Institiúid Teicneolaíochta Cheatharlach
INSTITUTE of TECHNOLOGY CARLOW
At the Heart of South Leinster

Student ID card Barcode Recognition for Android Mobile Phone

Code Listing

Student Name: Long Long

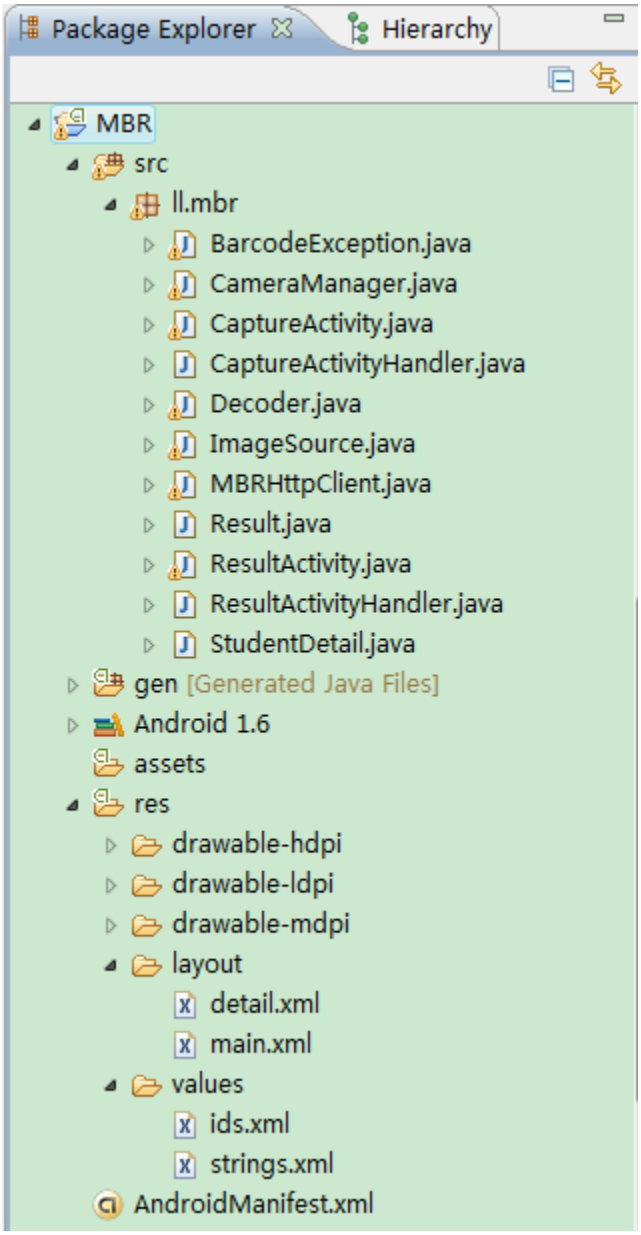
Student ID: C00131028

Supervisor: Christophe Meudec

Date: 16 Apr 2010

Contents

Client Hierarchy tree



1. SCR

1.1 BarcodeException.java

```
package ll.mbr;

public final class BarcodeException extends Exception {
    BarcodeException(){

    }

    public Throwable BackStackTrace(){
        return null;
    }
}
```

1.2 CameraManager.java

```
package ll.mbr;

import java.io.IOException;

import android.content.Context;
import android.graphics.Point;
import android.graphics.Rect;
import android.hardware.Camera;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.Display;
import android.view.SurfaceHolder;
import android.view.WindowManager;

public final class CameraManager {
    private final static String TAG = "CameraManager";


    private static CameraManager cameraManager;
    private Camera mCamera;
    private final Context context;
```

```
private Point screenResolution;
private Point cameraResolution;
private Handler previewHandler;
private int previewMessage;
private Handler autoFocusHandler;
private int autoFocusMessage;
private boolean initialized;
private boolean previewing;
private int previewFormat;
private String previewFormatString;
private boolean useOneShotPreviewCallback;

public static void init(Context context) {
    if (cameraManager == null) {
        cameraManager = new CameraManager(context);
    }
}

public static CameraManager get() {
    return cameraManager;
}

private CameraManager(Context context) {
    this.context = context;
    mCamera = null;
    initialized = false;
    previewing = false;
    useOneShotPreviewCallback = true;
}

private final Camera.PreviewCallback previewCallback = new Camera.PreviewCallback() {
    public void onPreviewFrame(byte[] data, Camera camera) {
        if (!useOneShotPreviewCallback) {
            camera.setPreviewCallback(null);
        }
        if (previewHandler != null) {
            Message message = previewHandler.obtainMessage(previewMessage,
                    cameraResolution.x, cameraResolution.y, data);
            message.sendToTarget();
            previewHandler = null;
        }
    }
};
```

```
    private final Camera.AutoFocusCallback autoFocusCallback = new
Camera.AutoFocusCallback() {
        public void onAutoFocus(boolean success, Camera camera) {
            if (autoFocusHandler != null) {
                Message message = autoFocusHandler.obtainMessage(
                        autoFocusMessage, success);
                // Simulate continuous autofocus by sending a focus request every 1.0
seconds.
                autoFocusHandler.sendMessageDelayed(message, 1000L);
                autoFocusHandler = null;
            }
        }
    };

    public Camera openDriver(SurfaceHolder holder) throws IOException {
        if (mCamera == null) {
            mCamera = Camera.open();
            mCamera.setPreviewDisplay(holder);

            if (!initialized) {
                initialized = true;
                getScreenResolution();
            }

            setCameraParameters();
        }
        return mCamera;
    }

    public void closeDriver() {
        if (mCamera != null) {
            mCamera.release();
            mCamera = null;
        }
    }

    public void startPreview() {
        if (mCamera != null && !previewing) {
            mCamera.startPreview();
            previewing = true;
        }
    }

    public void stopPreview() {
```

```
        if (mCamera != null && previewing) {
            if (!useOneShotPreviewCallback) {
                mCamera.setPreviewCallback(null);
            }
            mCamera.stopPreview();
            previewHandler = null;
            autoFocusHandler = null;
            previewing = false;
        }
    }

    public void requestPreview(Handler handler, int message) {
        if (mCamera != null && previewing) {
            previewHandler = handler;
            previewMessage = message;

            mCamera.setOneShotPreviewCallback(previewCallback);

        }
    }

    public void requestAutoFocus(Handler handler, int message) {
        if (mCamera != null && previewing) {
            autoFocusHandler = handler;
            autoFocusMessage = message;
            mCamera.autoFocus(autoFocusCallback);
        }
    }

    private void setCameraParameters() {
        Camera.Parameters parameters = mCamera.getParameters();
        Camera.Size size = parameters.getPreviewSize();
        Log.v(TAG, "Default preview size: " + size.width + ", " + size.height);
        previewFormat = parameters.getPreviewFormat();
        previewFormatString = parameters.get("preview-format");
        Log.v(TAG, "Default preview format: " + previewFormat);

        // make sure that the camera resolution is a multiple of 8, as the screen
        // may not be.
        cameraResolution = new Point();
        cameraResolution.x = screenResolution.x;
        cameraResolution.y = screenResolution.y;
        Log.v(TAG, "Setting preview size: " + cameraResolution.x + ", "
                + cameraResolution.y);
```

```
            parameters.setPreviewSize(cameraResolution.x, cameraResolution.y);

            parameters.set("flash-mode", "off");

            mCamera.setParameters(parameters);
        }

        private Point getScreenResolution() {
            if (screenResolution == null) {
                WindowManager manager = (WindowManager) context
                        .getSystemService(Context.WINDOW_SERVICE);
                Display display = manager.getDefaultDisplay();
                screenResolution = new Point(display.getWidth(), display
                        .getHeight());
            }
            return screenResolution;
        }

}
```

