**Design Part Coding**

**Main.xml**

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

>

<Button

android:id=*"@+id/auth\_button"*

android:text=*"@string/lik"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

/>

<ScrollView xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*>

<LinearLayout

android:id=*"@+id/logged\_in\_display"*

android:orientation=*"vertical"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"0dp"*

android:layout\_weight=*"1"* >

<TableLayout

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:orientation=*"vertical"* >

<TableRow >

<Button

android:id=*"@+id/store"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:background=*"@drawable/new\_button"* />

<TextView

android:id=*"@+id/tv1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Write Text"*

android:textSize=*"20sp"*

android:layout\_marginLeft=*"10dp"*

/>

</TableRow>

<TableRow >

<Button

android:id=*"@+id/photo\_button"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:background=*"@drawable/second"* />

<TextView

android:id=*"@+id/tv2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Take picture"*

android:textSize=*"20sp"*

android:layout\_marginLeft=*"10dp"*

/>

</TableRow>

<TableRow >

<Button

android:id=*"@+id/video\_button"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:background=*"@drawable/third"* />

<TextView

android:id=*"@+id/tv3"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Record Video"*

android:textSize=*"20sp"*

android:layout\_marginLeft=*"10dp"*

/>

</TableRow>

<TableRow >

<Button

android:id=*"@+id/audio\_button"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:background=*"@drawable/third"* />

<TextView

android:id=*"@+id/tv3"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Record Audio"*

android:textSize=*"20sp"*

android:layout\_marginLeft=*"10dp"*

/>

</TableRow>

<TableRow >

<Button

android:id=*"@+id/list\_button"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:background=*"@drawable/fourth"* />

<TextView

android:id=*"@+id/tv4"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"View File"*

android:textSize=*"20sp"*

android:layout\_marginLeft=*"10dp"*

/>

</TableRow>

</TableLayout>

<ImageView

android:id=*"@+id/image\_view"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

/>

</LinearLayout>

</ScrollView>

</LinearLayout>

**Background Coding**

**Home.Java**

\* Copyright (c) 2010-11 Dropbox, Inc.

package com.sweet.memories;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

import android.annotation.SuppressLint;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.Dialog;

import android.content.ActivityNotFoundException;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.SharedPreferences;

import android.content.SharedPreferences.Editor;

import android.content.pm.PackageManager;

import android.database.Cursor;

import android.location.Address;

import android.location.Geocoder;

import android.location.Location;

import android.location.LocationListener;

import android.location.LocationManager;

import android.media.MediaRecorder;

import android.net.Uri;

import android.os.Bundle;

import android.os.Environment;

import android.provider.MediaStore;

import android.util.Log;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.LinearLayout;

import android.widget.Toast;

import com.dropbox.client2.DropboxAPI;

import com.dropbox.client2.android.AndroidAuthSession;

import com.dropbox.client2.android.AuthActivity;

import com.dropbox.client2.session.AccessTokenPair;

import com.dropbox.client2.session.AppKeyPair;

import com.dropbox.client2.session.Session.AccessType;

import com.dropbox.client2.session.TokenPair;

@SuppressLint("ParserError")

public class Home extends Activity {

private static final String TAG = "Home";

private String addr = "Location Unknown";

// /////////////////////////////////////////////////////////////////////////

// Your app-specific settings. //

// /////////////////////////////////////////////////////////////////////////

// Replace this with your app key and secret assigned by Dropbox.

// Note that this is a really insecure way to do this, and you shouldn't

// ship code which contains your key & secret in such an obvious way.

// Obfuscation is good.

final static private String APP\_KEY = "id8km2kwy3z5pn7";

final static private String APP\_SECRET = "glz4rcwbtqdk3nu";

// If you'd like to change the access type to the full Dropbox instead of

// an app folder, change this value.

final static private AccessType ACCESS\_TYPE = AccessType.APP\_FOLDER;

// /////////////////////////////////////////////////////////////////////////

// End app-specific settings. //

// /////////////////////////////////////////////////////////////////////////

// You don't need to change these, leave them alone.

final static private String ACCOUNT\_PREFS\_NAME = "prefs";

final static private String ACCESS\_KEY\_NAME = "ACCESS\_KEY";

final static private String ACCESS\_SECRET\_NAME = "ACCESS\_SECRET";

final static int REQUEST\_VIDEO\_CAPTURED = 2;

private static int REQUEST\_AUDIO\_RECORDER = 3;

