Nama : Aura Kanza Caesaria
Kelas : TI-4E
NIM : 1541180188

MODUL 4 LIBRARY

- **Manifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.aura.modul4">

  <meta-data
     android:name="com.google.android.gms.vision.DEPENDENCIES"
     android:value="face" />
  <application
     android:allowBackup="true"
     android:icon="@mipmap/ic_launcher"
     android:label="@string/app_name"
     android:roundIcon="@mipmap/ic_launcher_round"
     android:supportsRtl="true"
     android:theme="@style/AppTheme">
     <activity android:name=".MainActivity">
        <intent-filter>
           <action android:name="android.intent.action.MAIN" />

           <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
     </activity>
  </application>

</manifest>
```

- **MainActivity.java**

```
package com.example.aura.modul4;


import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.RectF;
import android.graphics.drawable.BitmapDrawable;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.util.SparseArray;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
```

```
import com.google.android.gms.vision.Frame;
import com.google.android.gms.vision.face.Face;
import com.google.android.gms.vision.face.FaceDetector;

public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);

     Button btn = (Button) findViewById(R.id.button);
     btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           ImageView myImageView = (ImageView) findViewById(R.id.imgview);

//            new emoji
           Emojifier emoji = new Emojifier();

           BitmapFactory.Options options = new BitmapFactory.Options();
           options.inMutable=true;
           Bitmap mBitmap = BitmapFactory.decodeResource(
            getApplicationContext().getResources(),
                 R.drawable.tarno,
                 options
           );

//            Paint myRectPaint = new Paint();
//            myRectPaint.setStrokeWidth(5);
//            myRectPaint.setColor(Color.YELLOW);
//            myRectPaint.setStyle(Paint.Style.STROKE);

//            FaceDetector fd =
//                  new FaceDetector.Builder(getApplicationContext())
//                  .setTrackingEnabled(false)
//                  .build();
//            if(!fd.isOperational()){
//               new AlertDialog.Builder(v.getContext())
//                     .setMessage("gabisa"+ "terdeteksi!")
//                     .show();
//               return;
//            }

//            Frame frame = new Frame.Builder().setBitmap(mBitmap).build();
//            SparseArray<Face> faces = fd.detect(frame);

           Bitmap tempBitmap = Bitmap.createBitmap(mBitmap.getWidth(), mBitmap.getHeight(),
Bitmap.Config.RGB_565);
//            Canvas tempCanvas = new Canvas(tempBitmap);
//            tempCanvas.drawBitmap(mBitmap, 0, 0 , null);

//            for(int i =0 ; i<faces.size(); i++){
//               Face thisFace = faces.valueAt(i);
//               float  x1 = thisFace.getPosition().x;
//               float y1 = thisFace.getPosition().y;
//               float x2 = x1 + thisFace.getWidth();
//               float y2 = y1 + thisFace.getHeight();
//               tempCanvas.drawRoundRect(new RectF(x1, y1, x2, y2), 2,2, myRectPaint);
```

```
//              }
//              myImageView.setImageDrawable( new BitmapDrawable(getResources(),tempBitmap));
            myImageView.setImageBitmap(emoji.detectFaces(getApplicationContext(),mBitmap));
        }
    });
  }
}
```

- **Emojifier.java**

```
package com.example.aura.modul4;


import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.util.SparseArray;
import android.widget.Toast;
import android.util.Log;

import com.google.android.gms.vision.Frame;
import com.google.android.gms.vision.face.Face;
import com.google.android.gms.vision.face.FaceDetector;

import timber.log.Timber;

public class Emojifier {
  private static double SMILING_PROP_THRESHOLD = .15;
  private static double EYE_OPEN_PROP_THRESHOLD = .5;
  private static float EMOJI_SCALE_FACTOR=.9f;

  // Enum for all possible Emojis
  private enum Emoji {
     SMILE,
     FROWN,
     LEFT_WINK,
     RIGHT_WINK,
     LEFT_WINK_FROWN,
     RIGHT_WINK_FROWN,
     CLOSED_EYE_SMILE,
     CLOSED_EYE_FROWN
  }

  private static final String TAG = Emojifier.class.getSimpleName();

  public static Bitmap detectFaces(Context context, Bitmap image) {
     //get the detector
     FaceDetector detector = new FaceDetector.Builder(context)
           .setTrackingEnabled(false)
           .setClassificationType(FaceDetector.ALL_CLASSIFICATIONS)
           .build();
     Frame frame = new Frame.Builder().setBitmap(image).build();
     SparseArray<Face> faces = detector.detect(frame);
     Timber.d("number of faces= " + faces.size());
     Bitmap resultBitmap = image;
     if (faces.size() == 0) {
```