1.3 CaptureActivity.java

```
package ll.mbr;

import java.io.IOException;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.ColorMatrix;
import android.graphics.ColorMatrixColorFilter;
import android.graphics.PixelFormat;
import android.graphics.Point;
import android.graphics.Rect;
import android.hardware.Camera;
import android.hardware.Camera.Parameters;
import android.hardware.Camera.PictureCallback;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
```

```
import android.util.Log;
import android.view.Display;
import android.view.KeyEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.view.View.OnClickListener;
import android.view.View.OnKeyListener;
import android.widget.ImageView;
import android.widget.TextView;

public class CaptureActivity extends Activity implements SurfaceHolder.Callback{
    private final static String TAG = "CameraActivity";

    private CaptureActivityHandler handler;

    private SurfaceView mSurfaceView;
    private ImageView mImageView;
    private SurfaceHolder mSurfaceHolder;
    private Camera mCamera;

    private boolean surfaceRunning;
    private Context context;

    private boolean showingResult = false;


    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        Log.v(TAG,"onCreate called");
        super.onCreate(savedInstanceState);

        context = getApplication();
        CameraManager.init(context);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFormat(PixelFormat.TRANSLUCENT);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
        setContentView(R.layout.main);
        mSurfaceView = (SurfaceView) findViewById(R.id.camera);
```

```
        mImageView = (ImageView) findViewById(R.id.image);
        mImageView.setVisibility(View.GONE);

        mSurfaceHolder = mSurfaceView.getHolder();
        mSurfaceHolder.addCallback(this);
        mSurfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

    }

    @Override
    protected void onStart(){
        Log.v(TAG,"onStart called");
        super.onStart();
    }

    @Override
    protected void onResume() {
        Log.v(TAG,"onResume called");
        super.onResume();
        Log.v(TAG,"surfaceRunning:"+surfaceRunning);
        Log.v(TAG,"mSurfaceHolder:"+mSurfaceHolder);
        Log.v(TAG,"mCamera:"+mCamera);
        if(surfaceRunning){
            initCamera(mSurfaceHolder);
        }
        SurfaceView mSurfaceView = (SurfaceView) findViewById(R.id.camera);
        SurfaceHolder mSurfaceHolder = mSurfaceView.getHolder();
        mSurfaceHolder.addCallback(this);
        mSurfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }

    @Override
    protected void onPause(){
        Log.v(TAG,"onPause called");
        super.onPause();

    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width,
            int height) {
        // TODO Auto-generated method stub
        Log.v(TAG,"surfaceChanged called");
        if(!surfaceRunning){
```

```
                surfaceRunning = true;
                initCamera(holder);
            }
        }

        @Override
        public void surfaceCreated(SurfaceHolder holder) {
            // TODO Auto-generated method stub
            Log.v(TAG,"surfaceCreated called");
            if(!surfaceRunning){
                surfaceRunning = true;
                initCamera(holder);
            }
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            // TODO Auto-generated method stub
            Log.v(TAG,"surfaceDestroyed called");
//          mCamera.stopPreview();
            surfaceRunning = false;
            mCamera.release();
        }

//      @Override
//      public boolean onKeyDown(int keyCode, KeyEvent event) {
//          if (keyCode == KeyEvent.KEYCODE_BACK) {
////                if(showingResult){
////                    showingResult = false;
////
        Log.v(TAG,"surfaceRunning:"+surfaceRunning+",mPreviewRunning:"+mPreviewRunning);
////                    setContentView(R.layout.main);
////                    if(surfaceRunning){
////                        initCamera(mSurfaceHolder);
////                    }
////                    SurfaceView mSurfaceView = (SurfaceView) findViewById(R.id.camera);
////                    SurfaceHolder mSurfaceHolder = mSurfaceView.getHolder();
////                    mSurfaceHolder.addCallback(this);
////                    mSurfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
////                    CameraManager.get().startPreview();
////                    handler.sendEmptyMessage(R.id.restart);
////                    return true;
////                }else{
//                    setResult(RESULT_CANCELED);
```

```
//                      finish();
//                      return true;
////              }
//          } else
//                  return false;
//    }


    private void initCamera(SurfaceHolder holder) {
        try{
            mCamera = CameraManager.get().openDriver(holder);
        }catch(IOException ioe){
            Log.w(TAG,ioe);
            return;
        }
        if(handler==null){
            boolean beginScanning = true;
            handler = new CaptureActivityHandler(this,beginScanning);
        }
    }

    public Handler getHandler() {
        // TODO Auto-generated method stub
        return handler;
    }

    public void showResult(Result result){

        Intent intent = new Intent(CaptureActivity.this,ResultActivity.class);
        intent.putExtra("studentId", result.getText());
        startActivity(intent);

        showingResult = true;
    }

}
```

1.4 CaptureActivityHandler.java

```
package ll.mbr;

import android.os.Handler;
import android.os.Message;
```

```
import android.util.Log;

public class CaptureActivityHandler extends Handler {
    private final static String TAG = "CameraActivityHandler";

    private final CaptureActivity activity;
    private final Decoder decoder;

    CaptureActivityHandler(CaptureActivity activity, boolean beginScanning) {
        this.activity = activity;
        decoder = new Decoder(activity);
        decoder.start();

        CameraManager.get().startPreview();
        if (beginScanning) {
            restartPreview();
        }
    }

    @Override
    public void handleMessage(Message message) {
        switch (message.what){
            case R.id.auto_focus:
//                  Log.v(TAG, "requestAuto_focus");
                    CameraManager.get().requestAutoFocus(this, R.id.auto_focus);
                    break;
            case R.id.decode:
                    //here is the place requesting previewCallback and then send the message to
Decoder
                    Log.v(TAG, "requestDecode");
                    CameraManager.get().requestPreview(decoder.getHandler(),    R.id.decode);

                    break;
            case R.id.decode_succeeded:
                    Log.v(TAG,"decode succeeded");
                    activity.showResult((Result) message.obj);
            case R.id.restart_preview:
                    Log.v(TAG,"restart preview");
                    restartPreview();
                    break;
        }

    }
```

```
    public void restartPreview(){
        CameraManager.get().requestPreview(decoder.getHandler(), R.id.decode);
        CameraManager.get().requestAutoFocus(this,R.id.auto_focus);
    }
}
```

