DataAnalysis.java

package com.czc.android.chargeanalysis.dataprocessing;

import android.Manifest;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.net.Uri;

import android.os.Bundle;

import android.os.Environment;

import android.provider.Settings;

import android.support.v4.app.ActivityCompat;

import android.support.v4.app.Fragment;

import android.support.v4.app.FragmentActivity;

import android.support.v4.app.FragmentManager;

import android.support.v4.app.FragmentTransaction;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;

import com.czc.android.chargeanalysis.R;

import com.czc.android.chargeanalysis.chart.EfficiencyLineChart;

import com.czc.android.chargeanalysis.chart.PowerDistributePieChart;

import com.czc.android.chargeanalysis.chart.TimeDistributePieChart;

import com.czc.android.chargeanalysis.chart.WeeklyTimeBarChart;

import com.czc.android.chargeanalysis.record.MainActivity;

import com.czc.android.chargeanalysis.record.StartService;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.ArrayList;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import jxl.write.WriteException;

/\*\*

\* the main activity that controls the fragmentsForChart and gets permissions

\*/

public class DataAnalysis extends FragmentActivity implements View.OnClickListener {

private static final String TAG = "MAINACTIVITY";

private static final String READFILEPATH = "/chargestate.txt";

//set the request code for read and write

private static final int REQUEST\_EXTERNAL\_STORAGE = 1;

//set the permissions's name

private static String[] PERMISSIONS\_STORAGE = {

Manifest.permission.READ\_EXTERNAL\_STORAGE,

Manifest.permission.WRITE\_EXTERNAL\_STORAGE,

Manifest.permission.READ\_PHONE\_STATE};

private ArrayList<Integer> weekTimesData = new ArrayList<>();

private ArrayList<Double> efficiency = new ArrayList<>();

private double[] timeDistributeArray;

private double[] powerDistributeArray;

private int presentFragmentIndex;

private String totalRecord;

private String[] singleRecord;

private ExcelOperation excel;

private boolean findFile = false;

private Fragment[] fragmentsForChart;

private int[] fragmentsBtnIds;

private Button[] fragmentsBtns;

private Button saveExcelBtn;

private final static int REQUEST\_IGNORE\_BATTERY\_CODE = 1001;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

init();

if (findFile == true){

getFileContent();

}

setMainInterface();

Intent intent = new Intent(DataAnalysis.this,StartService.class);

startService(intent);

Log.e("jobservice","dataIntent");

}

private void init() {

getPermission();

/\*

\* require for battery optimization

\* \*/

try {

Intent intent = new Intent();

String packageName = getPackageName();

intent.setAction(Settings.ACTION\_REQUEST\_IGNORE\_BATTERY\_OPTIMIZATIONS);

intent.setData(Uri.parse("package:" + packageName));

startActivityForResult(intent, REQUEST\_IGNORE\_BATTERY\_CODE);

}catch (Exception e){

Toast.makeText(this,e.getMessage(),Toast.LENGTH\_LONG);

}

fragmentsForChart = new Fragment[]{new WeeklyTimeBarChart(),null,null,null};

fragmentsBtnIds = new int[]{R.id.btn\_show\_fragment1,R.id.btn\_show\_fragment2,R.id.btn\_show\_fragment3,R.id.btn\_show\_fragment4};

fragmentsBtns = new Button[fragmentsBtnIds.length];

totalRecord = readTextFromFile(READFILEPATH);

verifyFile();

init fragment buttons

for(int i=0;i<fragmentsBtnIds.length;i++){

fragmentsBtns[i] = findViewById(fragmentsBtnIds[i]);

fragmentsBtns[i].setOnClickListener(this);

}

// init save buttons

saveExcelBtn = this.findViewById(R.id.button\_save);

saveExcelBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View arg0) {

if (findFile == true){

try {

excel.creat(DataAnalysis.this);

} catch (WriteException e) {

e.printStackTrace();

}

if (excel.isdataFromat() == true) {

Toast.makeText(DataAnalysis.this, getString(R.string.save\_excel\_prompt), Toast.LENGTH\_LONG).show();

}

}

}

});

}

private void verifyFile(){

PackageManager pm = getPackageManager();

boolean permission = (PackageManager.PERMISSION\_GRANTED ==

pm.checkPermission("android.permission.WRITE\_EXTERNAL\_STORAGE", "com.czc.android.chargeanalysis"));

Log.e("permission", String.valueOf(permission));

if (permission == false){

Toast.makeText(DataAnalysis.this, "无文件权限，请获取权限后重启应用!", Toast.LENGTH\_LONG).show();

Toast.makeText(DataAnalysis.this, getString(R.string.no\_permission\_prompt), Toast.LENGTH\_LONG).show();

}else {

if (totalRecord.equals("")){

Toast.makeText(DataAnalysis.this, getString(R.string.no\_file\_prompt), Toast.LENGTH\_LONG).show();

}else{

findFile = true;

}

}

}

private void getFileContent() {

singleRecord = splitRecord(totalRecord);

excel = new ExcelOperation();

excel.arrangeData(singleRecord, this);

weekTimesData = excel.getChargeTimeList();