String audio\_path;

private static int MEDIA\_TYPE\_IMAGE = 1;

private static int MEDIA\_TYPE\_VIDEO = 2;

Uri uriVideo = null;

DropboxAPI<AndroidAuthSession> mApi;

private boolean mLoggedIn;

// Android widgets

private Button mSubmit;

private LinearLayout mDisplay;

private Button mPhoto;

private Button Viewfile;

private Button mVideo;

private Button audio;

private ImageView mImage;

private final String PHOTO\_DIR = "/Photos/";

String data = null;

final static private int NEW\_PICTURE = 1;

private String mCameraFileName;

private String mVideoFileName;

MediaRecorder mRecorder;

LocationManager locationManager;

MyLocationListener locationListener;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

if (savedInstanceState != null) {

mCameraFileName = savedInstanceState.getString("mCameraFileName");

}

if (savedInstanceState != null) {

mCameraFileName = savedInstanceState.getString("mVideoFileName");

}

// We create a new AuthSession so that we can use the Dropbox API.

AndroidAuthSession session = buildSession();

mApi = new DropboxAPI<AndroidAuthSession>(session);

// Basic Android widgets

setContentView(R.layout.main);

locationListener = new MyLocationListener();

locationManager = (LocationManager) getSystemService(Context.LOCATION\_SERVICE);

locationManager.requestLocationUpdates(LocationManager.NETWORK\_PROVIDER, 60000, 500, locationListener);

checkAppKeySetup();

mSubmit = (Button) findViewById(R.id.auth\_button);

mSubmit.setOnClickListener(new OnClickListener() {

public void onClick(View v) {

// This logs you out if you're logged in, or vice versa

if (mLoggedIn) {

logOut();

} else {

// Start the remote authentication

mApi.getSession().startAuthentication(Home.this);

}

}

});

mDisplay = (LinearLayout) findViewById(R.id.logged\_in\_display);

// This is where a photo is displayed

mImage = (ImageView) findViewById(R.id.image\_view);

// This is the button to take a photo

mPhoto = (Button) findViewById(R.id.photo\_button);

mPhoto.setOnClickListener(new OnClickListener() {

public void onClick(View v) {

Intent intent = new Intent();

// Picture from camera

intent.setAction(MediaStore.ACTION\_IMAGE\_CAPTURE);

Uri fileUri = getOutputMediaFileUri(MEDIA\_TYPE\_IMAGE);

intent.putExtra(android.provider.MediaStore.EXTRA\_OUTPUT,

fileUri);

Log.i(TAG, "Importing New Picture: " + mCameraFileName);

try {

startActivityForResult(intent, NEW\_PICTURE);

} catch (ActivityNotFoundException e) {

showToast("There doesn't seem to be a camera.");

}

}

});

mVideo = (Button) findViewById(R.id.video\_button);

mVideo.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

Intent intent = new Intent();

intent.setAction(MediaStore.ACTION\_VIDEO\_CAPTURE);

Uri fileUri = getOutputMediaFileUri(MEDIA\_TYPE\_VIDEO);

mVideoFileName = fileUri.toString();

Log.v("Video uri", "Uri:" + fileUri);

startActivityForResult(intent, REQUEST\_VIDEO\_CAPTURED);

}

});

audio = (Button) findViewById(R.id.audio\_button);

audio.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View arg0) {

startRecording();

AlertDialog.Builder alert = new AlertDialog.Builder(Home.this);

alert.setCancelable(false);

alert.setTitle("Recording audio");

alert.setMessage("Please start your speech to record");

alert.setPositiveButton("Stop",

new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

stopRecording();

File file;

try {

file = new File(audio\_path);

UploadFile upload = new UploadFile( Home.this, mApi, PHOTO\_DIR, file);

upload.execute(); } catch (Exception e) {

Log.e("Audio upload error:",

"Audio upload error:");

e.printStackTrace();

}

}

});

alert.setNegativeButton("Cancel",

new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog,

int which) {

stopRecording();

}

});

alert.show();

}

});

// final EditText etext=(EditText)findViewById(R.id.etext);

Button store = (Button) findViewById(R.id.store);

store.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

final Dialog d = new Dialog(Home.this);

d.setContentView(R.layout.dialog);

d.setCancelable(true);