1.5 Decoder.java

```
package ll.mbr;

import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;

import android.graphics.Bitmap;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.Looper;
import android.os.Message;
import android.util.Log;

public class Decoder extends Thread{
    private static final String TAG="Decoder";
    public static final String BARCODE_BITMAP = "barcode_bitmap";

    private static final int BLACK = 0;
    private static final int WHITE = 255;
    private static final String ALPHABET = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ-.
*$/+%";
    private static final int[] ALPHABET_ENCODINGS = {
        0x034, 0x121, 0x061, 0x160, 0x031, 0x130, 0x070, 0x025, 0x124, 0x064, // 0123456789
        0x109, 0x049, 0x148, 0x019, 0x118, 0x058, 0x00D, 0x10C, 0x04C, 0x01C, // ABCDEFGHIJ
        0x103, 0x043, 0x142, 0x013, 0x112, 0x052, 0x007, 0x106, 0x046, 0x016, //
KLMNOPQRST
        0x181, 0x0C1, 0x1C0, 0x091, 0x190, 0x0D0, 0x085, 0x184, 0x0C4, 0x094, // UVWXYZ-. *
        0x0A8, 0x0A2, 0x08A, 0x02A // $/+%
    };
    private static final int ASTERISK_ENCODING = 0x094;    //0x02A

    static int n = 0;
    static int blackValue;
```

```
    private Handler handler;
    private final CaptureActivity activity;

    Decoder(CaptureActivity activity){
        this.activity=activity;
    }

    public Handler getHandler(){
        return handler;
    }

    @Override
    public void run(){
        Looper.prepare();
        handler = new Handler(){
            @Override
            public void handleMessage(Message message){
                switch(message.what){
                case R.id.decode:
                    if(message.obj!=null) Log.v(TAG,"preview data recieved");
//                  if(decode((byte[])message.obj,message.arg1,message.arg2)){
//                      Log.v(TAG,"decode succeed");
//                      CameraManager.get().stopPreview();
//                  }else{
//                      CameraManager.get().requestPreviewFrame(handler, R.id.decode);
//                  }
                    decode((byte[])message.obj,message.arg1,message.arg2);
                    break;
                case R.id.quit:
                    Looper.myLooper().quit();
                    break;
                }
            }
        };
        Looper.loop();
    }

    protected void decode(byte[] data, int width, int height) {
        // TODO Auto-generated method stub
        long start = System.currentTimeMillis();
        boolean succeed;
        Result result = null;
        ImageSource source = new ImageSource(data,width,height);
```

```
            try{
                result = doDecode(source);
                succeed = true;
            } catch (BarcodeException be) {
                succeed = false;
            }
            long end = System.currentTimeMillis();

//          return succeed;

            if (succeed) {
                Log.v(TAG,"decode succeed");
                Log.v(TAG, "Found barcode (" + (end - start) + " ms):\n" + result.toString());
                CameraManager.get().stopPreview();
                Message message = Message.obtain(activity.getHandler(),
R.id.decode_succeeded, result);
//              Bundle bundle = new Bundle();
//              bundle.putParcelable(BARCODE_BITMAP, source.greyScale());
//              message.setData(bundle);
                message.sendToTarget();
            } else {
                Message message = Message.obtain(activity.getHandler(), R.id.decode);
                message.sendToTarget();
            }
        }

        public Result doDecode(ImageSource source) throws BarcodeException{
            int w = source.getWidth();
            int h = source.getHeight();
            byte[] data = source.getData();
            int[] row = new int[w];
            int middle = h/2;    // the most middle row
            int jump = 30;    // jump 30 rows each time
            int moveStep = 0;    // number of times to move over
            boolean isBelow = false;    // use to determine if is looking below

            for(int i=0;i<h;i++){
                // look at the middle row first, then the rows above, and then the rows below
                int rowNum = middle + jump * (isBelow? moveStep: -moveStep);
                // if reach the top, move to the part below
                if(rowNum < 0){
                    moveStep = 1;
                    isBelow = true;
                    rowNum = middle + jump * (isBelow? moveStep: -moveStep);
```

```
                }
                moveStep++;

                // if run out of the image boundary
                if(rowNum<0 || rowNum >= h){
                        break;
                }
                n=0;

                try{
                        row = source.doThresholding(rowNum,row,w,data);
                } catch(BarcodeException be){
                        continue;
                }

                for(int doReverse=0;doReverse <= 1;doReverse++){
                        // determine if will reverse or not
                        if(doReverse == 1){
                                row = reverse(row);
                        }
                        try{
                                // decode row by row to get a barcode
                                // if can't decode this row, the function decodeRow will throw a
BarcodeException
                                String text = decodeRow(row);
                                // return null if cannot decode
                                Result result = new Result(text,data,rowNum);
                                Log.v(TAG,"result:"+result.getText());
                                return result;
                        } catch(BarcodeException be){
                        // do nothing if cannot decode this row, and continue
                        }
                }
        }

//          Log.v(TAG,"doDecode exception, will throw BarcodeException");
            throw new BarcodeException();
    }

    public int[] reverse(int[] row){
            int temp;
            for(int i=0;i<row.length/2;i++){
                    temp = row[i];
                    row[i] = row[row.length-1-i];
```

```
            row[row.length-1-i] = temp;
        }
        return row;
    }

    public String decodeRow(int[] row) throws BarcodeException{
        Log.v(TAG,"decoding row");
        int[] startAsterisk = findAsterisk(row,0);

        if(startAsterisk.length==0){
            Log.v(TAG,"didn't find asterisk. will throw BarcodeException");
            throw new BarcodeException();
        }

        int next = startAsterisk[1] + 1;
        StringBuffer text = new StringBuffer();

        while(next < row.length && row[next]!=BLACK){
            next++;
        }

        int[] narrowWide = new int[9];
        char charecter;
        int last;

        /*
         *
         * other characters decoding
         *
         */
        do{
            narrowWide = recordNarrowWide(row,next);
            int pattern = patternToInt(convertToPattern(narrowWide));
            Log.v(TAG,"find alphabet encoding:"+pattern);
            charecter = intToChar(pattern);
            Log.v(TAG,"find alphabet:"+charecter);
            if(charecter != '*'){
                text.append(charecter);
            }
            last = next;

            for(int i=0;i < narrowWide.length;i++){
                next += narrowWide[i];
            }
```

```
                while(next < row.length && row[next]!=BLACK){
                    next++;
                }
            }while(charecter != '*');

            return text.toString();
        }

        public char intToChar(int pattern) throws BarcodeException{
            for(int i=0;i < ALPHABET_ENCODINGS.length;i++){
                if(pattern == ALPHABET_ENCODINGS[i]){
//                  Log.v(TAG,"find alphabet:"+ALPHABET.charAt(i));
                    return ALPHABET.charAt(i);
                }
            }
            Log.v(TAG,"patternToChar exception, will throw BarcodeException");
            throw new BarcodeException();
        }

        public static int[] recordNarrowWide(int[] row,int offset) throws BarcodeException{

            if(offset >= row.length){
                Log.v(TAG,"out of row.length, will throw BarcodeException");
                throw new BarcodeException();
            }

            int[] narrowWide = new int[9];

            int isWhiteOrBlack = 255 - row[offset]; // implies the colour of next pixel. 0 means is
black, 255 means is white
            int index = 0;
            int i = offset;
            for(;i < row.length;i++){
                int pixel = row[i];
                if(pixel!=isWhiteOrBlack){
                    narrowWide[index]++;
                }
                else{
                    index++;
                    if(index == narrowWide.length){
                        break;
                    } else{
                        narrowWide[index] = 1;
```

```
                        if(isWhiteOrBlack==0){
                            isWhiteOrBlack=255;
                        } else isWhiteOrBlack=0;
                    }
                }
            }
            // If we read fully the last section of pixels and filled up our counters -- or filled
            // the last counter but ran off the side of the image, OK. Otherwise, a problem.
            if(!(index == narrowWide.length || (index == narrowWide.length - 1 && i == row.length))){
                Log.v(TAG,"didn't read fully, will throw BarcodeException");
                throw new BarcodeException();
            }
            StringBuffer sb = new StringBuffer();
            for(int j=0;j < narrowWide.length;j++){
                sb.append(narrowWide[j]);
            }
            Log.v(TAG,"find pattern:"+sb+":"+offset+"-"+i);
            return narrowWide;
        }

        public static int[] findAsterisk(int[] row,int offset) throws BarcodeException{
            int w = row.length;
            int barcodeStartAt = offset;

            // find the start index of pixel of barcode hopefully,
            // if not, we can still find a way to get it,
            // by attempting to decode it and throw it away if get meaningless value
            // and move to the next pixel
            // but more expensive in that way.
            for(;barcodeStartAt < w;barcodeStartAt++){
                if(row[barcodeStartAt]==BLACK){
                    break;
                }
            }

            int index = 0;
            int[] narrowWide = new int[9];
            int asteriskStartAt = barcodeStartAt;
            int isWhiteOrBlack = 255; // implies the colour of next pixel. 0 means is black, 255 means is white
            int len = narrowWide.length;

            for(int i=barcodeStartAt;i < w;i++){
```

```
                int pixel = row[i];
                // if the current pixel has the same colour as the previous one has
                // then they are a part of the same bar
                if(pixel!=isWhiteOrBlack){
                        narrowWide[index]++;
                }
                // otherwise
                else{
                        // if reach the length of the narrowWide (9)
                        if(index == len - 1){
                                try{
                                        // and if found asterisk
                                        if(patternToInt(convertToPattern(narrowWide))==
ASTERISK_ENCODING){
                                                Log.v(TAG,"asterisk       found,position:"+asteriskStartAt+"       -
"+i+":"+n);
                                                n++;
//                                              CameraManager.get().stopPreview();
                                                return new int[]{asteriskStartAt,i};
                                        }
                                }catch(BarcodeException be){
                                        // continue
                                }
                                // if didn't get asterisk, move the array forward for 2 units
                                asteriskStartAt += narrowWide[0] + narrowWide[1];
                                for(int j=2;j < len;j++){
                                        narrowWide[j-2] = narrowWide[j];
                                }
                                // and set the last two 0, so we can start a new attempt
                                narrowWide[len-2] = 0;
                                narrowWide[len-1] = 0;
                                index--;
                        }else{
                                index++;
                        }
                        // the current pixel has a different colour than the previous one has
                        // start counting the width of a new bar(the next bar)
                        narrowWide[index] = 1;
                        // set the colour of the next pixel to the opposite
                        if(isWhiteOrBlack==0){
                                isWhiteOrBlack=255;
                        } else isWhiteOrBlack=0;
                }
        }
```

```
            // if cannot find the asterisk when reaching the end of the row, throw exception and
attempt a new row then
            Log.v(TAG,"find asterisk exception, will throw BarcodeException");
            throw new BarcodeException();
        }


        public static int[] convertToPattern(int[] narrowWide) throws BarcodeException{
            int len = narrowWide.length;
            int max = 0;    // max width of narrow bars
            int wideNum = 0;    // number of wide bars

            do{
                // looking for a suitable value for max value of narrow bars
                int x = Integer.MAX_VALUE;
                for(int i=0;i < len;i++){
                    if(narrowWide[i] < x && narrowWide[i] > max){
                        x = narrowWide[i];
                    }
                }
                max = x;
                wideNum = 0;
                int[] pattern = new int[9];
                for(int i=0;i < len;i++){
                    // set it 1 if > max, otherwise leave it(it was initialized 0)
                    // 1 means wide bar, while 0 means narrow bar
                    if(narrowWide[i] > max){
                        pattern[i] = 1;
                        wideNum++;
                    }
                }

                // if found 3 wide bars
                if(wideNum == 3){
                    // if the max width of narrow bars is too big, treat it an error and throw
exception
                    if(max == Integer.MAX_VALUE){
                        Log.v(TAG,"convertToNarrowWide exception, will throw
BarcodeException");
                        throw new BarcodeException();
                    }
                    // otherwise, return the found pattern
                    return pattern;
                }
```

```
            // if wideNum > 3, re-do it with making max a bit bigger
            }while(wideNum  > 3);

            // otherwise, this piece of narrowWide is meaningless because it doesn't have 3 wide
bars, and throw exception
            Log.v(TAG,"convertToNarrowWide exception, will throw BarcodeException");
            throw new BarcodeException();
        }

        public static int patternToInt(int[] pattern){
            int weight = pattern.length-1;
            int x = 0;
            for(int i=0;i < pattern.length;i++){
                // convert array expression to int expression, for easier in comparing
                // e.g. pattern:{0,1,0,1,0,1,0,0,0}-> x:=010101000 (binary)= 168 (decimal)= 0x0A8;
                x += (pattern[i] << weight);
                weight--;
            }
//          Log.v(TAG,"find alphabet encoding:"+x);
            return x;
        }

}
```