Log.e("week", String.valueOf(weekTimesData.size()));

efficiency = excel.getBatteryLife();

timeDistributeArray = excel.getChargeTimeNum();

powerDistributeArray = excel.getChargePowerNum();

}

private void setMainInterface() {

// add first framgement

fragmentsForChart[0] = new WeeklyTimeBarChart();

addFragment(fragmentsForChart[0], "fragment0");

fragmentsForChart[0].setArguments(deliverData());

presentFragmentIndex = 0;

}

@Override

public void onClick(View view) {

//get index of fragmentsBtnIds

int index = 0;

for(int i = 0; i<fragmentsBtnIds.length;i++){

if (view.getId() == fragmentsBtnIds[i]){

index = i;

break;

}

}

FragmentManager manager = getSupportFragmentManager();

FragmentTransaction transaction = manager.beginTransaction();

if (fragmentsForChart[index] == null) {

presentFragmentIndex = index;

switch (index) {

case 0:

fragmentsForChart[index] = new WeeklyTimeBarChart();

break;

case 1:

fragmentsForChart[index] = new EfficiencyLineChart();

break;

case 2:

fragmentsForChart[index] = new TimeDistributePieChart();

break;

case 3:

fragmentsForChart[index] = new PowerDistributePieChart();

break;

}

fragmentsForChart[index].setArguments(deliverData());

Log.d("zcc","new fragement"+ index);

transaction.add(R.id.fragment\_container, fragmentsForChart[index], "fragment" + index);

}

for(int i = 0; i<fragmentsForChart.length;i++){

if (fragmentsForChart[i]!= null){

if (i != index){

transaction.hide(fragmentsForChart[i]);

}else{

transaction.show(fragmentsForChart[i]);

presentFragmentIndex = i;

}

}

}

transaction.commit();

}

private void addFragment(Fragment fragment, String tag) {

FragmentManager manager = getSupportFragmentManager();

FragmentTransaction transaction = manager.beginTransaction();

transaction.add(R.id.fragment\_container, fragment, tag);

transaction.commit();

}

//deliver the data from the activity to the fragmentsForChart.

public Bundle deliverData() {

//transfer format for bundle

double[] efficiencyBundle = new double[efficiency.size()];

for (int i = 0; i < efficiency.size(); i++){

efficiencyBundle[i] = efficiency.get(i);

}

Bundle bundle = new Bundle();

Log.e("weektime", String.valueOf(weekTimesData.size()));

bundle.putIntegerArrayList("weekTimesData", weekTimesData);

bundle.putDoubleArray("timeDistribute", timeDistributeArray);

bundle.putDoubleArray("powerDistribute", powerDistributeArray);

bundle.putDoubleArray("efficiency", efficiencyBundle);

return bundle;

}

public void getPermission() {

int permission = ActivityCompat.checkSelfPermission(this, Manifest.permission.WRITE\_EXTERNAL\_STORAGE);

if (permission != PackageManager.PERMISSION\_GRANTED) {

// We don't have permission so prompt the user

ActivityCompat.requestPermissions(

this,

PERMISSIONS\_STORAGE,

REQUEST\_EXTERNAL\_STORAGE

);

}

}

private String readTextFromFile(String path) {

String result = "";

File file = new File(Environment.getExternalStorageDirectory().getAbsolutePath() + path);

if (!file.exists()) {

Log.e(TAG, "notExists");

try {

file.createNewFile();

} catch (IOException e) {

e.printStackTrace();

}

}

FileInputStream fis = null;

try {

fis = new FileInputStream(file);

int length = fis.available();

byte[] buffer = new byte[length];

fis.read(buffer);

fis.close();

result = new String(buffer, "UTF-8");

Log.e(TAG, "readSuccess");

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return result;

}

private String[] splitRecord(String rawRecord) {

String totalStandardData = "";

String regularExpression = "[1-9][0-9][0-9][0-9]-[0-1][0-9]-[0-3][0-9] [0-2][0-9]:[0-5][0-9]:[0-5][0-9] "

+ "\\d{1,3}%" + " [^x00-xff]{4,4}\n";

Pattern p = Pattern.compile(regularExpression);

Matcher m = p.matcher(rawRecord);

while (m.find()){

totalStandardData = totalStandardData + m.group() + " ";

}

String [] processedRecord = null;

processedRecord = totalStandardData.split(" ");

for (int i = 0; i < processedRecord.length; i++){

Log.e(TAG, processedRecord[i]);

}

return processedRecord;

}

@Override

public void onPointerCaptureChanged(boolean hasCapture) {

}

}

StartService.java

package com.czc.android.chargeanalysis.record;

import android.Manifest;

import android.app.job.JobInfo;

import android.app.job.JobParameters;

import android.app.job.JobScheduler;

import android.app.job.JobService;

import android.content.BroadcastReceiver;

import android.content.ComponentName;

import android.content.Context;

import android.content.Intent;

import android.content.IntentFilter;

import android.content.pm.PackageManager;

import android.os.BatteryManager;

import android.os.Binder;

import android.os.Environment;

import android.support.v4.app.ActivityCompat;

import android.telephony.TelephonyManager;

import android.util.Log;

import com.czc.android.chargeanalysis.R;

import java.io.BufferedWriter;

import java.io.ByteArrayOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.text.SimpleDateFormat;

/\*\*

\* Created by czc on 12/26/17.

