) {  
 Cname = cname;  
 }  
  
 public String getClasses() {  
 return Classes;  
 }  
  
 public void setClasses(String classes) {  
 Classes = classes;  
 }  
  
 public String getTno() {  
 return Tno;  
 }  
  
 public void setTno(String tno) {  
 Tno = tno;  
 }  
}

学生类：

package com.example.fouthapp.entity;  
  
import org.litepal.crud.LitePalSupport;  
  
public class Student extends LitePalSupport {  
 private String Sno;  
 private String Sname;  
 private int Sage;  
 private String Ssex;  
 private String Sdept;  
 private String Sclass;  
 private double Score;  
  
 private boolean isAttended;  
 public String getSno() {  
 return Sno;  
 }  
  
 public void setSno(String sno) {  
 Sno = sno;  
 }  
  
 public String getSname() {  
 return Sname;  
 }  
  
 public void setSname(String sname) {  
 Sname = sname;  
 }  
  
 public int getSage() {  
 return Sage;  
 }  
  
 public void setSage(int sage) {  
 Sage = sage;  
 }  
  
 public String getSsex() {  
 return Ssex;  
 }  
  
 public void setSsex(String ssex) {  
 Ssex = ssex;  
 }  
  
 public String getSdept() {  
 return Sdept;  
 }  
  
 public void setSdept(String sdept) {  
 Sdept = sdept;  
 }  
  
 public String getSclass() {  
 return Sclass;  
 }  
  
 public void setSclass(String sclass) {  
 Sclass = sclass;  
 }  
  
 public double getScore() {  
 return Score;  
 }  
  
 public void setScore(double score) {  
 Score = score;  
 }  
  
  
 public boolean isAttended() {  
 return isAttended;  
 }  
  
 public void setAttended(boolean attended) {  
 isAttended = attended;  
 }  
  
}

教师类：

package com.example.fouthapp.entity;  
  
import org.litepal.crud.LitePalSupport;  
  
public class Teacher extends LitePalSupport {  
 private String Tno;  
 private String Tname;  
 private String Tsex;  
 private int Tage;  
 private String Tdept;  
 private String Tclass;  
  
 public String getTno() {  
 return Tno;  
 }  
  
 public void setTno(String tno) {  
 Tno = tno;  
 }  
  
 public String getTname() {  
 return Tname;  
 }  
  
 public void setTname(String tname) {  
 Tname = tname;  
 }  
  
 public String getTsex() {  
 return Tsex;  
 }  
  
 public void setTsex(String tsex) {  
 Tsex = tsex;  
 }  
  
 public int getTage() {  
 return Tage;  
 }  
  
 public void setTage(int tage) {  
 Tage = tage;  
 }  
  
 public String getTdept() {  
 return Tdept;  
 }  
  
 public void setTdept(String tdept) {  
 Tdept = tdept;  
 }  
  
 public String getTclass() {  
 return Tclass;  
 }  
  
 public void setTclass(String Tclass) {  
 this.Tclass = Tclass;  
 }  
  
}

签到信息：

package com.example.fouthapp.entity;  
  
import org.litepal.crud.LitePalSupport;  
  
import java.util.Date;  
  
public class Record extends LitePalSupport {  
 private String Rno;  
 private String Sno;  
 private String record;  
 private Date Rtime;  
 private String Tno;  
 private String Note;  
  
 public String getRno() {  
 return Rno;  
 }  
  
 public void setRno(String rno) {  
 Rno = rno;  
 }  
  
 public String getSno() {  
 return Sno;  
 }  
  
 public void setSno(String sno) {  
 Sno = sno;  
 }  
  
 public String getRecord() {  
 return record;  
 }  
  
 public void setRecord(String record) {  
 this.record = record;  
 }  
  
 public Date getRtime() {  
 return Rtime;  
 }  
  
 public void setRtime(Date rtime) {  
 Rtime = rtime;  
 }  
  
 public String getTno() {  
 return Tno;  
 }  
  
 public void setTno(String tno) {  
 Tno = tno;  
 }  
  
 public String getNote() {  
 return Note;  
 }  
  
 public void setNote(String note) {  
 Note = note;  
 }  
}

班级类：

package com.example.fouthapp.entity;  
  
