|  |
| --- |
| RedcapHelper.java |

**package** diagnose.uvfree.uvfree;   
   
**import** android.content.Context;   
**import** android.graphics.Bitmap;   
**import** android.graphics.BitmapFactory;   
**import** android.net.http.AndroidHttpClient;   
**import** android.os.StrictMode;   
**import** android.os.SystemClock;   
**import** android.util.Log;   
   
**import** org.apache.commons.io.IOUtils;   
**import** org.apache.http.Header;   
**import** org.apache.http.HttpEntity;   
**import** org.apache.http.HttpResponse;   
**import** org.apache.http.NameValuePair;   
**import** org.apache.http.client.HttpClient;   
**import** org.apache.http.client.methods.HttpOptions;   
**import** org.apache.http.client.methods.HttpPost;   
**import** org.apache.http.conn.MultihomePlainSocketFactory;   
**import** org.apache.http.entity.mime.HttpMultipartMode;   
**import** org.apache.http.entity.mime.content.ByteArrayBody;   
**import** org.apache.http.entity.mime.content.InputStreamBody;   
**import** org.apache.http.impl.client.DefaultHttpClient;   
**import** org.apache.http.message.BasicHeader;   
**import** org.apache.http.message.BasicNameValuePair;   
**import** org.apache.http.client.entity.UrlEncodedFormEntity;   
   
**import** org.apache.http.params.HttpParams;   
**import** org.apache.http.util.EntityUtils;   
**import** org.apache.http.entity.mime.MultipartEntityBuilder;   
   
**import** java.io.ByteArrayInputStream;   
**import** java.io.ByteArrayOutputStream;   
**import** java.io.File;   
**import** java.io.FileInputStream;   
**import** java.io.IOException;   
**import** java.io.InputStream;   
**import** java.io.UnsupportedEncodingException;   
**import** java.util.ArrayList;   
**import** java.util.List;   
   
*/\*\* RedcapHelper   
 \* This class will take a list of diagInstance objects to upload, pull the appropriate local   
 \* database information, including AbnormPhotos related to the diagInstances, and the API   
 \* Key and Redcap Server URL that have been set up by each user. It will take that information   
 \* and the contained fields and create the appropriate files, then open up a connection to   
 \* the Redcap server and send the appropriate file by a POST connection.   
 \*/*   
**public class** RedcapHelper {   
 **private** List<DiagInstance> diagInstanceList;   
 **private** List<AbnormPhoto> abnormPhotoList;   
 **private** String APIKEY;   
 **private** String RedcapURL;   
 String im1output;   
 String sampleout;   
   
 *// Constructors*   
 **public** RedcapHelper(List<DiagInstance> diagInstanceList, List<AbnormPhoto> abnormPhotoList, Context context) {   
 **this**.diagInstanceList = diagInstanceList;   
 **this**.abnormPhotoList = abnormPhotoList;   
   
 *// Pull the API Key and Redcap URL from the User database*   
 UserDBHelper udb = **new** UserDBHelper(context);   
 User currentUser = udb.getActiveUser();   
 **this**.APIKEY = currentUser.getAPIKEY();   
 **this**.RedcapURL = currentUser.getRedcapURL();   
 }   
   
 *// Upload to Redcap*   
 **public** String upload() {   
   
 *// Define Redcap Variable Names*   
 String RC\_recordID = **"record\_id"**;   
 String RC\_nurseName = **"nursename"**;   
 String RC\_nurseEmail = **"nurseemail"**;   
 String RC\_patientName = **"patientname"**;   
 String RC\_patientDOB = **"patientdob"**;   
 String RC\_patientGender = **"patientgender"**;   
 String RC\_capDateTime = **"capdatetime"**;   
 String RC\_uniquePatientID = **"unique\_patient\_id"**;   
 String RC\_gpslat = **"gpslat"**;   
 String RC\_gpslng = **"gpslng"**;   
 String RC\_bodyRegion = **"bodyregion"**;   
 String RC\_bodySide = **"bodyside"**;   
 String RC\_image1pre = **"image1\_pre"**;   
 String RC\_image1post = **"image1\_post"**;   
 String RC\_image2pre = **"image2\_pre"**;   
 String RC\_image2post = **"image2\_post"**;   
   