1.6 ImageSource.java

```
package ll.mbr;

import android.graphics.Bitmap;

public final class ImageSource{
    private byte[] yuvData;
    private int dataWidth;
    private int dataHeight;
    private int left;
    private int top;

    public ImageSource(byte[] yuvData, int dataWidth, int dataHeight, int
left,
            int top, int width, int height) {
        this.yuvData = yuvData;
        this.dataWidth = dataWidth;
        this.dataHeight = dataHeight;
```

```
        this.left = left;
        this.top = top;
    }

    public ImageSource(byte[] yuvData,int dataWidth, int dataHeight){
        this.yuvData = yuvData;
        this.dataWidth = dataWidth;
        this.dataHeight = dataHeight;
        this.left = 0;    //no use in this case
        this.top = 0; //no use in this case
    }

    public int getWidth(){
        return dataWidth;
    }

    public int getHeight(){
        return dataHeight;
    }

    public ImageSource getSource(){
        return this;
    }

    public byte[] getData(){
        return yuvData;
    }

    public byte[] getRow(int rowNum){
        byte[] row = new byte[dataWidth];
        int srcPos = rowNum * dataWidth;
        System.arraycopy(yuvData, srcPos, row, 0, dataWidth);
        return row;
    }

//  public Bitmap greyScale(){
//      return null;
//  }

    // grey scaling and thresholding are combined in this function
    public int[] doThresholding(int rowNum, int[] row, int width, byte[]
data) throws BarcodeException{
        int w = width;
        int srcPos = rowNum * w;
```

```
        byte[] imgData = new byte[w];
        // Copies the number of length elements of the Array data starting
at the offset srcPos into the Array row at the position 0.
        System.arraycopy(data, srcPos, imgData, 0, w);

        int[] colourBaskets = new int[256/8];  // divide 256 colours into
32 colour baskets, 32 = 256/8
        // make sure colourBaskets is initialized
        for(int i=0;i < colourBaskets.length;i++){
            colourBaskets[i] = 0;
        }

        // the first 2/3 of imgData are grey values
        for(int i=0;i<(w*2/3);i++){
            int pixel = imgData[i] & 0xff; // convert byte to decimal(int),
0xff = 11111111, java doesn't have unsigned type, so need convert to it
by hand
            colourBaskets[pixel >> 3]++;  // put the pixel into the
relative colour basket
        }
        // calculate the threshold value
        int thresholdValue = calcThresholdValue(colourBaskets);

        // will use a filter to optimize the colour of pixels
        int leftPixel = imgData[0] & 0xff;
        int middlePixel = imgData[1] & 0xff;
        for(int i=0;i<(w-1);i++){
            int rightPixel = imgData[i+1] & 0xff;
//          int pixel = ((middlePixel*4) + leftPixel + rightPixel) / 6;
            int pixel = (middlePixel*4 - leftPixel - rightPixel) / 2;
            // if colour of the pixel < criticalBlack, set it black
            if(pixel < thresholdValue){
                row[i] = setBlack(i,row);
            }
            // otherwise, set it white
            else{
                row[i] = setWhite(i,row);
            }
            leftPixel = middlePixel;
            middlePixel = rightPixel;
        }
        return row;
    }
```

```
    public int setBlack(int i,int[] row){
        return row[i]=0;
    }

    public int setWhite(int i,int[] row){
        return row[i]=255;
    }

    public int calcThresholdValue(int[] colourBuckets){
        // Find the tallest peak in the colourBuckets
        int numOfBuckets = colourBuckets.length;
        int max = 0; // max bucket count
        int firstPeakIndex = 0;
        int firstPeakSize = 0;
        for(int i=0;i < numOfBuckets;i++){
            if(colourBuckets[i] > firstPeakSize){
                firstPeakIndex = i;
                firstPeakSize = colourBuckets[i];
            }
            if(colourBuckets[i] > max){
                max = colourBuckets[i];
            }
        }

        // Find the second tallest peak
        int secondPeakIndex = 0;
        int secondPeakSize = 0;
        int secondPeakScore = 0;
        for(int i=0;i < numOfBuckets;i++){
            int distance = i - firstPeakIndex;
            int score = colourBuckets[i] * distance * distance;
            if(score > secondPeakScore){
                secondPeakIndex = i;
                secondPeakScore = score;
            }
        }

        // Guarantee firstPeak relates to the black peak
        // since colour.black(0 in int) and colour.white(255 in int),
        // black peak always comes first, then white usually comes the last
one
        if(firstPeakIndex > secondPeakIndex){
            int temp = secondPeakIndex;
            secondPeakIndex = firstPeakIndex;
```

```
            firstPeakIndex = temp;
        }

        // Find a suitable threshold value
        int thresholdValue = secondPeakIndex - 1;
        int criticalScore = -1;
        for(int i=secondPeakIndex -1;i > firstPeakIndex;i--){
            int fromFirst = i - firstPeakIndex;
            // to find a index that is closer to the white peak
            // and makes the value of colourBuckets[index] small
            int score = fromFirst * fromFirst * (secondPeakIndex - i) *
(max - colourBuckets[i]);
            if(score > criticalScore){
                thresholdValue = i;
                criticalScore = score;
            }
        }

        // as we reduce the range by 1/8 times before(in the case of
colourBaskets), now we revert it to the original range
        return thresholdValue * 8;
//      return 10 * 8;
    }

}
```