import org.litepal.crud.LitePalSupport;  
  
public class Classes extends LitePalSupport {  
 private String CLno;  
 private String CLname;  
 private String Tno;  
  
 public String getCLno() {  
 return CLno;  
 }  
  
 public void setCLno(String CLno) {  
 this.CLno = CLno;  
 }  
  
 public String getCLname() {  
 return CLname;  
 }  
  
 public void setCLname(String CLname) {  
 this.CLname = CLname;  
 }  
  
 public String getTno() {  
 return Tno;  
 }  
  
 public void setTno(String tno) {  
 Tno = tno;  
 }  
}

3．6存储分配

本程序由Android进行编码，直接内存分配由Android运行进行分配

本组件所依赖的变量，结构要求全部在组件元素内申明。

3．7 注释

在代码的适当处有代码注释：

（1）模块首部的注释，要大体上说明模块要实现的功能

（2）各个语句后面的注释要说明语句实现的功能

（3）注释要规范，命名要标准

1. 8 限制条件

所有的用户的权限都经过严格的设置，不能进行权限外的操作。

1. 9 测试计划

|  |  |  |  |
| --- | --- | --- | --- |
| 测试名称 | 测试进度安排 | 测试目的 | 测试内容 |
| 基本数据输入 | 系统完成后 | 系统功能是否达到要求 | 输入简单的一致的数据 |
| 非法数据输入 | 基本数据测试后 | 系统对非法数据的反应 | 输入一些特殊字符和字符串、边界值的数据 |
| 空字符串 | 非法测试以后 | 系统对空字符串的反应 | 在信息上输入空字符 |

# 4 其它附加功能及帮助系统

## 4．1 程序描述

本程序设计是实现学生端签到教师端可以收到签到信息的这么一个功能教师端可导出学生本学期缺勤次数。系统的主要目的是为教师对学生出勤率的考察提供的平台。

**手势打卡：**

GestureDetector.OnGestureListener

这个是常用的接口，作用手势识别的回调。它所有的方法以及意义。

OnDown(MotionEvent e)：用户按下屏幕就会触发；

onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) ，滑屏，用户按下触摸屏、快速移动后松开，由1个MotionEvent ACTION\_DOWN, 多个ACTION\_MOVE, 1个ACTION\_UP触发 参数解释： e1：第1个ACTION\_DOWN MotionEvent e2：最后一个ACTION\_MOVE MotionEvent velocityX：X轴上的移动速度，像素/秒 velocityY：Y轴上的移动速度，像素/秒。

onLongPress(MotionEvent e) ，长按触摸屏，超过一定时长，就会触发这个事件 触发顺序： onDown->onShowPress->onLongPress。

onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY) ，在屏幕上拖动事件。无论是用手拖动view，或者是以抛的动作滚动，都会多次触发,这个方法 在ACTION\_MOVE动作发生时就会触发 滑屏：手指触动屏幕后，稍微滑动后立即松开 onDown—–》onScroll—-》onScroll—-》onScroll—-》………—–>onFling 拖动 onDown——》onShowPress —》onScroll—-》onScroll——》onFiling 可见，无论是滑屏，还是拖动，影响的只是中间OnScroll触发的数量多少而已，最终都会触发onFling事件！

onShowPress(MotionEvent e) ，如果是按下的时间超过瞬间，而且在按下的时候没有松开或者是拖动的，那么onShowPress就会执行。这个时间是底层封装，设定的。开发者在应用层只需应用，不必知晓时间设定以及原理。

onSingleTapUp(MotionEvent e) ，顾名思义，一次单独的轻击抬起操作,也就是轻击一下屏幕，立刻抬起来，才会有这个触发，当然,如果除了Down以外还有其它操作,那就不再算是Single操作了,所以也就不会触发这个事件 触发顺序： 点击一下非常快的（不滑动）Touchup： onDown->onSingleTapUp->onSingleTapConfirmed 点击一下稍微慢点的（不滑动）Touchup： onDown->onShowPress->onSingleTapUp->onSingleTapConfirmed。