 *// The fields to be sent to the Redcap Server*   
 String RedcapFields = RC\_recordID + **","** + RC\_nurseName + **","** + RC\_nurseEmail + **","**   
 + RC\_patientName + **","** + RC\_patientDOB + **","** + RC\_patientGender + **","**   
 + RC\_capDateTime + **","** + RC\_uniquePatientID + **","** + RC\_gpslat + **","** + RC\_gpslng   
 + **","** + RC\_bodyRegion + **","** + RC\_bodySide + **"\n"**;   
   
 String qt = **"\""**;   
 String di = **"\",\""**; *// Divider between fields, basically prints this: ","*   
   
 String InstanceData = **""**;   
   
 **for** (**int** i = 0; i < diagInstanceList.size(); i++ ) {   
   
 DiagInstance dInst = diagInstanceList.get(i); *// The DiagInstance to be uploaded*   
   
 *// Build list of AbnormPhotos that match the UID*   
 List<AbnormPhoto> aPhotos = **new** ArrayList<AbnormPhoto>();   
 **for** (**int** j = 0; j < abnormPhotoList.size(); j++ ) {   
 **if**( abnormPhotoList.get(j).getParentUID().equals(dInst.getUID())) {   
 aPhotos.add(abnormPhotoList.get(j));   
 }   
 }   
   
 *// Convert Genders in dInst to 1/2 that Redcap supports*   
 **if** (dInst.getPatientGender().equals(**"Male"**)) {   
 dInst.setPatientGender(**"1"**);   
 } **else** {   
 dInst.setPatientGender(**"2"**);   
 }   
   
 String RecordData = **""**;   
   
 *// Make a list of BodyLocations and BodySides for AbnormPhotos related to this dInst*   
 **for**(**int** j = 0; j < aPhotos.size(); j++) {   
 List<String> decodedLoc = decodeBodyLoc(aPhotos.get(j).getBodyLocation());   
 **int** loc = Integer.parseInt(decodedLoc.get(0));   
 **int** side = Integer.parseInt(decodedLoc.get(1));   
   
 *// Add the body side*   
 String bodySide = Integer.toString(side);   
 String bodyLoc = **""**;   
   
 *// Change location to reflect SNOMED CT Codes, and add to the array*   
 **if**( side == 1 ) {   
 **if** ( loc == 1 ) { bodyLoc = **"361355005"**; }   
 **else if** ( loc == 2 ) { bodyLoc = **"368106002"**; }   
 **else if** ( loc == 3 ) { bodyLoc = **"368224007"**; }   
 **else if** ( loc == 4 ) { bodyLoc = **"368107006"**; }   
 **else if** ( loc == 5 ) { bodyLoc = **"368225008"**; }   
 **else if** ( loc == 6 ) { bodyLoc = **"264242009"**; }   
 **else if** ( loc == 7 ) { bodyLoc = **"302553009"**; }   
 **else if** ( loc == 8 ) { bodyLoc = **"209570001"**; }   
 **else if** ( loc == 9 ) { bodyLoc = **"213289002"**; }   
 **else if** ( loc == 10 ) { bodyLoc = **"209672000"**; }   
 **else if** ( loc == 11 ) { bodyLoc = **"213384005"**; }   
 } **else** {   
 **if** ( loc == 1 ) { bodyLoc = **"361355005"**; }   
 **else if** ( loc == 2 ) { bodyLoc = **"368107006"**; }   
 **else if** ( loc == 3 ) { bodyLoc = **"368225008"**; }   
 **else if** ( loc == 4 ) { bodyLoc = **"368106002"**; }   
 **else if** ( loc == 5 ) { bodyLoc = **"368224007"**; }   
 **else if** ( loc == 6 ) { bodyLoc = **"35549004"**; }   
 **else if** ( loc == 7 ) { bodyLoc = **"37822005"**; }   
 **else if** ( loc == 8 ) { bodyLoc = **"209672000"**; }   
 **else if** ( loc == 9 ) { bodyLoc = **"213384005"**; }   
 **else if** ( loc == 10 ) { bodyLoc = **"209570001"**; }   
 **else if** ( loc == 11 ) { bodyLoc = **"213289002"**; }   
 }   
   