1.7 MBRHttpClient

```
package ll.mbr;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map.Entry;

import org.apache.http.HttpResponse;
```

```
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import android.util.Log;

public class MBRHttpClient {
	private static final String TAG = "HttpClient";

	public static StudentDetail doRequest(String url,
			HashMap<String, ? extends Object> paramMap) {
		Log.v(TAG, "deRequest");

		HttpClient httpClient = new DefaultHttpClient();
		HttpPost httpPost = new HttpPost(url);
		List<NameValuePair> param = new ArrayList<NameValuePair>();
		Iterator iterator = paramMap.entrySet().iterator();
		while (iterator.hasNext()) {
			Entry entry = (Entry) iterator.next();
			param.add(new BasicNameValuePair((String) entry.getKey(),(String)
entry.getValue()));
		}

		String line = null;
		try {
			httpPost.setEntity(new UrlEncodedFormEntity(param));
			HttpResponse httpResponse = httpClient.execute(httpPost);
			line = readResponse(httpResponse);

		} catch (UnsupportedEncodingException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (ClientProtocolException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			Log.e(TAG,"Could not establish a HTTP connection to the server or could not get a
```

```
response properly from the server.",e);
                e.printStackTrace();
            }

            StudentDetail sd = new StudentDetail("fail");
            try {
                JSONObject jsonObject = new JSONObject(line);
                String loginState = jsonObject.getString("loginState");
                /**
                 * other jsonObject read work will be appended here
                 */

                if (loginState.equals("fail")) {
                    sd.setLoginState(loginState);
//                  return sd;
                } else {
                    sd.setLoginState("valid");
//                  sd.setIsFound(jsonObject.getString("notFound"));
                    sd.setId(jsonObject.getString("studentId"));
                    sd.setFirstname(jsonObject.getString("firstname"));
                    sd.setSurname(jsonObject.getString("surname"));
                    sd.setGender(jsonObject.getString("gender"));
                    sd.setDateOfBirth(jsonObject.getString("dateOfBirth"));
                    sd.setCourse(jsonObject.getString("course"));
                    sd.setCourseCode(jsonObject.getString("courseCode"));
//                  return sd;
                }

            } catch (JSONException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            return sd;
            // return line.trim();
            // return new StudentDetail();
        }

        public static String readResponse(HttpResponse httpResponse) {
            String line = null;

            try {
                InputStream content = httpResponse.getEntity().getContent();
                BufferedReader in = new BufferedReader(new InputStreamReader(content));
```

```
                line = in.readLine();
                in.close();

            } catch (IllegalStateException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            return line.trim();

        }
}
```

1.8 Result.java

```
package ll.mbr;

public final class Result {
    private final String text;
    private final byte[] rawData;
    private final int rowNum;

    public Result(String text,byte[] rawData,int rowNum){
        this.text=text;
        this.rawData=rawData;
        this.rowNum=rowNum;
    }

    public String getText(){
        return text;
    }

    public byte[] getData(){
        return rawData;
    }

    public int getRowNum(){
        return rowNum;
    }
```

```
    public byte[] getRow(){
        byte[] row = new byte[rawData.length];
        int srcPos = getRowNum() * rawData.length;
        System.arraycopy(rawData, srcPos, row, 0, rawData.length);
        return row;
    }

    public Result getResult(){
        return this;
    }
}
```

1.9 ResultActivity.java

```
package ll.mbr;

import java.util.HashMap;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.pm.ActivityInfo;
import android.graphics.PixelFormat;
import android.graphics.Point;
import android.hardware.Camera;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;

public class ResultActivity extends Activity implements SurfaceHolder.Callback{
    private static final String TAG = "ResultActivity";
```

```
    private static final String SERVER_URL = "http://192.168.1.11:8080/StudentServer/";

    private Bundle extras;
    private String studentId;
    private Context context;
    private ProgressDialog progressDialog;
    private MBRHttpClient httpClient;
    private StudentDetail sd;

    private Handler handler;

    // UI components
    private TextView barcode_TextView;
    private TextView username_Tag;
    private EditText userName_EditText;
    private TextView password_Tag;
    private EditText password_EditText;
    private Button getDetail_Button;

    private TextView notFound_TextView;

    // UI components: student details
    private TextView firstname_Tag;
    private TextView firstname_TextView;
    private TextView surname_Tag;
    private TextView surname_TextView;
    private TextView gender_Tag;
    private TextView gender_TextView;
    private TextView dateOfBirth_Tag;
    private TextView dateOfBirth_TextView;
    private TextView course_Tag;
    private TextView course_TextView;
    private TextView courseCode_Tag;
    private TextView courseCode_TextView;

//  private boolean synLock = false;


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        context = getApplication();
```

```
        handler = new ResultActivityHandler(this);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFormat(PixelFormat.TRANSLUCENT);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
        setContentView(R.layout.detail);

        barcode_TextView = (TextView) findViewById(R.id.barcode_TextView);
        username_Tag = (TextView) findViewById(R.id.username_Tag);
        userName_EditText = (EditText) findViewById(R.id.userName_EditText);
        password_Tag = (TextView) findViewById(R.id.password_Tag);
        password_EditText = (EditText) findViewById(R.id.password_EditText);

        notFound_TextView = (TextView) findViewById(R.id.notFound_TextView);

        // student details
        firstname_Tag =      (TextView) findViewById(R.id.firstname_Tag);
        firstname_TextView = (TextView) findViewById(R.id.firstname_TextView);
        surname_Tag = (TextView) findViewById(R.id.surname_Tag);
        surname_TextView = (TextView) findViewById(R.id.surname_TextView);
        gender_Tag = (TextView) findViewById(R.id.gender_Tag);
        gender_TextView = (TextView) findViewById(R.id.gender_TextView);
        dateOfBirth_Tag = (TextView) findViewById(R.id.dateOfBirth_Tag);
        dateOfBirth_TextView = (TextView) findViewById(R.id.dateOfBirth_TextView);
        course_Tag = (TextView) findViewById(R.id.course_Tag);
        course_TextView = (TextView) findViewById(R.id.course_TextView);
        courseCode_Tag = (TextView) findViewById(R.id.courseCode_Tag);
        courseCode_TextView = (TextView) findViewById(R.id.courseCode_TextView);

        // set details invisible when start
//      firstname_Tag.setVisibility(View.GONE);
//      firstname_TextView.setVisibility(View.GONE);
//      surname_Tag.setVisibility(View.GONE);
//      surname_TextView.setVisibility(View.GONE);
//      gender_Tag.setVisibility(View.GONE);
//      gender_TextView.setVisibility(View.GONE);
//      course_Tag.setVisibility(View.GONE);
//      course_TextView.setVisibility(View.GONE);
//      courseCode_Tag.setVisibility(View.GONE);
//      courseCode_TextView.setVisibility(View.GONE);

        extras = getIntent().getExtras();
        studentId = extras.getString("studentId");
```

```
            barcode_TextView.setText(studentId);
            getDetail_Button = (Button) findViewById(R.id.getDetail_Button);
            getDetail_Button.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    // TODO Auto-generated method stub
                    String userName = userName_EditText.getText().toString().trim();
                    String password = password_EditText.getText().toString().trim();
                    Log.v(TAG,"userName:"+userName+",password:"+password);
                    if(!userName.equals("") && !password.equals("")){
                        doLogin(userName,password);
                        Log.v(TAG,"doLogin really finish");
                    }else{
                        new AlertDialog.Builder(ResultActivity.this).setMessage("Please ensure
user name and password are not empty.").setPositiveButton("OK", null).show();
                    }
//                  if(synLock==true){
//                      if(sd.getLoginState().equals("false")){
//                          new AlertDialog.Builder(ResultActivity.this).setMessage("Login
failed, Incorrect user name or password.").setPositiveButton("OK", null).show();
//                      }else{
//                          setDetail(sd);
//                      }
//                  }
                }
            });
        }

        @Override
        protected void onResume() {
            super.onResume();
            Log.v(TAG,"onResume called");

        }

        @Override
        public void surfaceChanged(SurfaceHolder holder, int format, int width,int height) {
            // TODO Auto-generated method stub
        }

        @Override
        public void surfaceCreated(SurfaceHolder holder) {
            // TODO Auto-generated method stub
```

```
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        // TODO Auto-generated method stub
    }

    public Handler getHandler(){
        return handler;
    }

    private void doLogin(final String userName,final String password){
        Log.v(TAG,"doLogin");
        progressDialog = ProgressDialog.show(ResultActivity.this, "Hold On a second",
"Connecting to Server", true);

        final String url = SERVER_URL + "login.json";

        // start a new thread to connect to web server
        new Thread(){
            public void run(){
                HashMap<String,String> paramMap = new HashMap<String,String>();
                paramMap.put("userName", userName);
                paramMap.put("password", password);
                paramMap.put("studentId", studentId);
                Log.v(TAG,"userName:"+paramMap.get("userName")+",
password:"+paramMap.get("password")+",student id:"+paramMap.get("studentId"));

                sd = MBRHttpClient.doRequest(url, paramMap);

                progressDialog.dismiss();
//                synLock = true;

                if(sd.getLoginState().equals("fail")){
                    // send message to handler to deal login fail
                    Message message = Message.obtain(handler, R.id.login_fail);
                    message.sendToTarget();
                }else{
                    // send message to handler to deal setDetail
                    Message message = Message.obtain(handler, R.id.set_detail, sd);
                    message.sendToTarget();
                }
            }
        }.start();
```

```
        Log.v(TAG,"doLogin finished");
    }

    public void setLoginFailAlert(){
        new AlertDialog.Builder(ResultActivity.this).setMessage("Login failed,\nuser name and
password invalid.").setPositiveButton("OK", null).show();
        userName_EditText.setText("");
        password_EditText.setText("");
    }

    public void setDetail(StudentDetail sd){

//      barcode_TextView.setVisibility(View.GONE);
        username_Tag.setVisibility(View.GONE);
        userName_EditText.setVisibility(View.GONE);
        password_Tag.setVisibility(View.GONE);
        password_EditText.setVisibility(View.GONE);
        getDetail_Button.setVisibility(View.GONE);

        barcode_TextView.setText(studentId);

//      if(sd.getNotFound().equals("true")){
        if(sd.getId().equals("Student Not Found")){
            notFound_TextView.setVisibility(View.VISIBLE);
            notFound_TextView.setText(sd.getId());
        }else{
            firstname_Tag.setVisibility(View.VISIBLE);
            firstname_TextView.setVisibility(View.VISIBLE);
            surname_Tag.setVisibility(View.VISIBLE);
            surname_TextView.setVisibility(View.VISIBLE);
            gender_Tag.setVisibility(View.VISIBLE);
            gender_TextView.setVisibility(View.VISIBLE);
            dateOfBirth_Tag.setVisibility(View.VISIBLE);
            dateOfBirth_TextView.setVisibility(View.VISIBLE);
            course_Tag.setVisibility(View.VISIBLE);
            course_TextView.setVisibility(View.VISIBLE);
            courseCode_Tag.setVisibility(View.VISIBLE);
            courseCode_TextView.setVisibility(View.VISIBLE);

            firstname_TextView.setText(sd.getFirstname());
            surname_TextView.setText(sd.getSurname());
            gender_TextView.setText(sd.getGender());
            dateOfBirth_TextView.setText(sd.getDateOfBirth());
            course_TextView.setText(sd.getCourse());
```

```
                courseCode_TextView.setText(sd.getCourseCode());
            }
        }
}
```