**定位打卡：**

Android平台下支持提供位置服务的API，在开发过程中主要用到LocationManager和LocationProviders对象：

(1).LocationManager可以用来获取当前的位置，追踪设备的移动路线，或设定敏感区域，在进入或离开敏感区域时设备会发出特定警报 。

(2).LocationProviders则是提供定位功能的组件集合，集合中的每种组件以不同的技术提供设备的当前位置，区别在于定位的精度、速度和成本等方面 。

接下来将讲述一个如何实现获取位置的经纬度，并且如果位置改变，如何通过位置的改变，经纬度也发生变化的例子，这里以LocationManager对象为例：

(1).首先，第一步，获取LocationManager对象，可以通过调用android.app.Activity.getSystemService()函数获取，代码如下：

String serviceString = Context.LOCATION\_SERVICE;// 获取的是位置服务

LocationManager locationManager = (LocationManager) getSystemService(serviceString);// 调用getSystemService()方法来获取LocationManager对象

其中的LOCATION\_SERVICE是Android支持的系统级服务，控制位置等设备的更新。

(2).在获取到LocationManager对象后，还需要指定LocationManager的定位方法 ，然后才能够调用LocationManager.getLastKnowLocation()方法获取当前位置 ，目前LocationManager主要有两种定位方法

GPS定位：可以提供更加精确的位置信息，但定位速度和质量受到卫星数量和环境情况的影响，需要android.permissions.ACCESS\_FINE\_LOCATION用户权限。

网络定位：提供的位置信息精度差，但速度较GPS定位要迅速 ，利用基站或WiFi访问的提供近似的位置信息，需要具有如下权限：android.permission.ACCESS\_COARSE\_LOCATION 或 android.permission.ACCESS\_FINE\_LOCATION。

注：(使用GPS定位和网络定位的LocationManager类的静态常量不一样，GPS定位的LocationManager类的静态常量为：GPS\_PROVIDER，网络定位的LocationManager类的静态常量为：NETWORK\_PROVIDER，这两个静态常量在获取当前位置时要用到。)

下面以使用GPS定位为例，获取位置信息代码如下：

String provider = LocationManager.GPS\_PROVIDER;// 指定LocationManager的定位方法

Location location = locationManager.getLastKnownLocation(provider);// 调用getLastKnownLocation()方法获取当前的位置信息

(3).通过调用Location中的getLatitude()和getLonggitude()方法可以分别获取位置信息中的纬度和经度，代码如下：

double lat = location.getLatitude();//获取纬度

double lng = location.getLongitude();//获取经度

(4).在很多提供定位服务的应用程序中，不仅需要获取当前的位置信息，还需要监视位置的变化，在位置改变时调用特定的处理方法 ，其中LocationManager提供了一种便捷、高效的位置监视方法requestLocationUpdates()，可以根据位置的距离变化和时间间隔设定，产生位置改变事件的条件，这样可以避免因微小的距离变化而产生大量的位置改变事件 ，LocationManager中设定监听位置变化的代码如下：

locationManager.requestLocationUpdates(provider, 2000, 10,locationListener);// 产生位置改变事件的条件设定为距离改变10米，时间间隔为2秒，设定监听位置变化

接下来介绍上面这行代码的各个参数，第一个参数是我们之前指定LocationManager的定位方法，GPS定位或网络定位，第二个参数指的是产生位置改变事件的时间间隔，单位为微秒，第三个参数指的是距离条件，单位为米，第四个参数是回调函数，用于处理位置改变事件，即设置LocationListener监听器。总的来说，那一行代码将产生位置改变事件的条件设定为距离改变10米，时间间隔为2秒。