Button b = (Button) d.findViewById(R.id.bt);

b.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

d.dismiss();

EditText e = (EditText) d.findViewById(R.id.et);

data = e.getText().toString();

// Toast.makeText(Home.this,"Con: "+s,

// Toast.LENGTH\_SHORT).show();

storeText();

}

});

d.show();

// data=etext.getText().toString();

}

});

// This is the button to take a photo

Viewfile = (Button) findViewById(R.id.list\_button);

Viewfile.setOnClickListener(new OnClickListener() {

public void onClick(View v) {

ViewFileList download = new ViewFileList(Home.this, mApi,

PHOTO\_DIR, mImage);

download.execute();

}

});

// Display the proper UI state if logged in or not

setLoggedIn(mApi.getSession().isLinked());

}

private void startRecording() {

audio\_path = Environment.getExternalStorageDirectory()

.getAbsolutePath() + File.separator;

String timeStamp = new SimpleDateFormat("yyyy-MM-dd-kk-mm")

.format(new Date());

audio\_path += "audio\_";

audio\_path += timeStamp + ".3gp";

mRecorder = new MediaRecorder();

mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);

mRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE\_GPP);

mRecorder.setOutputFile(audio\_path);

mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR\_NB);

try {

mRecorder.prepare();

} catch (IOException e) {

Log.e("Audio record error", "prepare() failed");

}

mRecorder.start();

}

private void stopRecording() {

mRecorder.stop();

mRecorder.release();

mRecorder = null;

}

private Uri getOutputMediaFileUri(int type) {

return Uri.fromFile(getOutputMediaFile(type));

}

/\*\* Create a File for saving an image or video \*/

private File getOutputMediaFile(int type) {

// To be safe, you should check that the SDCard is mounted

// using Environment.getExternalStorageState() before doing this.

File mediaStorageDir = new File(

Environment

.getExternalStoragePublicDirectory(Environment.DIRECTORY\_PICTURES),

"MyCameraApp");

// This location works best if you want the created images to be shared

// between applications and persist after your app has been uninstalled.

// Create the storage directory if it does not exist

if (!mediaStorageDir.exists()) {

if (!mediaStorageDir.mkdirs()) {

Log.d("MyCameraApp", "failed to create directory");

return null;

}

}

// Create a media file name

String timeStamp = new SimpleDateFormat("yyyy-MM-dd-kk-mm-ss")

.format(new Date());

File mediaFile;

if (type == MEDIA\_TYPE\_IMAGE) {

mediaFile = new File(mediaStorageDir.getPath() + File.separator

+ "IMG\_" + timeStamp + ".jpg");

mCameraFileName = mediaFile.getPath();

} else if (type == MEDIA\_TYPE\_VIDEO) {

mediaFile = new File(mediaStorageDir.getPath() + File.separator

+ "VID\_" + timeStamp + ".mp4");

mVideoFileName = mediaFile.getPath();

} else {

return null;

}

return mediaFile;

}

@Override

protected void onSaveInstanceState(Bundle outState) {

outState.putString("mCameraFileName", mCameraFileName);

outState.putString("mVideoFileName", mVideoFileName);

super.onSaveInstanceState(outState);

}

@Override

protected void onResume() {

super.onResume();

AndroidAuthSession session = mApi.getSession();

// The next part must be inserted in the onResume() method of the

// activity from which session.startAuthentication() was called, so

// that Dropbox authentication completes properly.

if (session.authenticationSuccessful()) {

try {

// Mandatory call to complete the auth

session.finishAuthentication();

// Store it locally in our app for later use

TokenPair tokens = session.getAccessTokenPair();

storeKeys(tokens.key, tokens.secret);

setLoggedIn(true);

} catch (IllegalStateException e) {

showToast("Couldn't authenticate with Dropbox:"

+ e.getLocalizedMessage());

Log.i(TAG, "Error authenticating", e);

}

}

}

// This is what gets called on finishing a media piece to import

@Override

public void onActivityResult(int requestCode, int resultCode, Intent data) {

if (requestCode == NEW\_PICTURE) {

// return from file upload

if (resultCode == Activity.RESULT\_OK) {

Uri uri = null;

if (data != null) {

uri = data.getData();