 *// Build upload string for each record*   
 RecordData = RecordData + **"\n"** + qt + dInst.getUID() + **"r"** + Integer.toString(j+1)   
 + di + dInst.getNurse() + di + dInst.getNurseEmail() + di + dInst.getPatient()   
 + di + dInst.getPatientDOB() + di + dInst.getPatientGender() + di + dInst.getDate()   
 + **" "** + dInst.getTime() + di + dInst.getPatientUID() + di + dInst.getLatitude()   
 + di + dInst.getLongitude() + di + bodyLoc + di + bodySide + qt;   
 }   
   
 InstanceData = InstanceData + RecordData;   
   
 }   
   
 *// Prepare upload of records (not files)*   
 List<NameValuePair> eparams = **new** ArrayList<NameValuePair>();   
   
 *// Add appropriate API parameters*   
 eparams.add(**new** BasicNameValuePair(**"token"**, APIKEY));   
 eparams.add(**new** BasicNameValuePair(**"content"**, **"record"**));   
 eparams.add(**new** BasicNameValuePair(**"format"**, **"csv"**));   
 eparams.add(**new** BasicNameValuePair(**"type"**, **"flat"**));   
 eparams.add(**new** BasicNameValuePair(**"data"**, RedcapFields+InstanceData));   
 eparams.add(**new** BasicNameValuePair(**"returnContent"**, **"ids"**));   
 eparams.add(**new** BasicNameValuePair(**"returnFormat"**, **"csv"**));   
   
 *// Open HTTP POST Connection*   
 HttpClient httpclient = **new** DefaultHttpClient();   
 HttpPost httppost = **new** HttpPost(RedcapURL);   
   
 String httpOutput = **"No Output"**;   
   
 *// Set Networking to run on Main Thread (bad practice / change if time)*   
 StrictMode.ThreadPolicy policy = **new** StrictMode.ThreadPolicy.Builder().permitAll().build();   
 StrictMode.setThreadPolicy(policy);   
   
 *// Send POST connection to Redcap Server*   
 **try** {   
 httppost.setEntity(**new** UrlEncodedFormEntity(eparams));   
 HttpResponse response = httpclient.execute(httppost);   
 HttpEntity entity = response.getEntity();   
 httpOutput = EntityUtils.toString(entity);   
 } **catch** (UnsupportedEncodingException uee) {   
 Log.d(**"Unsupported Encoding"**, uee.getMessage());   
 } **catch** (IOException ioe) {   
 Log.d(**"IO Error"**, ioe.getMessage());   
 }   
   
   
 */\*\*\*\*\* Upload Photos Here \*\*\*\*\*\*/*   
   
 */\*\*\*\*\*\*\*\*\*\*\*\*\* TEST CODE FOR PHOTO UPLOAD \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*   
 *// Get the DiagInstance for testing*   
 DiagInstance dInst = diagInstanceList.get(0);   
   
 *// Get an associated AbnormPhoto*   
 AbnormPhoto toUpload = **new** AbnormPhoto();   
   
 **for**(**int** j = 0; j < abnormPhotoList.size(); j++ ) {   
 **if**( abnormPhotoList.get(j).getParentUID().equals(dInst.getUID())) {   
 toUpload = abnormPhotoList.get(j);   
 **break**;   
 }   
 }   
   
 String corrRecordID = dInst.getUID() + **"r1"**;   
 File img = **new** File(toUpload.getImage1());   
   
 ByteArrayOutputStream baos = **new** ByteArrayOutputStream();   
 Bitmap bmpimg = BitmapFactory.decodeFile(img.getAbsolutePath());   
   
 bmpimg.compress(Bitmap.CompressFormat.PNG, 100, baos);   
 **byte**[] imgByte = baos.toByteArray();   
   
 HttpClient httpClientImg = **new** DefaultHttpClient();   
 HttpPost httpPostImg = **new** HttpPost(RedcapURL);   
   
 String boundary = **"-------------"** + System.currentTimeMillis();   
   
 httpPostImg.setHeader(**"Content-type"**, **"multipart/form-data; boundary="**+boundary);   
   