\*/

public class StartService extends JobService {

//public static String PHONENO;

private final static String SAVEPATH = "/chargestate\_" + android.os.Build.MODEL + ".txt";

public static boolean onepxActivity = false;

private int BatteryN; //目前电量

private int recordBatteryN;

private SimpleDateFormat fillTimeFormat;

private String lastStatus = "";

private String recordTime;

private boolean ifSaveCharge = true;

private boolean ifSaveDischarge = true;

private boolean saveCharge = true;

private boolean firstBoot = false;

public class LocalBinder extends Binder {

StartService getService() {

return StartService.this;

}

}

@Override

public boolean onStartJob(JobParameters params) {

registerIntentReceiver();

startJobSheduler();

Log.e("jobservice","onStart");

return false;

}

@Override

public boolean onStopJob(JobParameters params) {

return false;

}

private void registerIntentReceiver() {

//此处添加启动服务要执行的操作代码

new Thread(new Runnable() {

@Override

public void run() {

for (int i = 0; i < 99; i++) {

try {

// Log.e("bootstartservice", String.valueOf(i));

Thread.sleep(6000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}).start();

registerReceiver(mBatInfoReceiver, new IntentFilter(Intent.ACTION\_BATTERY\_CHANGED));

registerReceiver(mShutReceiver, new IntentFilter(Intent.ACTION\_SHUTDOWN));

registerReceiver(mScreenStatusReceiver, new IntentFilter(Intent.ACTION\_SCREEN\_ON));

registerReceiver(mScreenStatusReceiver, new IntentFilter(Intent.ACTION\_SCREEN\_OFF));

try {

// saveStream("/chargestate1.bin");

} catch (Exception e) {

e.printStackTrace();

}

fillTimeFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

// Log.e("currentTime", String.valueOf(System.currentTimeMillis()));

}

public void onStart(Intent intent, int startId) {

super.onStart(intent, startId);

}

@Override

public void onCreate() {

super.onCreate();

Log.e("jobservice","onCreate");

startJobSheduler();

}

public void startJobSheduler() {

// new Thread(new Runnable() {

// @Override

// public void run() {

// for (int i = 0; i < 99; i++) {

// try {

// Log.e("bootstartservice", String.valueOf(i));

// Thread.sleep(6000);

//

// } catch (InterruptedException e) {

// e.printStackTrace();

// }

// }

// }

//

// }).start();

try {

JobInfo.Builder builder = new JobInfo.Builder(1, new ComponentName(getPackageName(), StartService.class.getName()));

builder.setRequiresCharging(true); // 充电状态

JobScheduler jobScheduler = (JobScheduler) this.getSystemService(Context.JOB\_SCHEDULER\_SERVICE);

jobScheduler.schedule(builder.build());

Log.e("jobservice","startJobSheduler");

} catch (Exception ex) {

ex.printStackTrace();

}

}

public BroadcastReceiver mBatInfoReceiver = new BroadcastReceiver() {

public void onReceive(Context context, Intent intent) {

String action = intent.getAction();

//如果捕捉到的action是ACTION\_BATTERY\_CHANGED， 就运行onBatteryInfoReceiver()

if (Intent.ACTION\_BATTERY\_CHANGED.equals(action)) {

PackageManager pm = getPackageManager();

boolean permission = (PackageManager.PERMISSION\_GRANTED ==

pm.checkPermission("android.permission.WRITE\_EXTERNAL\_STORAGE", "com.czc.android.chargerecord"));

if (permission == true){

// if (MainActivity.isHideIcon == false){

// Intent mainActivity = new Intent(context, MainActivity.class);

// mainActivity.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

// startActivity(mainActivity);

// Log.e("hide","once");

// }

if (onepxActivity == false) {

if (MainActivity.ifHideIcon == true){

// ActivityManager a = new ActivityManager()

// 说明系统中不存在这个activity

Intent onrPXActivity = new Intent(context, OnePxActivity.class);

onrPXActivity.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(onrPXActivity);

Log.e("permision", "true");

}

}

}

BatteryN = intent.getIntExtra("level", 0); //目前电量（0~100）

switch (intent.getIntExtra("status", BatteryManager.BATTERY\_STATUS\_UNKNOWN)) {

case BatteryManager.BATTERY\_STATUS\_CHARGING:

if (firstBoot == true) {

saveToPath(fillTimeFormat.format(System.currentTimeMillis()) + " " + BatteryN + "% " + getString(R.string.restart\_charge) + "\n");

firstBoot = false;

}else {

if (!lastStatus.equals("charging")){

if (ifSaveCharge == true){

ifSaveCharge = false;

recordTime = fillTimeFormat.format(System.currentTimeMillis());

recordBatteryN = BatteryN;

lastStatus = "charging";

new Thread(new Runnable() {

public void run(){

try {

Thread.sleep(5000);