}

if (uri == null && mCameraFileName != null) {

uri = Uri.fromFile(new File(mCameraFileName));

Log.v("Picture Uri", uri.toString() + " " + uri.getPath());

}

File file = new File(mCameraFileName);

Log.v("Picture file", "" + file.getPath());

if (uri != null) {

UploadFile upload = new UploadFile(Home.this, mApi,

PHOTO\_DIR, file);

upload.execute();

}

}

} else if (requestCode == REQUEST\_VIDEO\_CAPTURED) {

if (resultCode == RESULT\_OK)

{

uriVideo = data.getData();

// String uripath = data.getStringExtra("path");

Log.w(TAG, uriVideo.getPath() + "activity result video 1"

+ uriVideo.toString());

File file;

try {

file = new File(getRealPathFromURI(uriVideo));

Log.d("Video file", "File:" + file.getPath());

UploadFile upload = new UploadFile(Home.this, mApi,

PHOTO\_DIR, file);

upload.execute();

File fi = new File(file.getPath());

fi.delete();

} catch (Exception e) {

Log.e("Video upload error:", "Video upload error:");

e.printStackTrace();

}

// showToast("till capture");

} else if (resultCode == RESULT\_CANCELED) {

uriVideo = null;

Toast.makeText(Home.this, "Cancelled!", Toast.LENGTH\_LONG)

.show();

} else {

Log.w(TAG, "Unknown Activity Result from mediaImport: "

+ resultCode);

}

}

}

private String getRealPathFromURI(Uri contentURI) {

Cursor cursor = getContentResolver().query(contentURI, null, null,

null, null);

cursor.moveToFirst();

int idx = cursor.getColumnIndex(MediaStore.Images.ImageColumns.DATA);

return cursor.getString(idx);

}

private void logOut() {

// Remove credentials from the session

mApi.getSession().unlink();

// Clear our stored keys

clearKeys();

// Change UI state to display logged out version

setLoggedIn(false);

}

private void setLoggedIn(boolean loggedIn) {

mLoggedIn = loggedIn;

if (loggedIn) {

mSubmit.setText("Unlink from Dropbox");

mDisplay.setVisibility(View.VISIBLE);

} else {

mSubmit.setText("Link with Dropbox");

mDisplay.setVisibility(View.GONE);

mImage.setImageDrawable(null);

}

}

private void checkAppKeySetup() {

// Check to make sure that we have a valid app key

if (APP\_KEY.startsWith("CHANGE") || APP\_SECRET.startsWith("CHANGE")) {

showToast("You must apply for an app key and secret from developers.dropbox.com, and add them to the Home ap before trying it.");

finish();

return;

}

// Check if the app has set up its manifest properly.

Intent testIntent = new Intent(Intent.ACTION\_VIEW);

String scheme = "db-" + APP\_KEY;

String uri = scheme + "://" + AuthActivity.AUTH\_VERSION + "/test";

testIntent.setData(Uri.parse(uri));

PackageManager pm = getPackageManager();

if (0 == pm.queryIntentActivities(testIntent, 0).size()) {

showToast("URL scheme in your app's "

+ "manifest is not set up correctly. You should have a "

+ "com.dropbox.client2.android.AuthActivity with the "

+ "scheme: " + scheme);

finish();

}

}

private void showToast(String msg) {

Toast error = Toast.makeText(this, msg, Toast.LENGTH\_LONG);

error.show();

}

private String[] getKeys() {

SharedPreferences prefs = getSharedPreferences(ACCOUNT\_PREFS\_NAME, 0);

String key = prefs.getString(ACCESS\_KEY\_NAME, null);

String secret = prefs.getString(ACCESS\_SECRET\_NAME, null);

if (key != null && secret != null) {

String[] ret = new String[2];

ret[0] = key;

ret[1] = secret;

return ret;

} else {

return null;

}

}

private void storeKeys(String key, String secret) {

// Save the access key for later

SharedPreferences prefs = getSharedPreferences(ACCOUNT\_PREFS\_NAME, 0);

Editor edit = prefs.edit();

edit.putString(ACCESS\_KEY\_NAME, key);

edit.putString(ACCESS\_SECRET\_NAME, secret);

edit.commit();

}

private void clearKeys() {

SharedPreferences prefs = getSharedPreferences(ACCOUNT\_PREFS\_NAME, 0);