 ByteArrayBody bab = **new** ByteArrayBody(imgByte, **"img.png"**);   
   
 HttpEntity entityImg = MultipartEntityBuilder.create()   
 .setMode(HttpMultipartMode.BROWSER\_COMPATIBLE)   
 .setBoundary(boundary)   
 .addTextBody(**"token"**,APIKEY)   
 .addTextBody(**"content"**,**"file"**)   
 .addTextBody(**"action"**,**"import"**)   
 .addTextBody(**"record"**,corrRecordID)   
 .addTextBody(**"field"**,**"image1\_pre"**)   
 .addPart(**"file"**,bab)   
 .build();   
   
 httpPostImg.setEntity(entityImg);   
   
 String imgout = **""**;   
 String lb = System.getProperty(**"line.separator"**);   
   
 **try** {   
 HttpResponse response = httpClientImg.execute(httpPostImg);   
 HttpEntity entity = response.getEntity();   
 imgout = EntityUtils.toString(entity);   
 } **catch**( IOException ioe) {   
 Log.d(**"File IO Error"**, ioe.getMessage());   
 }   
   
 **return** httpOutput + lb + imgout;   
   
   
   
   
 *// Upload all files*   
 */\*   
 for(int i = 0; i < diagInstanceList.size(); i++) {   
 DiagInstance dInst = diagInstanceList.get(i);   
   
 // Build list of AbnormPhotos that match the UID   
 List<AbnormPhoto> aPhotos = new ArrayList<AbnormPhoto>();   
 for (int j = 0; j < abnormPhotoList.size(); j++ ) {   
 if( abnormPhotoList.get(j).getParentUID().equals(dInst.getUID())) {   
 aPhotos.add(abnormPhotoList.get(j));   
 }   
 }   
   
 for (int j = 0; j < aPhotos.size(); j++) {   
 // Do file upload here - reason is to encode record\_id correctly with r1, r2, etc.   
 AbnormPhoto toUpload = aPhotos.get(j);   
   
 // Image 1 Pre   
 File im1pre = new File(toUpload.getImage1());   
 byte[] im1preDat = null;   
 try {   
 InputStream is = new FileInputStream(im1pre);   
 im1preDat = IOUtils.toByteArray(is);   
 } catch(Exception e) {}   
   
 InputStreamBody im1preISB = new InputStreamBody(new ByteArrayInputStream(im1preDat), toUpload.getImage1());   
 MultipartEntityBuilder im1preMPEB = MultipartEntityBuilder.create();   
   
 im1preMPEB.addTextBody("token", APIKEY);   
 im1preMPEB.addTextBody("content", "file");   
 im1preMPEB.addTextBody("action", "import");   
 im1preMPEB.addTextBody("record", dInst.getUID()+"r"+Integer.toString(j+1));   
 im1preMPEB.addTextBody("field", RC\_image1pre);   
 im1preMPEB.addPart("file", im1preISB);   
   
   
   
 sampleout = APIKEY + " " + dInst.getUID()+"r"+Integer.toString(j+1) + " " + RC\_image1pre + " " + im1preISB.toString();   
   
 im1output = "ERROR: No Image 1 Output";   
   
 // Send it   
 httppost.setEntity(im1preMPEB.build());   
 try {   
 HttpResponse imageResponse = httpclient.execute(httppost);   
 HttpEntity entity = imageResponse.getEntity();   
 im1output = EntityUtils.toString(entity);   
 } catch (IOException ioe) {   
 Log.d("File IO Error", ioe.getMessage());   
 }   
   
 // Image 1 Post   
 if(toUpload.getImage1post() != null) {   
 if (!toUpload.getImage1post().equals("None")) {   
 File im1post = new File(toUpload.getImage1post());   
 byte[] im1postDat = null;   
 try {   
 InputStream is1post = new FileInputStream(im1post);   
 im1postDat = IOUtils.toByteArray(is1post);   
 } catch (Exception e) {   
 }   
   