} catch (InterruptedException e) {

e.printStackTrace();

}

if (lastStatus.equals("charging")){

saveToPath(recordTime + " " + recordBatteryN + "% " + getString(R.string.start\_charge) + "\n");

saveCharge = true;

}

ifSaveCharge = true;

}

}).start();

}

}

}

lastStatus = "charging";

break;

case BatteryManager.BATTERY\_STATUS\_DISCHARGING :

notCharge();

break;

case BatteryManager.BATTERY\_STATUS\_NOT\_CHARGING:

notCharge();

break;

case BatteryManager.BATTERY\_STATUS\_FULL:

notCharge();

break;

case BatteryManager.BATTERY\_STATUS\_UNKNOWN:

break;

}

}

}

};

public BroadcastReceiver mScreenStatusReceiver = new BroadcastReceiver() {

public void onReceive(Context context, Intent intent) {

String action = intent.getAction();

if(action.equals(Intent.ACTION\_SCREEN\_OFF)){

// 当屏幕关闭时，启动一个像素的Activity

if (MainActivity.ifHideIcon == true){

// 说明系统中不存在这个activity

Intent onrPXActivity = new Intent(context, OnePxActivity.class);

onrPXActivity.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(onrPXActivity);

Log.e("permision", "true");

}

} else if (action.equals(Intent.ACTION\_SCREEN\_ON)){

// 发这个广播就是为了最小化OnePxActivity

Intent broadcast = new Intent("FinishActivity");

// broadcast.setFlags(32);Intent.FLAG\_INCLUDE\_STOPPED\_PACKAGES

context.sendBroadcast(broadcast);//发送对应的广播

}

}

};

public BroadcastReceiver mShutReceiver = new BroadcastReceiver() {

public void onReceive(Context context, Intent intent) {

String action = intent.getAction();

if (Intent.ACTION\_SHUTDOWN.equals(action)) {

saveToPath(fillTimeFormat.format(System.currentTimeMillis()) + " " + BatteryN + "% " + getString(R.string.shut\_down)+ "\n");

Log.e("shutdown", "shutdown");

lastStatus = "shutdown";

}

}

};

private void saveToPath(String content) {

SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd:hh:mm:ss");

Log.e("SaveToPath", sdf.format(System.currentTimeMillis()));

File file = new File(Environment.getExternalStorageDirectory().getAbsolutePath() + SAVEPATH);

Log.e("create", String.valueOf(file));

if (!file.exists()) {

try {

file.createNewFile();

TelephonyManager telephonyManager = (TelephonyManager) this.getSystemService(Context.TELEPHONY\_SERVICE);

if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ\_PHONE\_STATE) != PackageManager.PERMISSION\_GRANTED) {

// TODO: Consider calling

// ActivityCompat#requestPermissions

// here to request the missing permissions, and then overriding

// public void onRequestPermissionsResult(int requestCode, String[] permissions,

// int[] grantResults)

// to handle the case where the user grants the permission. See the documentation

// for ActivityCompat#requestPermissions for more details.

return;

}

String imei = telephonyManager.getDeviceId();

content = "IMEI: " + imei + "\n" + content;

Log.e("IMEI", imei);

} catch (IOException e) {

e.printStackTrace();

}

}

FileOutputStream fos = null;

try {

fos = new FileOutputStream(file,true);

OutputStreamWriter write = new OutputStreamWriter(fos,"UTF-8");

BufferedWriter writer=new BufferedWriter(write);

writer.write(content);

writer.flush();

writer.close();

} catch (FileNotFoundException e) {

} catch (IOException e) {

e.printStackTrace();

}

}

private String readTextFromFile(String path) {

String result = "";

File file = new File(Environment.getExternalStorageDirectory().getAbsolutePath() + path);

if (!file.exists()) {

try {

file.createNewFile();

} catch (IOException e) {

e.printStackTrace();

}

}

FileInputStream fis = null;

try {

fis = new FileInputStream(file);

int length = fis.available();

byte[] buffer = new byte[length];

Log.e("byte", String.valueOf(buffer.length));

fis.read(buffer);

fis.close();

result = new String(buffer, "UTF-8");

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

//it may be Coding problem

e.printStackTrace();

}

return result;

}

private void notCharge() {

if (firstBoot == true){

saveToPath(fillTimeFormat.format(System.currentTimeMillis()) + " " + BatteryN + "% " + getString(R.string.restart\_discharge) + "\n");

firstBoot = false;

} else {

if (!lastStatus.equals("discharging")){

if (ifSaveDischarge == true){

ifSaveDischarge = false;

recordTime = fillTimeFormat.format(System.currentTimeMillis());

recordBatteryN = BatteryN;

Log.e("记录","startDischarge");

if (saveCharge == true){

saveToPath(recordTime + " " + recordBatteryN + "% " + getString(R.string.end\_charge) + "\n");

saveCharge = false;

}

ifSaveDischarge = true;

}

}

}

lastStatus = "discharging";