1.10 ResultActivityHandler.java

```
package ll.mbr;

import android.os.Handler;
import android.os.Message;
import android.util.Log;

public class ResultActivityHandler extends Handler{
    private final static String TAG = "ResultActivityHandler";

    private final ResultActivity activity;

    ResultActivityHandler(ResultActivity activity){
        this.activity = activity;
    }

    @Override
    public void handleMessage(Message message) {
        switch (message.what){
            case R.id.set_detail:
                Log.v(TAG,"set detail");
                activity.setDetail((StudentDetail) message.obj);
                break;
            case R.id.login_fail:
                Log.v(TAG,"login fail");
                activity.setLoginFailAlert();
                break;
        }
    }
}
```

1.11 StudentDetail.java

```
package ll.mbr;
```

```
public class StudentDetail {

    private String loginState;
//  private String notFound;

    // student detail
    private String studentId;
    private String surname;
    private String firstname;
    private String gender;
    private String homeAddress;
    private String dateOfBirth;
    private String tel;
    private String mobile;
    private String email;
    private String course;
    private String courseCode;
    private String employmentRecord;

    public StudentDetail(){

    }

    public StudentDetail(String loginState){
        this.loginState = loginState;
    }

    public StudentDetail(String studentId,
            String surname,
            String firstname,
            String gender,
            String homeAddress,
            String dateOfBirth,
            String tel,
            String mobile,
            String email,
            String course,
            String courseCode,
            String employmentRecord){
        this.studentId = studentId;
        this.surname = surname;
        this.firstname = firstname;
        this.gender = gender;
        this.homeAddress = homeAddress;
```

```
        this.dateOfBirth = dateOfBirth;
        this.tel = tel;
        this.mobile = mobile;
        this.email = email;
        this.course = course;
        this.courseCode = courseCode;
        this.employmentRecord = employmentRecord;
    }

    public void setLoginState(String loginState){
        this.loginState = loginState;
    }

    public String getLoginState(){
        return loginState;
    }

//  public void setNotFound(String notFound){
//      this.notFound = notFound;
//  }
//
//  public String getNotFound(){
//      return notFound;
//  }

    public void setId(String studentId){
        this.studentId = studentId;
    }

    public String getId(){
        return this.studentId;
    }

    public void setSurname(String surname){
        this.surname = surname;
    }

    public String getSurname(){
        return this.surname;
    }

    public void setFirstname(String firstname){
        this.firstname = firstname;
    }
```

```
public String getFirstname(){
    return this.firstname;
}

public void setGender(String gender){
    this.gender = gender;
}

public String getGender(){
    return this.gender;
}

public void setHomeAddress(String homeAddress){
    this.homeAddress = homeAddress;
}

public String getHomeAddress(){
    return this.homeAddress;
}

public void setDateOfBirth(String dateOfBirth){
    this.dateOfBirth = dateOfBirth;
}

public String getDateOfBirth(){
    return this.dateOfBirth;
}

public void setTel(String tel){
    this.tel = tel;
}

public String getTel(){
    return this.tel;
}

public void setMobile(String mobile){
    this.mobile = mobile;
}

public String getMobile(){
    return this.mobile;
}
```

```
    public void setEmail(String email){
        this.email = email;
    }

    public String getEmail(){
        return this.email;
    }

    public void setCourse(String course){
        this.course = course;
    }

    public String getCourse(){
        return this.course;
    }

    public void setCourseCode(String courseCode){
        this.courseCode = courseCode;
    }

    public String getCourseCode(){
        return this.courseCode;
    }

    public void setEmploymentRecord(String employmentRecord){
        this.employmentRecord = employmentRecord;
    }

    public String getEmploymentRecord(){
        return this.employmentRecord;
    }
}
```

2. RES

2.1 layout

2.1.1 detail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
android:id="@+id/widget0"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<EditText
android:id="@+id/userName_EditText"
android:layout_width="120px"
android:layout_height="wrap_content"
android:textSize="18sp"
android:layout_x="10px"
android:layout_y="102px"
>
</EditText>
<TextView
android:id="@+id/username_Tag"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="User name:"
android:layout_x="0px"
android:layout_y="82px"
>
</TextView>
<EditText
android:id="@+id/password_EditText"
android:layout_width="120px"
android:layout_height="wrap_content"
android:textSize="18sp"
android:layout_x="10px"
android:layout_y="172px"
>
</EditText>
<TextView
```

```
android:id="@+id/password_Tag"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Password"
android:layout_x="0px"
android:layout_y="152px"
>
</TextView>
<Button
android:id="@+id/getDetail_Button"
android:layout_width="79px"
android:layout_height="40px"
android:text="Get Detail"
android:textSize="12sp"
android:layout_x="10px"
android:layout_y="232px"
>
</Button>
<TextView
android:id="@+id/widget55"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Mobile Barcode Recognition"
android:layout_x="0px"
android:layout_y="2px"
>
</TextView>
<TextView
android:id="@+id/widget57"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Found Student Id:"
android:layout_x="10px"
android:layout_y="22px"
>
</TextView>
<TextView
android:id="@+id/barcode_TextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Stu. Id"
android:textSize="20sp"
android:textStyle="bold"
android:layout_x="30px"
```

```
android:layout_y="42px"
>
</TextView>
<TextView
android:id="@+id/firstname_Tag"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:visibility="gone"
android:text="Fisrtname"
android:layout_x="140px"
android:layout_y="72px"
>
</TextView>
<TextView
android:id="@+id/firstname_TextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:visibility="gone"
android:text="firstname"
android:textSize="18sp"
android:layout_x="160px"
android:layout_y="102px"
>
</TextView>
<TextView
android:id="@+id/surname_Tag"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:visibility="gone"
android:text="Surname"
android:layout_x="270px"
android:layout_y="72px"
>
</TextView>
<TextView
android:id="@+id/surname_TextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:visibility="gone"
android:text="surname"
android:textSize="18sp"
android:layout_x="310px"
android:layout_y="102px"
>
```