 InputStreamBody im1postISB = new InputStreamBody(new ByteArrayInputStream(im1postDat), toUpload.getImage1post());   
 MultipartEntityBuilder im1postMPEB = MultipartEntityBuilder.create();   
   
 im1postMPEB.addTextBody("token", APIKEY);   
 im1postMPEB.addTextBody("content", "file");   
 im1postMPEB.addTextBody("action", "import");   
 im1postMPEB.addTextBody("record", dInst.getUID() + "r" + Integer.toString(j + 1));   
 im1postMPEB.addTextBody("field", RC\_image1post);   
 im1postMPEB.addPart("file", im1postISB);   
   
 // Send it   
 httppost.setEntity(im1postMPEB.build());   
 try {   
 httpclient.execute(httppost);   
 } catch (IOException ioe) {   
 Log.d("File IO Error", ioe.getMessage());   
 }   
 }   
 }   
   
 // Image 2 Pre   
 File im2pre = new File(toUpload.getImage2());   
 byte[] im2preDat = null;   
 try {   
 InputStream is2pre = new FileInputStream(im2pre);   
 im2preDat = IOUtils.toByteArray(is2pre);   
 } catch(Exception e) {}   
   
 InputStreamBody im2preISB = new InputStreamBody(new ByteArrayInputStream(im2preDat), toUpload.getImage2());   
 MultipartEntityBuilder im2preMPEB = MultipartEntityBuilder.create();   
   
 im2preMPEB.addTextBody("token", APIKEY);   
 im2preMPEB.addTextBody("content", "file");   
 im2preMPEB.addTextBody("action", "import");   
 im2preMPEB.addTextBody("record", dInst.getUID()+"r"+Integer.toString(j+1));   
 im2preMPEB.addTextBody("field", RC\_image2pre);   
 im2preMPEB.addPart("file", im2preISB);   
   
 // Send it   
 httppost.setEntity(im2preMPEB.build());   
 try {   
 httpclient.execute(httppost);   
 } catch (IOException ioe) {   
 Log.d("File IO Error", ioe.getMessage());   
 }   
   
 // Image 2 Post   
 if(toUpload.getImage2post() != null ) {   
 if (!toUpload.getImage2post().equals("None")) {   
 File im2post = new File(toUpload.getImage2post());   
 byte[] im2postDat = null;   
 try {   
 InputStream is2post = new FileInputStream(im2post);   
 im2postDat = IOUtils.toByteArray(is2post);   
 } catch (Exception e) {   
 }   
   
 InputStreamBody im2postISB = new InputStreamBody(new ByteArrayInputStream(im2postDat), toUpload.getImage2post());   
 MultipartEntityBuilder im2postMPEB = MultipartEntityBuilder.create();   
   
 im2postMPEB.addTextBody("token", APIKEY);   
 im2postMPEB.addTextBody("content", "file");   
 im2postMPEB.addTextBody("action", "import");   
 im2postMPEB.addTextBody("record", dInst.getUID() + "r" + Integer.toString(j + 1));   
 im2postMPEB.addTextBody("field", RC\_image2post);   
 im2postMPEB.addPart("file", im2postISB);   
   
 // Send it   
 httppost.setEntity(im2postMPEB.build());   
 try {   
 httpclient.execute(httppost);   
 } catch (IOException ioe) {   
 Log.d("File IO Error", ioe.getMessage());   
 }   
 }   
 }   
 }   
 }\*/*   
   
 *//return httpOutput;*   
 *//return "Image Status: "+ im1output + " | " + httpOutput + sampleout;*   
 }   
   
 **static public** List<String> decodeBodyLoc( String bodyLocation ) {   
 *// Split the string at ,*   
 String[] splitFirst = bodyLocation.split(**","**);   
   
 *// Split the resulting Strings at colons (:) and pull the resulting numbers*   
 String location = splitFirst[0].split(**":"**)[1];   
 String side = splitFirst[1].split(**":"**)[1];   
 String xCoord = splitFirst[2].split(**":"**)[1];   
 String yCoord = splitFirst[3].split(**":"**)[1];   
   
 *// Set up a list of strings as the output, add the strings from above, and return it*   
 List<String> output = **new** ArrayList<String>();   
 output.add(location);   
 output.add(side);   
 output.add(xCoord);   
 output.add(yCoord);   
   
 **return** output;   
 }   
}