}

public int onStartCommand(Intent intent, int flags, int startId) {

if (intent != null){

firstBoot = intent.getBooleanExtra("boot", false);

}

if (firstBoot == true){

unregisterReceiver(mBatInfoReceiver);

registerReceiver(mBatInfoReceiver, new IntentFilter(Intent.ACTION\_BATTERY\_CHANGED));

}

return START\_STICKY;

}

public void onDestroy() {

Intent sevice = new Intent(this, StartService.class);

this.startService(sevice);

unregisterReceiver(mBatInfoReceiver);

unregisterReceiver(mShutReceiver);

unregisterReceiver(mScreenStatusReceiver);

super.onDestroy();

}

public void saveStream(String path) throws Exception {

char[] transfer = "rfesza".toCharArray();

byte[] toBy = new byte[transfer.length];

for(int i=0;i < toBy.length; i++){

toBy[i] = (byte)transfer[i];

Log.e("Byte", String.valueOf(toBy[i]));

}

FileOutputStream fos = new FileOutputStream(Environment.getExternalStorageDirectory().getAbsolutePath() + path);

ByteArrayOutputStream outStream = new ByteArrayOutputStream();

outStream.write(toBy);

Log.e("outStream", outStream.toString());

outStream.writeTo(fos);

outStream.flush();

outStream.close();

}

}

ExcelOperation.java

import android.content.Context;

import android.os.Environment;

import android.support.annotation.Nullable;

import android.util.Log;

import android.widget.Toast;

import com.czc.android.chargeanalysis.R;

import java.io.File;

import java.io.IOException;

import java.text.DecimalFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import jxl.Workbook;

import jxl.format.Alignment;

import jxl.format.Colour;

import jxl.format.UnderlineStyle;

import jxl.format.VerticalAlignment;

import jxl.write.Label;

import jxl.write.Number;

import jxl.write.WritableCellFormat;

import jxl.write.WritableFont;

import jxl.write.WritableSheet;

import jxl.write.WritableWorkbook;

import jxl.write.WriteException;

/\*\*

\* Created by czc on 12/7/17.

\*/

public class ExcelOperation {

private static final String TAG = "ExcelOperation";

private static final int MSECINWEEK = 604800000;

private static final int RESTARTLIMIT = 300000;

private static final int EFFIECTIVETIME = 600000;

private WritableSheet sheet;

private WritableWorkbook book;

private ArrayList<String> timeRecord = new ArrayList<>();

private ArrayList<String> powerRecord = new ArrayList<>();

private ArrayList<String> stateRecord = new ArrayList<>();

private ArrayList<Integer> weekList = new ArrayList<>();

private ArrayList<Integer> chargeTimeList = new ArrayList<>();

private ArrayList<Integer> shutTimeList = new ArrayList<>();

private ArrayList<Integer> restartTimeList = new ArrayList<>();

private ArrayList<Double> batteryLife = new ArrayList<>();

private ArrayList<Double> effectiveDischargeTime = new ArrayList<>();

private ArrayList<Integer> effectiveDischargePower = new ArrayList<>();

private double[] chargeTimeNum;

private double[] chargePowerNum;

private boolean dataFromat = true;

private int totalWeek = 1;

private int reStartLimitTime = 30000;

private final String SAVEPATH = Environment.getExternalStorageDirectory().getAbsolutePath() + "/chargeanalysis.xls";

private final String STARTCHARGINGKEY = "开始充电";

private final String ENDCHARGINGKEY = "结束充电";

private final String RESTARTCHARGINGKEY = "开机充电";

private final String RESTARTNOTCHARGEKEY = "开机放电";

private final String SHUNDOWNKEY = "系统关机";

public double[] getChargeTimeNum() { return chargeTimeNum; }

public double[] getChargePowerNum() { return chargePowerNum; }

public ArrayList<Double> getBatteryLife() { return batteryLife; }

public ArrayList<Integer> getChargeTimeList() { return chargeTimeList; }

public ArrayList<Double> getEffectiveDischargeTime() { return effectiveDischargeTime; }

public ArrayList<Integer> getEffectiveDischargePower() { return effectiveDischargePower; }

public boolean isdataFromat() { return dataFromat; }

//classify, filter and statistics the original data

public void arrangeData(String[] data, Context context) {

classifyData(data); //classify the txt to three ArrayLists

if (dataFromat == true){

correctAndFilter(); //adjust the ArrayList as required

for (int index = 0; index < stateRecord.size(); index++){

Log.e("filted", stateRecord.get(index));

}

chargeTimeNum = new double[]{0, 0, 0, 0};

chargePowerNum = new double[]{0, 0, 0, 0};

getWeekChargeFreq(timeRecord, powerRecord, stateRecord); //calculate the times every week

calculateBatteryLife();

getEffectiveChargingInterval();

}else {

Toast.makeText(context, context.getString(R.string.file\_error\_prompt), Toast.LENGTH\_LONG).show();

}

}

public void creat(Context context) throws WriteException {

if (dataFromat == true){

try {

book = Workbook.createWorkbook(new File(SAVEPATH));