```
</TextView>
<TextView
android:id="@+id/gender_Tag"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:visibility="gone"
android:text="Gender"
android:layout_x="140px"
android:layout_y="142px"
>
</TextView>
<TextView
android:id="@+id/gender_TextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:visibility="gone"
android:text="gender"
android:textSize="18sp"
android:layout_x="170px"
android:layout_y="172px"
>
</TextView>
<TextView
android:id="@+id/course_Tag"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:visibility="gone"
android:text="Course"
android:layout_x="140px"
android:layout_y="212px"
>
</TextView>
<TextView
android:id="@+id/course_TextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:visibility="gone"
android:text="course"
android:textSize="18sp"
android:layout_x="170px"
android:layout_y="242px"
>
</TextView>
<TextView
```

```
android:id="@+id/courseCode_Tag"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:visibility="gone"
android:text="CourseCode"
android:layout_x="270px"
android:layout_y="212px"
>
</TextView>
<TextView
android:id="@+id/courseCode_TextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:visibility="gone"
android:text="courseCode"
android:textSize="18sp"
android:layout_x="300px"
android:layout_y="242px"
>
</TextView>
<TextView
android:id="@+id/notFound_TextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:visibility="gone"
android:text="notFound"
android:textSize="28px"
android:layout_x="90px"
android:layout_y="132px"
>
</TextView>
<TextView
android:id="@+id/dateOfBirth_Tag"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:visibility="gone"
android:text="DateOfBirth"
android:layout_x="270px"
android:layout_y="142px"
>
</TextView>
<TextView
android:id="@+id/dateOfBirth_TextView"
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:visibility="gone"
android:text="dateOfBirth"
android:textSize="18sp"
android:layout_x="300px"
android:layout_y="172px"
>
</TextView>
</AbsoluteLayout>
```

2.1.2 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="fill_parent"
    >
    <SurfaceView  android:id="@+id/camera"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    />
    <ImageView android:id="@+id/image"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    />
</LinearLayout>
```

2.2 values

2.2.1 ids.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item type="id" name="auto_focus"/>
    <item type="id" name="decode"/>
    <item type="id" name="quit"/>
    <item type="id" name="decode_failed"/>
    <item type="id" name="decode_succeeded"/>
```

```
    <item type="id" name="restart_preview"/>
    <item type="id" name="set_detail"/>
    <item type="id" name="login_fail"/>
</resources>
```

2.2.2 strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, CameraActivity!</string>
    <string name="app_name">MBR</string>
</resources>
```

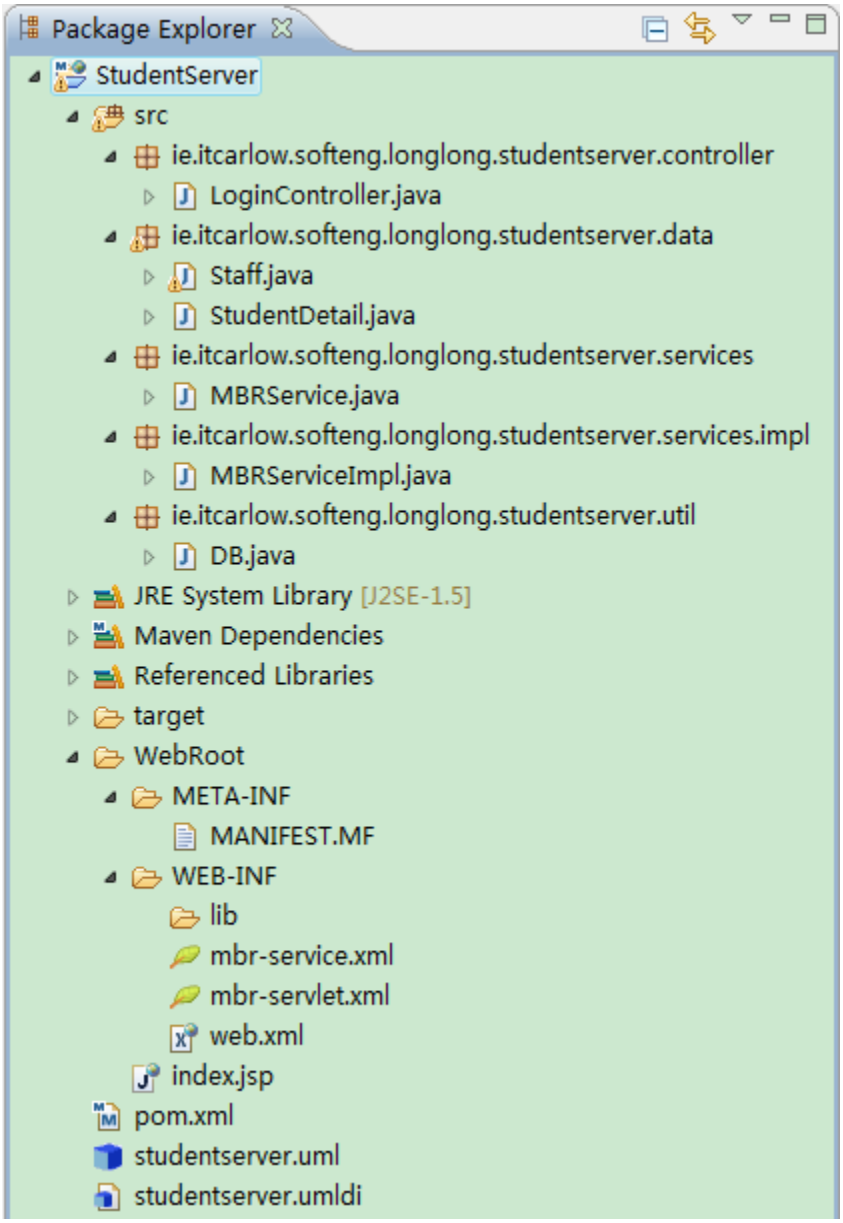
3. AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="ll.mbr"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
android:label="@string/app_name" android:debuggable="true">
        <activity android:name=".CaptureActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".ResultActivity"
                  android:label="@string/app_name">
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="4" />
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.INTERNET"/>

</manifest>
```

Web Server Hierarchy tree



1. SRC

1.1 ie.itcarlow.softeng.longlong.studentserver.controller

1.1.1 LoginController.java

```
package ie.itcarlow.softeng.longlong.studentserver.controller;

import ie.itcarlow.softeng.longlong.studentserver.data.StudentDetail;
import ie.itcarlow.softeng.longlong.studentserver.services.MBRService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
@RequestMapping("/login.json")
public class LoginController {
	private MBRService mbrService;

	@RequestMapping(method = RequestMethod.POST)
	public String handleLogin(@RequestParam("userName") String userName,
			@RequestParam("password") String password,
			@RequestParam("studentId")String studentId,
			ModelMap modelMap) throws Exception {

//		if(mbrService.login(userName,  password)){
//			modelMap.put("loginState","valid");
//		}
//
//		return "list";

		if(mbrService.login(userName,  password)){
		modelMap.put("loginState", "valid");
		StudentDetail sd = mbrService.retrieve(studentId);
		if(sd!=null){
			modelMap.put("studentId",sd.getId());
			modelMap.put("surname",sd.getSurname());
```

```
                modelMap.put("firstname",sd.getFirstname());
                modelMap.put("gender",sd.getGender());
                modelMap.put("homeAddress",sd.getHomeAddress());
                modelMap.put("dateOfBirth",sd.getDateOfBirth());
                modelMap.put("tel",sd.getTel());
                modelMap.put("mobile",sd.getMobile());
                modelMap.put("email",sd.getEmail());
                modelMap.put("course",sd.getCourse());
                modelMap.put("courseCode",sd.getCourseCode());
                modelMap.put("employmentRecord",sd.getEmploymentRecord());
            }else{
                modelMap.put("studentId", "Student Not Found");
            }
        }else{
            modelMap.put("loginState", "fail");
        }

        return "student";
        }

        @Autowired
        public void setMBRService(MBRService mbrService){
            this.mbrService = mbrService;
        }
}
```

1.2 ie.itcarlow.softeng.longlong.studentserver.data

1.2.1 Staff.java