**指纹打卡：**

指纹识别的对称加密实现

使用指纹识别的对称加密功能的主要流程如下：

使用 KeyGenerator 创建一个对称密钥，存放在 KeyStore 里。

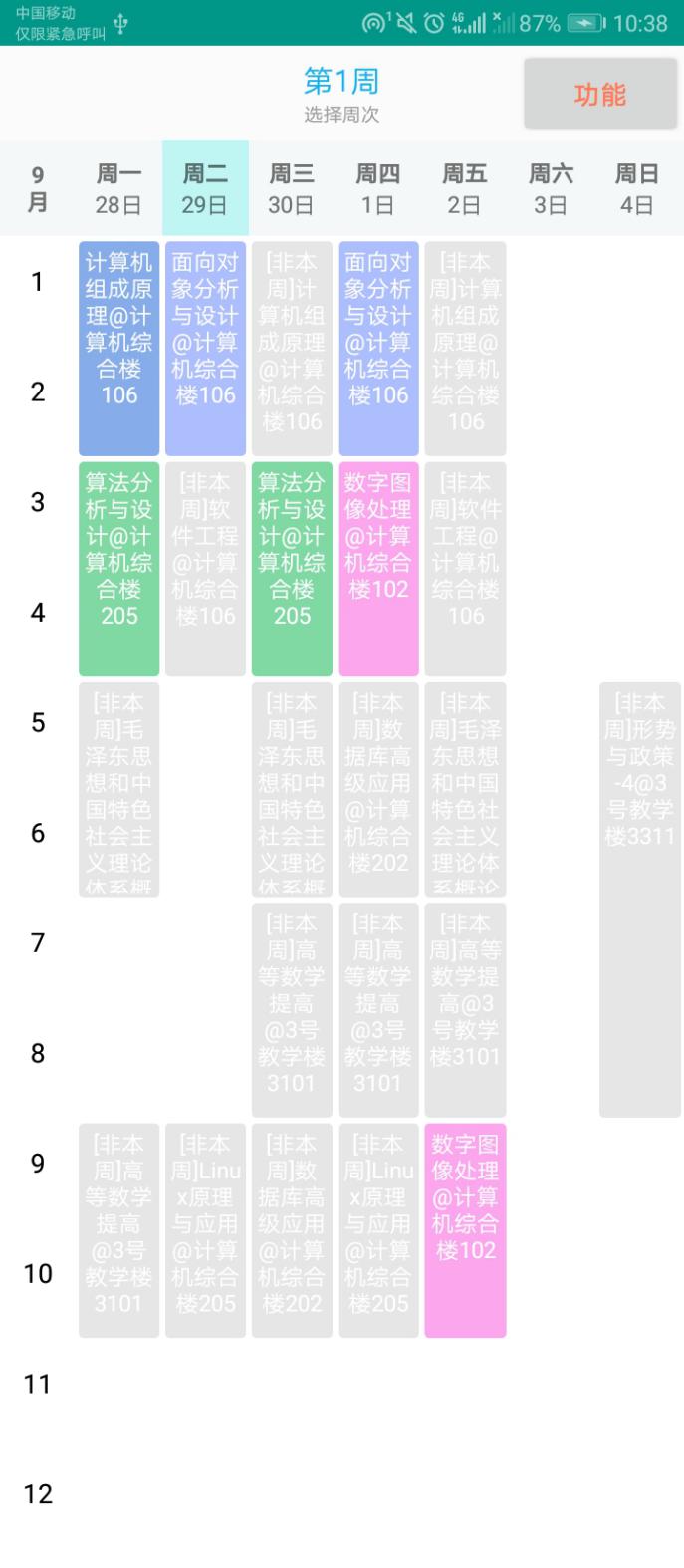
设置 KeyGenParameterSpec.Builder.setUserAuthenticationRequired() 为true，

使用创建好的对称密钥初始化一个Cipher对象，并用该对象调用 FingerprintManager.authenticate() 方法启动指纹传感器并开始监听。

重写 FingerprintManager.AuthenticationCallback 的几个回调方法，以处理指纹识别成功（onAuthenticationSucceeded()）、失败（onAuthenticationFailed() 和 onAuthenticationError()）等情况。

