1. Add the following lines to **AndroidManifest.xml** before <application> tag:

**<uses-permission android:name="android.permission.INTERNET"/>**

**<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />**

**<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />**

**Face Recognition (in MainActivity.java)**

1. Declare the following variables in MainActivity class:

private static final String *TAG* = "MainActivity";

private ImageView mImageView;

private Button mLoadButton;

private Button mFaceButton;

private Button mBarcodeBtn;

private Bitmap mSelectedImage;

private GraphicOverlay mGraphicOverlay;

// Max width (portrait mode)

private Integer mImageMaxWidth;

// Max height (portrait mode)

private Integer mImageMaxHeight;

private static int *RESULT\_LOAD\_IMAGE* = 1;

1. Put the following code at the bottom of the **onCreate** method. Finds the view from the *activity\_main.xml* attached with MainActivity (in **onCreate** method):

mImageView = findViewById(R.id.*image\_view*);

mLoadButton = findViewById(R.id.*load\_btn*);

mFaceButton = findViewById(R.id.*button\_face*);

mBarcodeBtn = findViewById(R.id.*barcode\_btn*);

mGraphicOverlay = findViewById(R.id.*graphic\_overlay*);

1. Set onClickedListener for Load Image button and Detect Face button (in **onCreate** method) [ignore error]:

mLoadButton.setOnClickListener(new View.OnClickListener() {

   @Override

   public void onClick(View view) {

       Intent i = new Intent(

               Intent.*ACTION\_PICK*,

               android.provider.MediaStore.Images.Media.*EXTERNAL\_CONTENT\_URI*);

       startActivityForResult(i, *RESULT\_LOAD\_IMAGE*);

   }

});

// Add on click listener for face

mFaceButton.setOnClickListener(new View.OnClickListener() {

   @Override

   public void onClick(View view) {

       if(mSelectedImage != null)

           runFaceContourDetection();

       else

           showToast("Please load image first");

   }

});

1. Check and request Read permission from users (in **onCreate** method):

// Acquire read permission from external storage

if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {

   if (checkSelfPermission(Manifest.permission.*READ\_EXTERNAL\_STORAGE*)

           == PackageManager.*PERMISSION\_GRANTED*) {

       Log.*v*(*TAG*,"Permission is granted");

   } else {

       Log.*v*(*TAG*,"Permission is revoked");

       ActivityCompat.*requestPermissions*(this, new String[]{Manifest.permission.*READ\_EXTERNAL\_STORAGE*}, 1);

   }

}

else {

   // permission is automatically granted on sdk<23 upon installation

   Log.*v*(*TAG*,"Permission is granted");

}

1. Right after **onCreate** method, add the following method. This method is a callback for the result from requesting permissions:

@Override

public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {

   super.onRequestPermissionsResult(requestCode, permissions, grantResults);

   if(grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*){

       Log.*v*(*TAG*,"Permission: " + permissions[0] +  "was " + grantResults[0]);

       //resume tasks needing this permission

   }

}

1. Add the following methods in MainActivity class (show toast to users, get image maximum width and height):

private void showToast(String message) {

   Toast.*makeText*(getApplicationContext(), message, Toast.*LENGTH\_SHORT*).show();

}

private Integer getImageMaxWidth() {

   if (mImageMaxWidth == null) {

       mImageMaxWidth = mImageView.getWidth();

   }

   return mImageMaxWidth;

}

private Integer getImageMaxHeight() {

   if (mImageMaxHeight == null) {

       mImageMaxHeight =

               mImageView.getHeight();

   }

   return mImageMaxHeight;

}

// Gets the targeted width / height.

private Pair<Integer, Integer> getTargetedWidthHeight() {

   int targetWidth;

   int targetHeight;

   int maxWidthForPortraitMode = getImageMaxWidth();

   int maxHeightForPortraitMode = getImageMaxHeight();

   targetWidth = maxWidthForPortraitMode;

   targetHeight = maxHeightForPortraitMode;

   return new Pair<>(targetWidth, targetHeight);

}

1. Override **onActivityResult** method in **MainActivity** class:

@Override

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

   super.onActivityResult(requestCode, resultCode, data);

   if (requestCode == *RESULT\_LOAD\_IMAGE* && resultCode == *RESULT\_OK* && null != data) {

       try {

  // Obtain the path to the image selected in the gallery

           Uri imageUri = data.getData();

  // Open a stream on to the content associated with a content URI

           InputStream imageStream = getContentResolver().openInputStream(imageUri);

           mSelectedImage = BitmapFactory.*decodeStream*(imageStream);

           mGraphicOverlay.clear();

           if (mSelectedImage != null) {

               // Get the dimensions of the View

               Pair<Integer, Integer> targetedSize = getTargetedWidthHeight();

               int targetWidth = targetedSize.first;

               int maxHeight = targetedSize.second;

               // Determine how much to scale down the image

               float scaleFactor =

                       Math.*max*(

                               (float) mSelectedImage.getWidth() / (float) targetWidth,

                               (float) mSelectedImage.getHeight() / (float) maxHeight);

               Bitmap resizedBitmap =

                       Bitmap.*createScaledBitmap*(

                               mSelectedImage,

                               (int) (mSelectedImage.getWidth() / scaleFactor),

                               (int) (mSelectedImage.getHeight() / scaleFactor),

                               true);

               mImageView.setImageBitmap(resizedBitmap);

               mSelectedImage = resizedBitmap;

           }

       } catch (Exception e){

           e.printStackTrace();

       }

   }

}

1. Create a method called runFaceContourDetection:

private void runFaceContourDetection() {

// ...