Editor edit = prefs.edit();

edit.clear();

edit.commit();

}

private AndroidAuthSession buildSession() {

AppKeyPair appKeyPair = new AppKeyPair(APP\_KEY, APP\_SECRET);

AndroidAuthSession session;

String[] stored = getKeys();

if (stored != null) {

AccessTokenPair accessToken = new AccessTokenPair(stored[0],

stored[1]);

session = new AndroidAuthSession(appKeyPair, ACCESS\_TYPE,

accessToken);

} else {

session = new AndroidAuthSession(appKeyPair, ACCESS\_TYPE);

}

return session;

}

public void startListActivity() {

startActivity(new Intent());

}

void storeText() {

Date date = new Date();//9790515256

DateFormat txt = new SimpleDateFormat("yyyy-MM-dd-kk-mm-ss");

String newTxtFile = txt.format(date);

String outPath = newTxtFile + ".txt";

File sdCard = Environment.getExternalStorageDirectory();

File directory = new File(sdCard.getAbsolutePath());

directory.mkdirs();

File file = new File(directory, outPath);

try {

FileOutputStream fout = new FileOutputStream(file);

OutputStreamWriter osw = new OutputStreamWriter(fout);

osw.write(data+"\n\n"+"Updated from: "+addr);

osw.flush();

osw.close();

Toast.makeText(getApplicationContext(),

"File successfully created ", Toast.LENGTH\_SHORT).show();

UploadFile upload = new UploadFile(Home.this, mApi, PHOTO\_DIR, file);

upload.execute();

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// TODO Auto-generated method stub

MenuInflater inflat = getMenuInflater();

inflat.inflate(R.menu.menu1, menu);

return super.onCreateOptionsMenu(menu);

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// TODO Auto-generated method stub

String m = item.getTitle().toString();

int i = item.getItemId();

switch (i) {

case R.id.se:

Toast.makeText(Home.this, "Selected Menu: " + m, Toast.LENGTH\_LONG)

.show();

break;

case R.id.e:

Home.this.finish();

break;

}

return super.onOptionsItemSelected(item);

}

public class MyLocationListener implements LocationListener {

@Override

public void onLocationChanged(final Location location) {

new Thread(new Runnable() {

@Override

public void run() {

Geocoder geo = new Geocoder(getApplicationContext());

try {

final List<Address> add = geo.getFromLocation(location.getLatitude(), location.getLongitude(), 1);

runOnUiThread(new Runnable() {

@Override

public void run() {

addr = add.get(0).getAddressLine(0);

}

});

} catch (IOException e) {

e.printStackTrace();

}

}

}).start();

}

@Override

public void onProviderDisabled(String arg0) {

// TODO Auto-generated method stub

}

@Override

public void onProviderEnabled(String arg0) {

// TODO Auto-generated method stub

}

@Override

public void onStatusChanged(String arg0, int arg1, Bundle arg2) {

// TODO Auto-generated method stub

}

}

@Override

public void onDestroy() {

super.onDestroy();

locationManager.removeUpdates(locationListener);

}

}

**UploadFile.java**

package com.sweet.memories;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import android.app.ProgressDialog;

import android.content.Context;

import android.content.DialogInterface;

import android.content.DialogInterface.OnClickListener;

import android.os.AsyncTask;

import android.util.Log;

import android.widget.Toast;

import com.dropbox.client2.DropboxAPI;

import com.dropbox.client2.DropboxAPI.UploadRequest;

import com.dropbox.client2.ProgressListener;

import com.dropbox.client2.exception.DropboxException;

import com.dropbox.client2.exception.DropboxFileSizeException;

import com.dropbox.client2.exception.DropboxIOException;

import com.dropbox.client2.exception.DropboxParseException;

import com.dropbox.client2.exception.DropboxPartialFileException;

import com.dropbox.client2.exception.DropboxServerException;

import com.dropbox.client2.exception.DropboxUnlinkedException;

/\*\*

\* Here we show uploading a file in a background thread, trying to show

\* typical exception handling and flow of control for an app that uploads a

\* file from Dropbox.