Log.e(TAG, "getbook");

sheet = book.createSheet("充电记录", 0);

// create the sheet with seting it's name and sequence number

sheet.addCell(new Label(0, 0, "时间"));

sheet.addCell(new Label(1, 0, "电量"));

sheet.addCell(new Label(2, 0, "状态"));

sheet.addCell(new Label(3, 0, "充放电效率"));

sheet.setColumnView(5, 16);

sheet.addCell(new Label(5, 0, "充电起始时间"));

sheet.addCell(new Label(5, 1, "0:00 —— 6:00"));

sheet.addCell(new Label(5, 2, "6:00 —— 12:00"));

sheet.addCell(new Label(5, 3, "12:00 —— 18:00"));

sheet.addCell(new Label(5, 4, "18:00 —— 0:00"));

sheet.addCell(new Label(6, 0, "充电次数"));

sheet.setColumnView(7, 16);

sheet.addCell(new Label(7, 0, "充电起始电量"));

sheet.addCell(new Label(7, 1, "100% —— 75%"));

sheet.addCell(new Label(7, 2, "75% —— 50%"));

sheet.addCell(new Label(7, 3, "50% —— 25%"));

sheet.addCell(new Label(7, 4, "25% —— 0%"));

sheet.addCell(new Label(8, 0, "充电次数"));

sheet.addCell(new Label(9, 0, "周次"));

sheet.addCell(new Label(10, 0, "充电次数"));

sheet.addCell(new Label(11, 0, "关机次数"));

sheet.addCell(new Label(12, 0, "重启次数"));

for (int i = 0; i < timeRecord.size(); i++) {

sheet.addCell(new Label(0, i + 1, timeRecord.get(i)));

sheet.addCell(new Label(1, i + 1, powerRecord.get(i)));

sheet.addCell(new Label(2, i + 1, stateRecord.get(i)));

}

sheet.addCell(new Number(3, timeRecord.size(), 0));

addAverageBatteryLife();

for (int i = 0; i < chargePowerNum.length; i++){

sheet.addCell(new Number(6, i + 1, chargeTimeNum[i]));

sheet.addCell(new Number(8, i + 1, chargePowerNum[i]));

}

for (int i = 0; i < weekList.size(); i++){

sheet.addCell(new Number(9, i + 1, i + 1));

sheet.addCell(new Number(10, i + 1, chargeTimeList.get(i)));

sheet.addCell(new Number(11, i + 1, shutTimeList.get(i)));

sheet.addCell(new Number(12, i + 1, restartTimeList.get(i)));

}

book.write();

book.close();

} catch (IOException e) {

e.printStackTrace();

}

}else {

Toast.makeText(context, context.getString(R.string.excel\_error\_prompt), Toast.LENGTH\_LONG).show();

}

}

private void calculateBatteryLife() {

double expendTime, expendPower;

for (int i = 0; i < stateRecord.size() - 1; i++){

expendTime = (timeToLong(timeRecord.get(i + 1)) - timeToLong(timeRecord.get(i))) / 1000;

expendPower = powerToInt(powerRecord.get(i + 1)) - powerToInt(powerRecord.get(i));

if (expendPower != 0){

Log.e("expendPower", String.valueOf(expendPower));

batteryLife.add( (expendTime / expendPower));

}else {

batteryLife.add((double) 0);

}

}

}

private void getEffectiveChargingInterval() {

for (int stateIndex = 1; stateIndex < stateRecord.size(); stateIndex++){

if (stateRecord.get(stateIndex).contains(STARTCHARGINGKEY) && stateRecord.get(stateIndex - 1).contains(ENDCHARGINGKEY)){

double pastTime = timeToLong(timeRecord.get(stateIndex)) - timeToLong(timeRecord.get(stateIndex - 1));

if (pastTime > EFFIECTIVETIME){

effectiveDischargeTime.add(pastTime / 3600000);

effectiveDischargePower.add(powerToInt(powerRecord.get(stateIndex - 1)) - powerToInt(powerRecord.get(stateIndex)));

}

}

}

}

/\*\*

\*set the format for the sheet

\*/

private WritableCellFormat drawCell( ) {

WritableFont font = new WritableFont(WritableFont.createFont("楷体"), 11, WritableFont.BOLD);

WritableCellFormat format = new WritableCellFormat(font);

try {

format.setBackground(Colour.LIGHT\_GREEN);

//set the horizontally

format.setAlignment(Alignment.CENTRE);

//set the Vertical

format.setVerticalAlignment(VerticalAlignment.CENTRE);

} catch (WriteException e) {

e.printStackTrace();

}

return format;

}

private void classifyData(String[] data) {

int dataType = 3;

final int firstTypeNumber = 0;

final int secondTypeNumber = 1;

final int thirdTypeNumber = 2;

int standardTimeLength = 19;

int minPowerLength = 2;

int maxPowerLength = 4;

int minStateLength = 3;

int maxStateLength = 5;

for (int dataIndex = 0; dataIndex < data.length; dataIndex++){

switch (dataIndex % dataType){

case firstTypeNumber:

{

timeRecord.add(data[dataIndex]);

Log.e("length", String.valueOf(timeRecord.get(dataIndex / dataType).length()));

if (timeRecord.get(dataIndex / dataType).length() != standardTimeLength){

dataFromat = false;

}

break;

}

case secondTypeNumber:

{

powerRecord.add(data[dataIndex]);

Log.e("length", String.valueOf(powerRecord.get(dataIndex / dataType).length()));

if (powerRecord.get(dataIndex / dataType).length() > maxPowerLength || powerRecord.get(dataIndex / dataType).length() < minPowerLength){

dataFromat = false;

}

break;

}

case thirdTypeNumber:

{

stateRecord.add(data[dataIndex]);

Log.e("length", String.valueOf(stateRecord.get(dataIndex / dataType).length()));

if (stateRecord.get(dataIndex / dataType).length() > maxStateLength || stateRecord.get(dataIndex / dataType).length() < minStateLength){

dataFromat = false;

}

break;

}

}

}

if (timeRecord.size() != powerRecord.size() && timeRecord.size() != stateRecord.size()){

dataFromat = false;

}

for (int timeIndex = 0; timeIndex < timeRecord.size(); timeIndex++){

int originalWeek = (int) (timeToLong(timeRecord.get(timeIndex)) - timeToLong(timeRecord.get(0))) / MSECINWEEK + 1;

if (originalWeek > totalWeek){

totalWeek = originalWeek;

}

}

for (int weekIndex = 0; weekIndex < totalWeek; weekIndex++){

weekList.add(weekIndex + 1);

}

}

@Nullable

private void getWeekChargeFreq(ArrayList<String> originalTime, ArrayList<String> originalPower, ArrayList<String> originalState) {

ArrayList<String> time = new ArrayList<>();

ArrayList<String> shutTime = new ArrayList<>();

ArrayList<String> restartTime = new ArrayList<>();

for (int originalIndex = 0; originalIndex < originalTime.size(); originalIndex++){

if (originalState.get(originalIndex).contains(STARTCHARGINGKEY) || originalState.get(originalIndex).contains(RESTARTCHARGINGKEY)){

time.add(originalTime.get(originalIndex));

int hour = cutTimeToHour(originalTime.get(originalIndex));

if (hour < 6){

chargeTimeNum[0]++;

}else if (hour < 12){

chargeTimeNum[1]++;

}else if (hour < 18){

chargeTimeNum[2]++;

}else {

chargeTimeNum[3]++;

}

int power = powerToInt(originalPower.get(originalIndex));

if (power > 75){

chargePowerNum[0]++;

}else if (power > 50){

chargePowerNum[1]++;

}else if (power > 25){

chargePowerNum[2]++;

}else {

chargePowerNum[3]++;

}

}else if (originalState.get(originalIndex).contains(SHUNDOWNKEY)){

if (originalIndex < originalState.size() - 1){

long intervalTime = timeToLong(originalTime.get(originalIndex + 1)) - timeToLong(originalTime.get(originalIndex));

if (intervalTime < RESTARTLIMIT){

restartTime.add(originalTime.get(originalIndex));

}else {

shutTime.add(originalTime.get(originalIndex));

}

}

}

}

chargeTimeList = weekStatistics(time);

Log.e("week", String.valueOf(chargeTimeList.size()));

shutTimeList = weekStatistics(shutTime);

restartTimeList = weekStatistics(restartTime);

}

private void addAverageBatteryLife() throws WriteException {

double expendPower = 0;

double expendTime = 0;

//构造格式：ARIAL字体、10号、粗体、非斜体、无下划线、黑色

WritableFont wordsFont = new WritableFont(WritableFont.ARIAL,10,WritableFont.NO\_BOLD,

false, UnderlineStyle.NO\_UNDERLINE, Colour.BLACK);

WritableCellFormat format = new WritableCellFormat(wordsFont);

//文字垂直居中对齐

format.setAlignment(Alignment.RIGHT);

// totalx2Format.setVerticalAlignment(VerticalAlignment.JUSTIFY);

for (int i = 0; i < stateRecord.size() - 1; i++){

expendTime = (timeToLong(timeRecord.get(i + 1)) - timeToLong(timeRecord.get(i))) / 1000;

expendPower = powerToInt(powerRecord.get(i + 1)) - powerToInt(powerRecord.get(i));

if (expendPower != 0){

Log.e("expendPower", String.valueOf(expendPower));

try {

DecimalFormat df = new DecimalFormat("0.00");

sheet.addCell(new Label(3, i + 1, df.format(expendTime / expendPower / 60), format));

} catch (WriteException e) {

e.printStackTrace();

}

}else {

try {

sheet.addCell(new Label(3, i + 1, "0", format));

} catch (WriteException e) {

e.printStackTrace();

}

}

// }

}

}

private ArrayList<Integer> weekStatistics(ArrayList<String> time){

ArrayList<Integer> countList = new ArrayList<>();

for (int listIndex = 0; listIndex < weekList.size(); listIndex++){

countList.add(0);