```
package ie.itcarlow.softeng.longlong.studentserver.data;

import ie.itcarlow.softeng.longlong.studentserver.util.DB;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;

public class Staff {

    private String staffId;
    private String password;
```

```
	public static Boolean checkValidity(String userName,String password){
		Connection conn = null;
		ResultSet rs = null;
		try{
			conn = DB.getConn();
			String sql = "select * from staff where staffId = '" + userName + "'";
			rs = DB.executeQuery(conn, sql);
			if(!rs.next()){
//				throw new StudentNotFoundException();
				return false;
			}else if(!rs.getString("password").equals(password)){
				return false;
			}
		}catch(SQLException ex){
			ex.printStackTrace();
		}finally{
			DB.close(conn);
			DB.close(rs);
		}
		// if userName and password both valid, login succeed return true
		return true;
	}
}
```

1.2.2 StudentDetail.java

```
package ie.itcarlow.softeng.longlong.studentserver.data;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;

import ie.itcarlow.softeng.longlong.studentserver.util.DB;

public class StudentDetail {

//	private String notFound;

	// student detail
	private String studentId;
	private String surname;
	private String firstname;
```

```
    private String gender;
    private String homeAddress;
    private String dateOfBirth;
    private String tel;
    private String mobile;
    private String email;
    private String course;
    private String courseCode;
    private String employmentRecord;



    public StudentDetail(){

    }

    public StudentDetail(String studentId,
            String surname,
            String firstname,
            String gender,
            String homeAddress,
            String dateOfBirth,
            String tel,
            String mobile,
            String email,
            String course,
            String courseCode,
            String employmentRecord){
        this.studentId = studentId;
        this.surname = surname;
        this.firstname = firstname;
        this.gender = gender;
        this.homeAddress = homeAddress;
        this.dateOfBirth = dateOfBirth;
        this.tel = tel;
        this.mobile = mobile;
        this.email = email;
        this.course = course;
        this.courseCode = courseCode;
        this.employmentRecord = employmentRecord;
    }

//  public String getNotFound(){
//      return this.notFound;
```

```
//    }
//
//    public void setNotFound(String notFound){
//        this.notFound = notFound;
//    }

    public void setId(String studentId){
        this.studentId = studentId;
    }

    public String getId(){
        return this.studentId;
    }

    public void setSurname(String surname){
        this.surname = surname;
    }

    public String getSurname(){
        return this.surname;
    }

    public void setFirstname(String firstname){
        this.firstname = firstname;
    }

    public String getFirstname(){
        return this.firstname;
    }

    public void setGender(String gender){
        this.gender = gender;
    }

    public String getGender(){
        return this.gender;
    }

    public void setHomeAddress(String homeAddress){
        this.homeAddress = homeAddress;
    }

    public String getHomeAddress(){
        return this.homeAddress;
```

```
}

public void setDateOfBirth(String dateOfBirth){
	this.dateOfBirth = dateOfBirth;
}

public String getDateOfBirth(){
	return this.dateOfBirth;
}

public void setTel(String tel){
	this.tel = tel;
}

public String getTel(){
	return this.tel;
}

public void setMobile(String mobile){
	this.mobile = mobile;
}

public String getMobile(){
	return this.mobile;
}

public void setEmail(String email){
	this.email = email;
}

public String getEmail(){
	return this.email;
}

public void setCourse(String course){
	this.course = course;
}

public String getCourse(){
	return this.course;
}

public void setCourseCode(String courseCode){
	this.courseCode = courseCode;
```

```
	}

	public String getCourseCode(){
		return this.courseCode;
	}

	public void setEmploymentRecord(String employmentRecord){
		this.employmentRecord = employmentRecord;
	}

	public String getEmploymentRecord(){
		return this.employmentRecord;
	}


	public static StudentDetail retrieveStudent(String studentId){
		Connection conn = null;
		ResultSet rs = null;
		StudentDetail sd = null;
		try{
			conn = DB.getConn();
			String sql = "select * from stu_details where studentId = '" + studentId + "'";
			rs = DB.executeQuery(conn, sql);
			if(!rs.next()){
//				throw new StudentNotFoundException();
				return null;
			}else{
				sd = new StudentDetail();
				sd.setId(rs.getString("studentId"));
				sd.setSurname(rs.getString("surname"));
				sd.setFirstname(rs.getString("firstname"));
				sd.setGender(rs.getString("gender"));
				sd.setHomeAddress(rs.getString("homeAddress"));
				sd.setDateOfBirth(rs.getString("dateOfBirth"));
				sd.setTel(rs.getString("tel"));
				sd.setMobile(rs.getString("mobile"));
				sd.setEmail(rs.getString("email"));
				sd.setCourse(rs.getString("course"));
				sd.setCourseCode(rs.getString("courseCode"));
				sd.setEmploymentRecord(rs.getString("employmentRecord"));
//				System.out.println(sd.getId());
//				System.out.println(rs.getString("studentId"));
			}
		}catch(SQLException ex){
```

```
            ex.printStackTrace();
        }finally{
            DB.close(conn);
            DB.close(rs);
        }
        return sd;
    }
}
```

1.3 ie.itcarlow.softeng.longlong.studentserver.services

1.3.1 MBRService.java

```
package ie.itcarlow.softeng.longlong.studentserver.services;

import ie.itcarlow.softeng.longlong.studentserver.data.StudentDetail;

public interface MBRService {
    public boolean login(String userName,String password);
    public StudentDetail retrieve(String studentId);
}
```

1.4 ie.itcarlow.softeng.longlong.studentserver.services.impl

1.4.1 MBRServiceImpl.java

```
package ie.itcarlow.softeng.longlong.studentserver.services.impl;

import ie.itcarlow.softeng.longlong.studentserver.data.Staff;
import ie.itcarlow.softeng.longlong.studentserver.data.StudentDetail;
import ie.itcarlow.softeng.longlong.studentserver.services.MBRService;

public class MBRServiceImpl implements MBRService {

    public boolean login(String userName, String password) {
//        if (userName.equals("staff") && password.equals("staff")) {
//            System.out.println("Login succeed");
```

```
//                  return true;
//            } else {
//                  System.out.println("Login failed");
//                  return false;
//            }

            if(Staff.checkValidity(userName,  password)){
                  System.out.println("Login succeed");
                  return true;
            }else{
                  System.out.println("Login failed");
                  return false;
            }
      }

      public  StudentDetail retrieve(String studentId) {
            // TODO Auto-generated method stub
            StudentDetail sd = null;
            sd = StudentDetail.retrieveStudent(studentId);
            return sd;
      }
}
```

1.5 ie.itcarlow.softeng.longlong.studentserver.util

1.5.1 DB.java

```
package ie.itcarlow.softeng.longlong.studentserver.util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public  class DB {

      static {
            try {
                  Class.forName("com.mysql.jdbc.Driver");
            } catch (ClassNotFoundException  e) {
```

```
                e.printStackTrace();
            }
        }

        private DB() {
        }

        public static Connection getConn() {
            Connection conn = null;
            try {
                conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/student?user=root&password=root"
);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return conn;
        }

        public static void close(Connection conn) {
            try {
                if (conn != null) {
                    conn.close();
                    conn = null;
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        public static Statement getStmt(Connection conn) {
            Statement stmt = null;
            try {
                stmt = conn.createStatement();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return stmt;
        }

        public static PreparedStatement getPStmt(Connection conn, String sql) {
            PreparedStatement pStmt = null;
            try {
                pStmt = conn.prepareStatement(sql);
```

```
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return pStmt;
    }

    public static ResultSet executeQuery(Statement stmt, String sql) {
        ResultSet rs = null;
        try {
            rs = stmt.executeQuery(sql);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return rs;
    }

    public static void close(Statement stmt) {
        try {
            if (stmt != null) {
                stmt.close();
                stmt = null;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static void close(ResultSet rs) {
        try {
            if (rs != null) {
                rs.close();
                rs = null;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static ResultSet executeQuery(Connection conn, String sql) {
        ResultSet rs = null;
        try {
            rs = conn.createStatement().executeQuery(sql);
        } catch (SQLException e) {
            e.printStackTrace();
```

```
            }
            return rs;
        }

        public static int executeUpdate(Connection conn, String sql) {
            int update = 0;
            try {
                update = conn.createStatement().executeUpdate(sql);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return update;
        }

}
```