\*/

public class UploadFile extends AsyncTask<Void, Long, Boolean> {

String TAG="UploadFile";

private DropboxAPI<?> mApi;

private String mPath;

private File mFile;

private long mFileLen;

private UploadRequest mRequest;

private Context mContext;

private final ProgressDialog mDialog;

private String mErrorMsg;

public UploadFile(Context context, DropboxAPI<?> api, String dropboxPath,

File file) {

// We set the context this way so we don't accidentally leak activities

mContext = context.getApplicationContext();

mFileLen = file.length();

mApi = api;

mPath = dropboxPath;

mFile = file;

mDialog = new ProgressDialog(context);

mDialog.setMax(100);

mDialog.setMessage("Uploading " + file.getName());

mDialog.setProgressStyle(ProgressDialog.STYLE\_HORIZONTAL);

mDialog.setProgress(0);

mDialog.setButton("Cancel", new OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

// This will cancel the putFile operation

mRequest.abort();

}

});

mDialog.show();

}

@Override

protected Boolean doInBackground(Void... params) {

try {

// By creating a request, we get a handle to the putFile operation,

// so we can cancel it later if we want to

FileInputStream fis = new FileInputStream(mFile);

String path = mPath + mFile.getName();

Log.w(TAG, "activity result video 4");

mRequest = mApi.putFileOverwriteRequest(path, fis, mFile.length(),

new ProgressListener() {

@Override

public long progressInterval() {

// Update the progress bar every half-second or so

return 500;

}

@Override

public void onProgress(long bytes, long total) {

publishProgress(bytes);

}

});

Log.w(TAG, "activity result video 5");

if (mRequest != null) {

Log.w(TAG, "activity result video 6");

mRequest.upload();

Log.w(TAG, "activity result video 7");

return true;

}

} catch (DropboxUnlinkedException e) {

// This session wasn't authenticated properly or user unlinked

mErrorMsg = "This app wasn't authenticated properly.";

} catch (DropboxFileSizeException e) {

// File size too big to upload via the API

mErrorMsg = "This file is too big to upload";

} catch (DropboxPartialFileException e) {

// We canceled the operation

mErrorMsg = "Upload canceled";

} catch (DropboxServerException e) {

// Server-side exception. These are examples of what could happen,

// but we don't do anything special with them here.

if (e.error == DropboxServerException.\_401\_UNAUTHORIZED) {

// Unauthorized, so we should unlink them. You may want to

// automatically log the user out in this case.

} else if (e.error == DropboxServerException.\_403\_FORBIDDEN) {

// Not allowed to access this

} else if (e.error == DropboxServerException.\_404\_NOT\_FOUND) {

// path not found (or if it was the thumbnail, can't be

// thumbnailed)

} else if (e.error == DropboxServerException.\_507\_INSUFFICIENT\_STORAGE) {

// user is over quota

} else {

// Something else

}

// This gets the Dropbox error, translated into the user's language

mErrorMsg = e.body.userError;

if (mErrorMsg == null) {

mErrorMsg = e.body.error;

}

} catch (DropboxIOException e) {

// Happens all the time, probably want to retry automatically.

mErrorMsg = "Network error. Try again.";

} catch (DropboxParseException e) {

// Probably due to Dropbox server restarting, should retry

mErrorMsg = "Dropbox error. Try again.";

} catch (DropboxException e) {

// Unknown error

mErrorMsg = "Unknown error. Try again.";

} catch (FileNotFoundException e) {

}

return false;

}

@Override

protected void onProgressUpdate(Long... progress) {

int percent = (int)(100.0\*(double)progress[0]/mFileLen + 0.5);

mDialog.setProgress(percent);

}

@Override

protected void onPostExecute(Boolean result) {

mDialog.dismiss();

boolean del = mFile.delete();

System.out.println(mFile.getAbsolutePath()+"Delete file" +del);

if (result) {

Log.w(TAG, "activity result video 8");

showToast("File successfully uploaded");

} else {

Log.w(TAG, "activity result video 9");

showToast(mErrorMsg);

}

}

private void showToast(String msg) {

Log.w(TAG, "activity result video 10");

Toast error = Toast.makeText(mContext, msg, Toast.LENGTH\_LONG);

error.show();

}

}

**Display.java**

package com.sweet.memories;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.FileReader;

import android.app.Activity;

import android.app.ProgressDialog;

import android.content.Intent;

import android.os.Bundle;

import android.os.Handler;

import android.os.Message;

import android.util.Log;

import android.widget.TextView;

import android.widget.Toast;

import com.dropbox.client2.exception.DropboxException;

public class DisplayText extends Activity {

ProgressDialog pd;