```
            Toast.makeText(context, R.string.no_faces_message, Toast.LENGTH_SHORT).show();
        } else {
            for (int i = 0; i < faces.size(); i++) {
                Face face = faces.valueAt(i);
                Bitmap emojiBitmap;
                switch (whichEmoji(face)) {
                    case SMILE:
                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
                                R.drawable.smile);
                        break;
                    case FROWN:
                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
                                R.drawable.frown);
                        break;
                    case LEFT_WINK:
                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
                                R.drawable.leftwink);
                        break;
                    case RIGHT_WINK:
                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
                                R.drawable.rightwink);
                        break;
//                    case LEFT_WINK_FROWN:
//                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
//                                R.drawable.leftwinkfrown);
//                        break;
//                    case RIGHT_WINK_FROWN:
//                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
//                                R.drawable.rightwinkfrown);
//                        break;
                    case CLOSED_EYE_SMILE:
                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
                                R.drawable.happy2);
                        break;
                    case CLOSED_EYE_FROWN:
                        emojiBitmap = BitmapFactory.decodeResource(context.getResources(),
                                R.drawable.sad);
                        break;
                    default:
                        emojiBitmap = null;
                        Toast.makeText(context, R.string.no_emoji, Toast.LENGTH_SHORT).show();
                }
                // Add the emojiBitmap to the proper position in the original image
                resultBitmap = addBitmapToFace(resultBitmap, emojiBitmap, face);
            }
        }
        detector.release();
        return resultBitmap;
    }

    private static Bitmap addBitmapToFace(Bitmap backgroundBitmap, Bitmap emojiBitmap, Face
face) {
        // Initialize the results bitmap to be a mutable copy of the original image
        Bitmap resultBitmap = Bitmap.createBitmap(backgroundBitmap.getWidth(),
                backgroundBitmap.getHeight(), backgroundBitmap.getConfig());

        // Scale the emoji so it looks better on the face
        float scaleFactor = EMOJI_SCALE_FACTOR;

        // Determine the size of the emoji to match the width of the face and preserve aspect ratio
```

```
        int newEmojiWidth = (int) (face.getWidth() * scaleFactor);
        int newEmojiHeight = (int) (emojiBitmap.getHeight() *
                newEmojiWidth / emojiBitmap.getWidth() * scaleFactor);


        // Scale the emoji
        emojiBitmap = Bitmap.createScaledBitmap(emojiBitmap, newEmojiWidth, newEmojiHeight,
false);

        // Determine the emoji position so it best lines up with the face
        float emojiPositionX =
                (face.getPosition().x + face.getWidth() / 2) - emojiBitmap.getWidth() / 2;
        float emojiPositionY =
                (face.getPosition().y + face.getHeight() / 2) - emojiBitmap.getHeight() / 3;

        // Create the canvas and draw the bitmaps to it
        Canvas canvas = new Canvas(resultBitmap);
        canvas.drawBitmap(backgroundBitmap, 0, 0, null);
        canvas.drawBitmap(emojiBitmap, emojiPositionX, emojiPositionY, null);

        return resultBitmap;
    }

    private static Emoji whichEmoji(Face face) {
        Timber.d( "getClassifications: smilingProb = " + face.getIsSmilingProbability());
        Timber.d("getClassifications: leftEyeOpenProb = "
                + face.getIsLeftEyeOpenProbability());
        Timber.d("getClassifications: rightEyeOpenProb = "
                + face.getIsRightEyeOpenProbability());
        boolean smiling = face.getIsSmilingProbability() > SMILING_PROP_THRESHOLD;
        boolean leftEyeClosed = face.getIsLeftEyeOpenProbability() <
EYE_OPEN_PROP_THRESHOLD;
        boolean rightEyeClosed = face.getIsRightEyeOpenProbability() <
EYE_OPEN_PROP_THRESHOLD;
        // Determine and log the appropriate emoji
        Emoji emoji;
        if (smiling) {
            if (leftEyeClosed && !rightEyeClosed) {
                emoji = Emoji.LEFT_WINK;
            } else if (rightEyeClosed && !leftEyeClosed) {
                emoji = Emoji.RIGHT_WINK;
            } else if (leftEyeClosed) {
                emoji = Emoji.CLOSED_EYE_SMILE;
            } else {
                emoji = Emoji.SMILE;
            }
        } else {
            if (leftEyeClosed && !rightEyeClosed) {
                emoji = Emoji.LEFT_WINK_FROWN;
            } else if (rightEyeClosed && !leftEyeClosed) {
                emoji = Emoji.RIGHT_WINK_FROWN;
            } else if (leftEyeClosed) {
                emoji = Emoji.CLOSED_EYE_FROWN;
            } else {
                emoji = Emoji.FROWN;
            }
        }
        // Log the chosen Emoji
        Timber.d("whichEmoji: " + emoji.name());
        return emoji;
```