## 4．2 功能

1.查看课程表



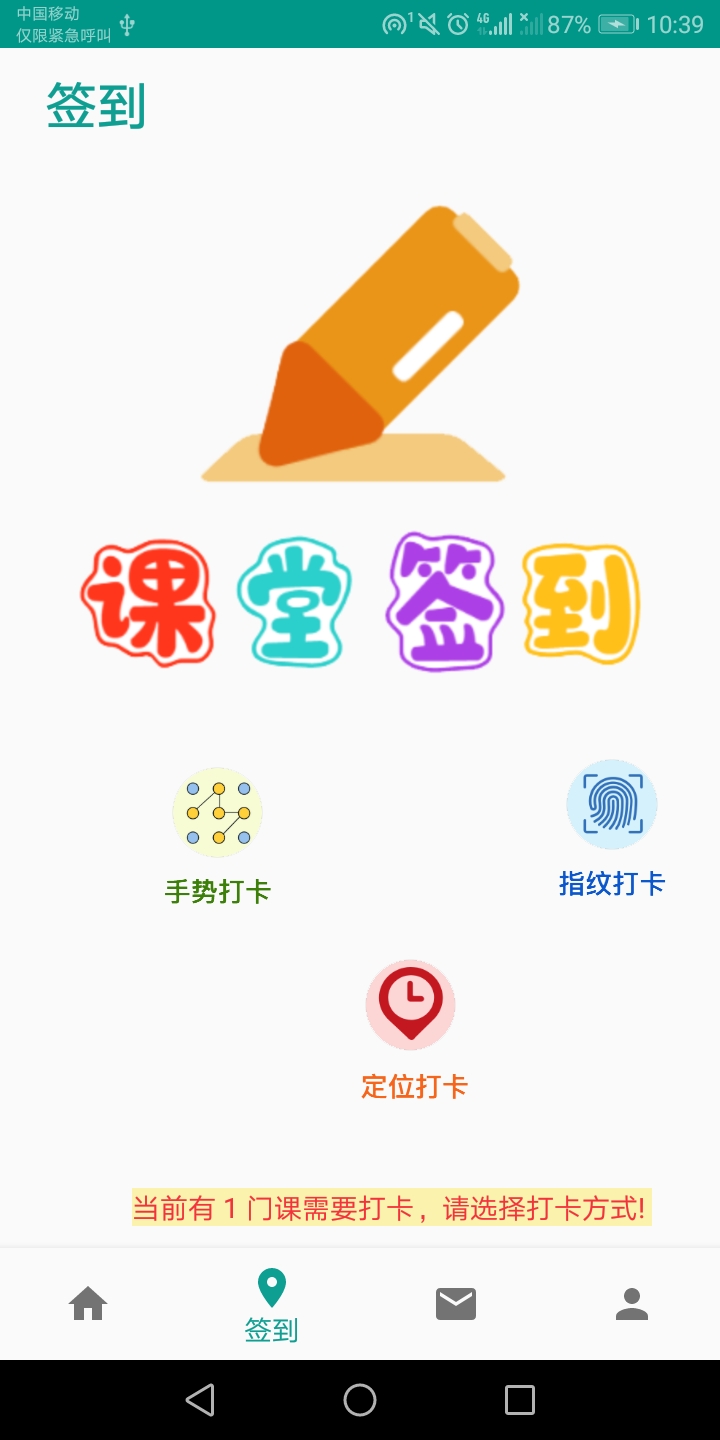
2.请假页面



3.学生打卡信息表



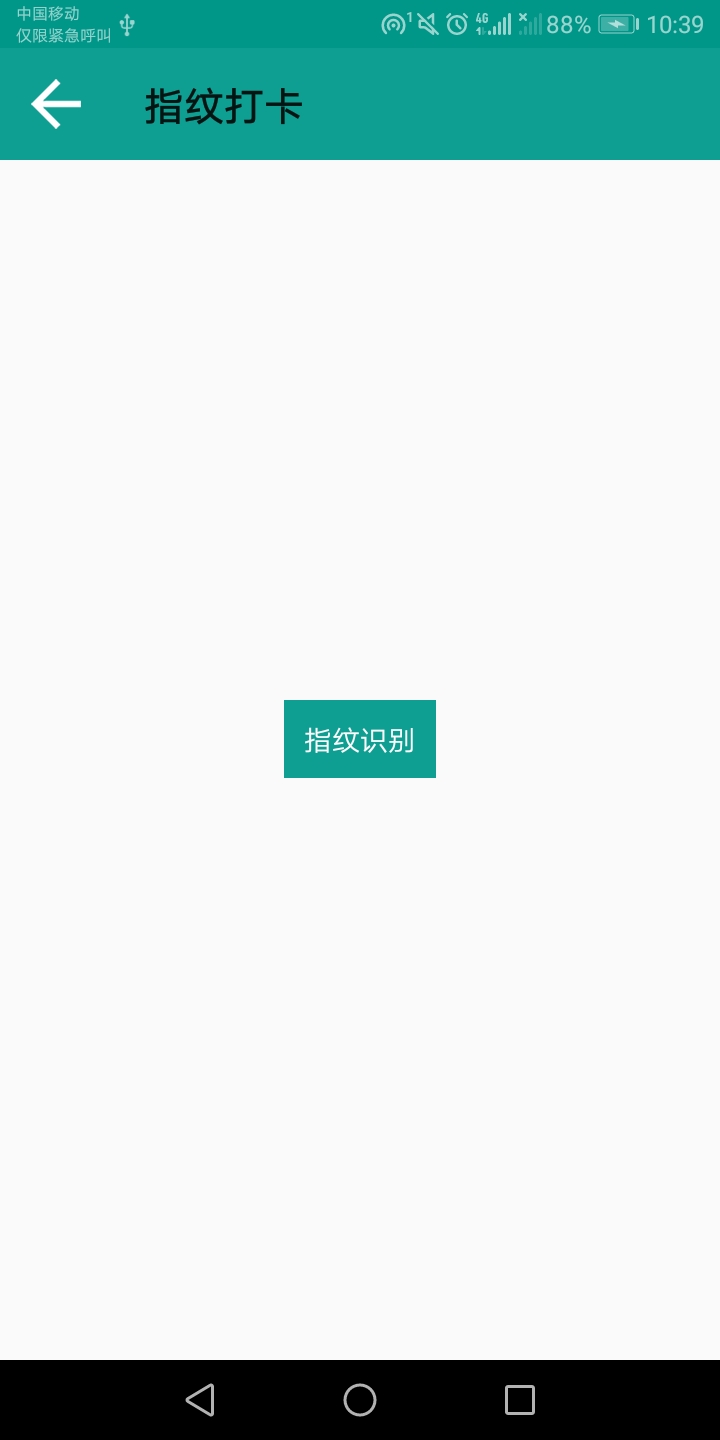
3.签到分类界面



4.手势打卡



5.指纹打卡



6.定位打卡



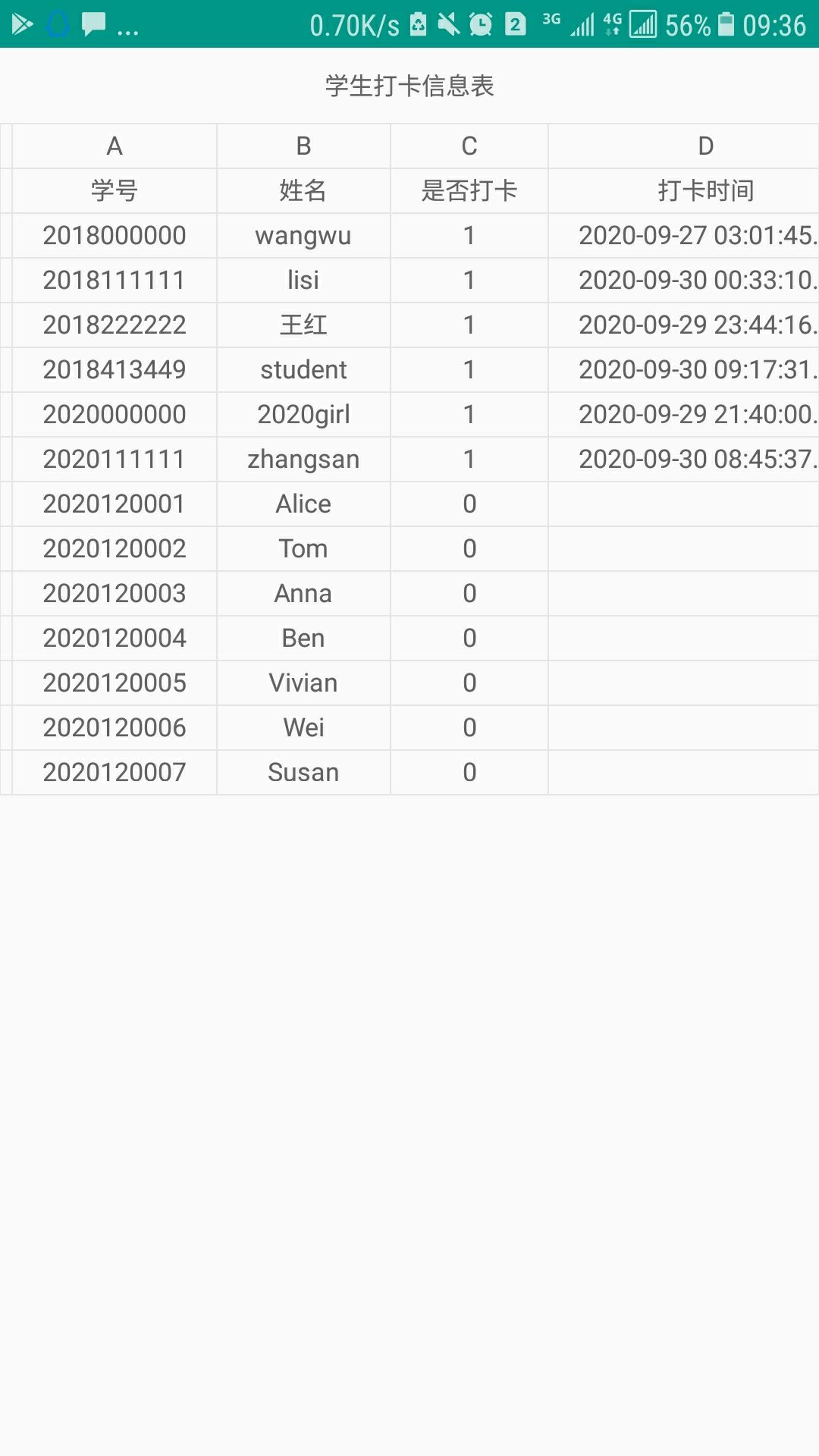
7.消息通知页面



8.教师设置手势打卡密码



1. 教师查看学生打卡信息表



1. 修改密码



## 4．3 性能

### 4.3.1 时间特性

（1）响应时间：启动系统后，系统就能运行。

（2）更新处理时间：由系统运行状态来决定。

（3）数据的转换和传送时间：尽可能尽快地完成数据转换和传送。