Intent intent;

private FileOutputStream mFos;

private String cachePath;

private final static String TEXT\_FILE\_NAME = "sweet.txt";

TextView textView;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

textView = new TextView(this);

setContentView(textView);

intent = getIntent();

pd = ProgressDialog.show(DisplayText.this, "", "Importing text...", false, true);

new Thread() {

public void run() {

cachePath = getCacheDir().getAbsolutePath() + "/" + TEXT\_FILE\_NAME;

try {

mFos = new FileOutputStream(cachePath);

} catch (FileNotFoundException e) {

Log.e("Get Image error:", ""+e.toString());

}

// This downloads a smaller, thumbnail version of the file. The

// API to download the actual file is roughly the same.

try {

ViewFileList.mApi.getFile(intent.getStringExtra("TextPath"), null, mFos, null);

} catch (DropboxException e) {

handler.sendEmptyMessage(0);

}

handler.sendEmptyMessage(1);

}

}.start();

}

private Handler handler = new Handler() {

public void handleMessage(Message msg) {

pd.dismiss();

if (msg.what == 1) {

File file = new File(getCacheDir().getAbsolutePath(),TEXT\_FILE\_NAME);

Log.d("text file", ""+file.getPath());

//Read text from file

StringBuilder text = new StringBuilder();

try {

BufferedReader br = new BufferedReader(new FileReader(file));

String line;

while ((line = br.readLine()) != null) {

text.append(line);

text.append('\n');

}

textView.setText(text.toString());

}

catch (Exception e) {

//You'll need to add proper error handling here

}

}

else if (msg.what == 0) {

Toast.makeText(DisplayText.this, "Error while retrieving image", Toast.LENGTH\_LONG).show();

DisplayText.this.finish();

}

}

};

}

**Displayimage.java**

package com.sweet.memories;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import android.app.Activity;

import android.app.ProgressDialog;

import android.content.Intent;

import android.graphics.drawable.Drawable;

import android.os.Bundle;

import android.os.Handler;

import android.os.Message;

import android.util.Log;

import android.widget.ImageView;

import android.widget.LinearLayout;

import android.widget.TextView;

import android.widget.Toast;

import com.dropbox.client2.DropboxAPI.ThumbFormat;

import com.dropbox.client2.DropboxAPI.ThumbSize;

import com.dropbox.client2.exception.DropboxException;

public class DisplayImage extends Activity {

ProgressDialog pd;

ImageView imageView;

Intent intent;

String cachePath;

private final static String IMAGE\_FILE\_NAME = "dbroulette.png";

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

LinearLayout linearLayout = new LinearLayout(this);

linearLayout.setOrientation(LinearLayout.VERTICAL);

imageView = new ImageView(this);

TextView textView = new TextView(this);

linearLayout.addView(textView);

linearLayout.addView(imageView);

setContentView(linearLayout);

intent = getIntent();

textView.setText(intent.getStringExtra("ImagePath"));

pd = ProgressDialog.show(DisplayImage.this, "", "Importing image...", false, true);

new Thread() {

private FileOutputStream mFos;

public void run() {

cachePath = getCacheDir().getAbsolutePath() + "/" + IMAGE\_FILE\_NAME;

try {

mFos = new FileOutputStream(cachePath);

} catch (FileNotFoundException e) {

Log.e("Get Image error:", ""+e.toString());

}

// This downloads a smaller, thumbnail version of the file. The

// API to download the actual file is roughly the same.

try {

ViewFileList.mApi.getThumbnail(intent.getStringExtra("ImagePath"), mFos, ThumbSize.BESTFIT\_960x640,

ThumbFormat.JPEG, null);

} catch (DropboxException e) {

handler.sendEmptyMessage(0);

}

handler.sendEmptyMessage(1);

}

}.start();

}

private Handler handler = new Handler() {

public void handleMessage(Message msg) {

pd.dismiss();

if (msg.what == 1) {

imageView.setImageDrawable(Drawable.createFromPath(cachePath));

}

else if (msg.what == 0) {

Toast.makeText(DisplayImage.this, "Error while retrieving image", Toast.LENGTH\_LONG).show();

DisplayImage.this.finish();

}

}

};}