```
    }
}
```

- **activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Process"
        android:id="@+id/button"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"/>

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imgview"/>

</RelativeLayout>
```

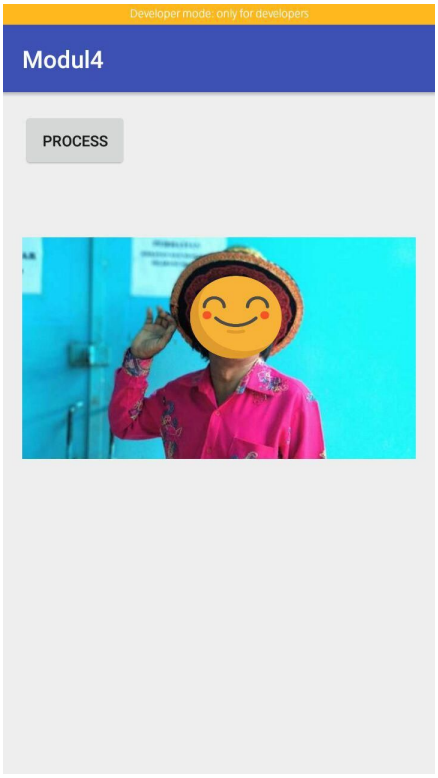
Hasil :

- **Gambar 1**

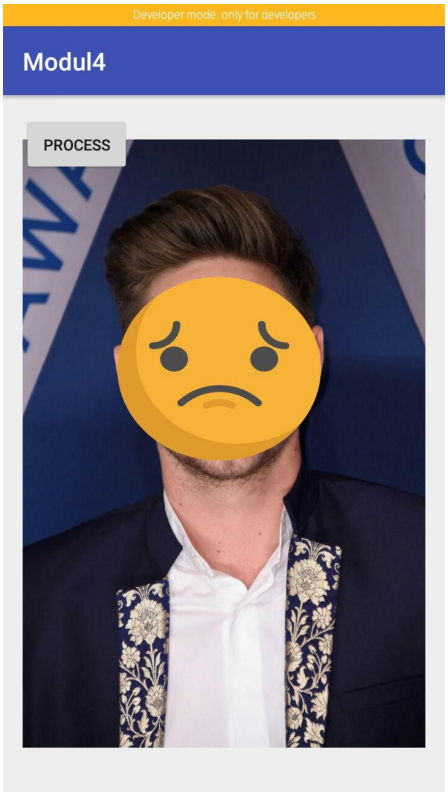
- **Gambar 2**

- **Gambar 3**

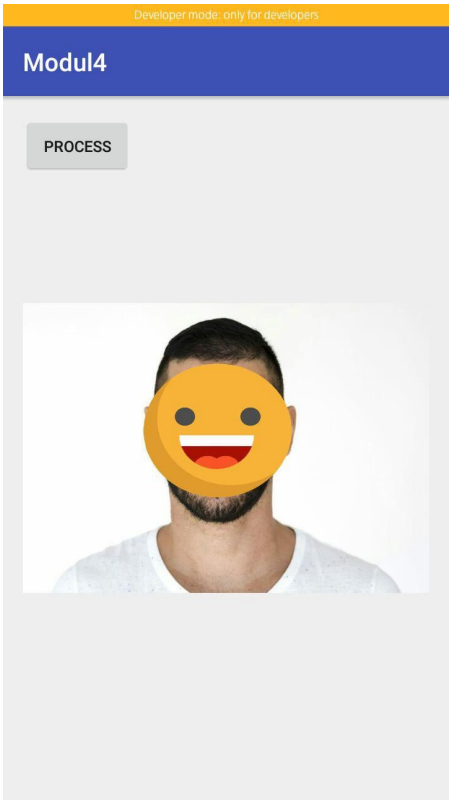
- **Hasil setelah gambar di proses**
  - **Happy**



  - **Sad**



- **Close eyes**



Kesimpulan : Emoji akan mengikuti ekspresi wajah pada foto dan emoji dapat kita sesuaikan dengan ekspresi wajah