### 4.3.2 安全性

（1）系统应设置访问用户的标识以鉴别是否是合法用户，并要求合法用户设置其密码，保证用户身份不被盗用。

（2）系统应对不同的数据设置不同的访问级别，限制访问用户可查询和处理数据的类别 和内容。

（3）系统应对不同用户设置不同的权限，区分不同的用户。

### 4.3.3 完整性

（1）各种信息记录的完整性，信息记录内容不能为空。

（2）各种数据间相互的联系的正确性。

（3）相同的数据在不同记录中的一致性。

### 4.3.4 可维护性

各个功能高度模块化，达到高内聚低耦合的目标，实现清晰的模块接口，明确每个模块的功能，方便以后的系统维护，如果一个功能模块出现问题，不会致使整个系统瘫痪，当系统发生异常时需要日志记录，以便于能够及时地进行维护。

## 4．4 输入输出项

数据从键盘，鼠标按提示输入。依服务器响应从界面输出响应信息。

## 4．5 限制条件

在界面上控制输入输出，服务器端如果出错也会返回出错信息。

# 5 设计特点

## 5．1通信便捷

本系统客户端与服务器端采用JSON格式作为信息载体，在开发中将信息作为对象，信息传递很便捷。

## 5．2 开发速度快

由于使用通信协议约定，所以只要客户端与服务器端约定好通信协议API文档，即可以进行通信。而且直接对对象操作，不存在类型转换的问题。

# 6 项目分工

李贻慧：技术担当，实现定位签到，指纹签到，打卡签到，请假等主要功能，完成数据库连接，与数据库的数据交互，进行教师端的技术实现，完善项目，攻克各种技术难题。

邹继敏：进行UI界面设计，实现底部导航等功能，协调项目进程，进行项

目测试，编写文档。

荆心科：设计数据库，部署数据库，协作连接数据库，完成登陆注册功能，编写文档PPT。

张艳丽：统筹安排项目，查找项目需要用的资料，编写文档

王瑶：编写文档