}

1. Inside **runFaceContourDetection** method (Step 9 - 12), create a FirebaseVisionImage object from a Bitmap:

FirebaseVisionImage image = FirebaseVisionImage.*fromBitmap*(mSelectedImage);

1. Configure the face detector: (Refer: https://firebase.google.com/docs/ml-kit/android/detect-faces#1.-configure-the-face-detector)

FirebaseVisionFaceDetectorOptions options =

       new FirebaseVisionFaceDetectorOptions.Builder()

               .setPerformanceMode(FirebaseVisionFaceDetectorOptions.*FAST*)

               //.setContourMode(FirebaseVisionFaceDetectorOptions.ALL\_CONTOURS)

               //.setLandmarkMode(FirebaseVisionFaceDetectorOptions.ALL\_LANDMARKS)

               .setClassificationMode(FirebaseVisionFaceDetectorOptions.*ALL\_CLASSIFICATIONS*)

               //.enableTracking()

.build();

1. Disable mFaceButton and get an instance of [FirebaseVisionFaceDetector](https://firebase.google.com/docs/reference/android/com/google/firebase/ml/vision/face/FirebaseVisionFaceDetector):

mFaceButton.setEnabled(false);

FirebaseVisionFaceDetector detector = FirebaseVision.*getInstance*().getVisionFaceDetector(options);

1. Finally, pass the image to the detectInImage method [ignore error]:

detector.detectInImage(image)

       .addOnSuccessListener(

               new OnSuccessListener<List<FirebaseVisionFace>>() {

                   @Override

                   public void onSuccess(List<FirebaseVisionFace> faces) {

                       mFaceButton.setEnabled(true);

                       processFaceContourDetectionResult(faces);

                   }

               })

       .addOnFailureListener(

               new OnFailureListener() {

                   @Override

                   public void onFailure(@NonNull Exception e) {

                       // Task failed with an exception

                       mFaceButton.setEnabled(true);

                       e.printStackTrace();

                   }

               });

1. Copy the following method into **MainActivity** class:

private void processFaceContourDetectionResult(List<FirebaseVisionFace> faces) {

   // Task completed successfully

   if (faces.size() == 0) {

       showToast("No face found");

       return;

   }

   mGraphicOverlay.clear();

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

       FirebaseVisionFace face = faces.get(i);

       FaceContourGraphic faceGraphic = new FaceContourGraphic(mGraphicOverlay);

       mGraphicOverlay.add(faceGraphic);

       faceGraphic.updateFace(face);

   }

}

1. Run App.

**Barcode scanning (in MainActivity.java)**

1. Inside **onCreate** method, right after mFaceButton.setOnClickedLisntener(...) [Ignore error]:

// Add on click listener

mBarcodeBtn.setOnClickListener(new View.OnClickListener() {

   @Override

   public void onClick(View view) {

       if(mSelectedImage != null)

           runBarcodeDetection();

       else

           showToast("Please load image first");

   }

});

1. Create a method called runBarcodeDetection:

private void runBarcodeDetection() {

// ...

}

1. Inside **runBarcodeDetection** method (step 17-21), create a FirebaseVisionImage object from a Bitmap:

FirebaseVisionImage image = FirebaseVisionImage.*fromBitmap*(mSelectedImage);

1. Then, configure barcode detector: (Refer: https://firebase.google.com/docs/ml-kit/android/read-barcodes#1.-configure-the-barcode-detector)

FirebaseVisionBarcodeDetectorOptions options =

       new FirebaseVisionBarcodeDetectorOptions.Builder()

               .setBarcodeFormats(

                       FirebaseVisionBarcode.*FORMAT\_QR\_CODE*,

                       FirebaseVisionBarcode.*FORMAT\_AZTEC*)

               .build();

1. Get an instance of [FirebaseVisionBarcodeDetector](https://firebase.google.com/docs/reference/android/com/google/firebase/ml/vision/barcode/FirebaseVisionBarcodeDetector):

FirebaseVisionBarcodeDetector detector = FirebaseVision.*getInstance*()

   .getVisionBarcodeDetector(options);

1. Finally, pass the image to the detectInImage method:

Task<List<FirebaseVisionBarcode>> result = detector.detectInImage(image)

       .addOnSuccessListener(new OnSuccessListener<List<FirebaseVisionBarcode>>() {

           @Override

           public void onSuccess(List<FirebaseVisionBarcode> barcodes) {

               // Task completed successfully

               // ...

           }

       })

       .addOnFailureListener(new OnFailureListener() {

           @Override

           public void onFailure(@NonNull Exception e) {

               // Task failed with an exception

               // ...

           }

       });

1. Inside **onSuccess(...)** method in Step 20, write the following code. Function: to get information from barcodes. If the barcode is of type URL, launch a web browser to visit the url:

for (FirebaseVisionBarcode barcode: barcodes) {

   Rect bounds = barcode.getBoundingBox();

   Point[] corners = barcode.getCornerPoints();

   int valueType = barcode.getValueType();

   // See API reference for complete list of supported types

   switch (valueType) {

       case FirebaseVisionBarcode.*TYPE\_URL*:

           String title = barcode.getUrl().getTitle();

           String url = barcode.getUrl().getUrl();

           if (!url.startsWith("http://") && !url.startsWith("https://"))

               url = "http://" + url;

           Intent browserIntent = new Intent(Intent.*ACTION\_VIEW*, Uri.*parse*(url));

           startActivity(browserIntent);

           break;

        default:

            showToast("The barcode is not an URL");

            break;

   }

}

1. Run App.