}

Log.e("currentWeek", String.valueOf(time.size()));

if (time.size() > 0){

for (int timeIndex = 0; timeIndex < time.size(); timeIndex++){

int currentWeek = (int) ((timeToLong(time.get(timeIndex)) - timeToLong(timeRecord.get(0))) / MSECINWEEK);

countList.set(currentWeek, countList.get(currentWeek) + 1);

}

}else {

}

return countList;

}

/\*

\*filter the repeating charge data when the power is 100%

\*\*/

private void rechargeFilter(ArrayList<String> state, ArrayList<String> power) {

ArrayList<Integer> removeList = new ArrayList<>();

if (state != null ){

if (state.size() > 1){

for (int stateIndex = 1; stateIndex < state.size() - 1; stateIndex++){

// if (power.get(stateIndex).equals("100%") && power.get(stateIndex - 1).equals("100%")){

if (power.get(stateIndex).equals(power.get(stateIndex - 1)) ){

if (state.get(stateIndex).contains(STARTCHARGINGKEY) || state.get(stateIndex).contains(ENDCHARGINGKEY)){

// removeList.add(stateIndex - 1);

removeList.add(stateIndex);

// stateIndex++;

}else {

}

}

}

}

}

removeRecord(removeList);

}

/\*

\*filter the repeating data because of the process reboot

\*\*/

private void processRebootFilter(ArrayList<String> state) {

ArrayList<Integer> removeList = new ArrayList<>();

if (state != null){

if (state.size() > 1){

for (int stateIndex = 1; stateIndex < state.size() ; stateIndex++){

if (state.get(stateIndex).equals(state.get(stateIndex - 1))){

removeList.add(stateIndex);

}

}

}

removeRecord(removeList);

}

}

/\*

\*correct the data from endCharging to startDischarge when after Boot

\*\*/

private void startDischargeCorrect(ArrayList<String> state, ArrayList<String> time) {

ArrayList<Integer> removeList = new ArrayList<>();

if (state.size() > 1){

for (int stateIndex = 0; stateIndex < state.size() - 2; stateIndex++){

int expendTime = (int) (timeToLong(time.get(stateIndex + 1)) - timeToLong(time.get(stateIndex)));

Log.e("expendTime", String.valueOf(expendTime));

if (state.get(stateIndex).contains(SHUNDOWNKEY)){

if(expendTime <= reStartLimitTime){

removeList.add(stateIndex + 1);

if (state.get(stateIndex + 2).contains(ENDCHARGINGKEY) && expendTime > reStartLimitTime){

stateRecord.set(stateIndex + 2, RESTARTNOTCHARGEKEY);

Log.e("correct", "once");

}

}else if (state.get(stateIndex + 1).contains(ENDCHARGINGKEY) && expendTime > reStartLimitTime){

stateRecord.set(stateIndex + 1, RESTARTNOTCHARGEKEY);

Log.e("correct", "once");

}

}

}

}

removeRecord(removeList);

}

private void restartDelayCorrect(ArrayList<String> state){

ArrayList<Integer> removeList = new ArrayList<>();

if (state.size() > 1){

for (int stateIndex = 2; stateIndex < state.size() ; stateIndex++){

if ((state.get(stateIndex).contains(RESTARTCHARGINGKEY) || state.get(stateIndex).contains(RESTARTNOTCHARGEKEY))){

if (!state.get(stateIndex - 2).contains(SHUNDOWNKEY)){

removeList.add(stateIndex - 2);

}

if (!state.get(stateIndex - 1).contains(SHUNDOWNKEY)){

removeList.add(stateIndex - 1);

}

}

}

}

removeRecord(removeList);

}

private void correctAndFilter() {

startDischargeCorrect(stateRecord, timeRecord); //correct the error in the last record's version for startDischargeState

rechargeFilter(stateRecord, powerRecord); //remove the rechange data

processRebootFilter(stateRecord); //remove the repeating data

restartDelayCorrect(stateRecord);//remove the bootdelay

}

private void removeRecord(ArrayList<Integer> removeList) {

if (removeList.size() > 0){

for (int removeIndex = removeList.size() - 1; removeIndex >= 0; removeIndex--) {

timeRecord.remove((int)removeList.get(removeIndex));

powerRecord.remove((int)removeList.get(removeIndex));

stateRecord.remove((int)removeList.get(removeIndex));

}

}

}

private long timeToLong(String time) {

SimpleDateFormat sdf = new SimpleDateFormat( "yyyy-MM-dd HH:mm:ss" );

Date data = new Date();

try {

data = sdf.parse(time);

} catch (ParseException e) {

e.printStackTrace();

}

return data.getTime();

}

private int powerToInt(String power) {

power = power.replace("%", "");

return Integer.parseInt(power);

}

private int cutTimeToHour(String time) {

int hour = 0;

int hourStartIndex = 11;

int hourEndIndex = 13;

int standardTimelength = 19;

if (time.length() > 10){

hour = Integer.parseInt(time.substring(hourStartIndex, hourEndIndex));

Log.e("hour", String.valueOf(hour));

}

return hour;

}

}