Case Report
RoadIntel App

Version 1.0 approved

Prepared by:

Sidharth Jindal

Rohith Reddy

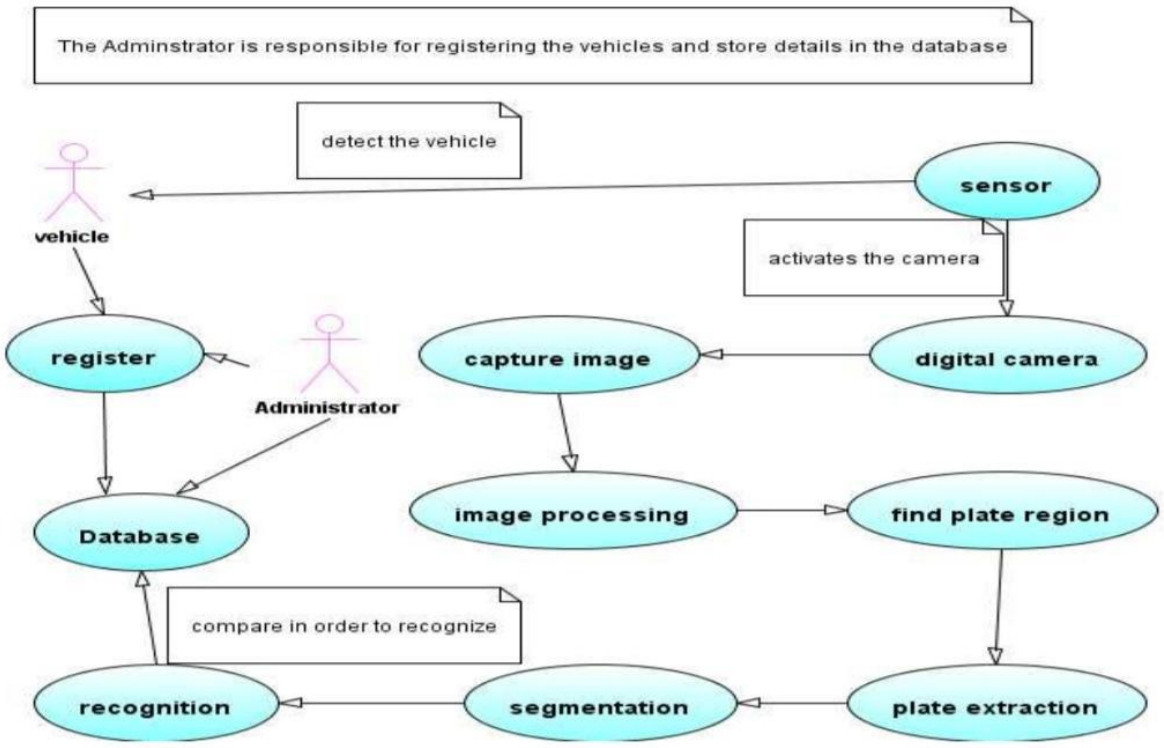
Yash

Manipal Institute of Technology, Manipal

16/04/2019

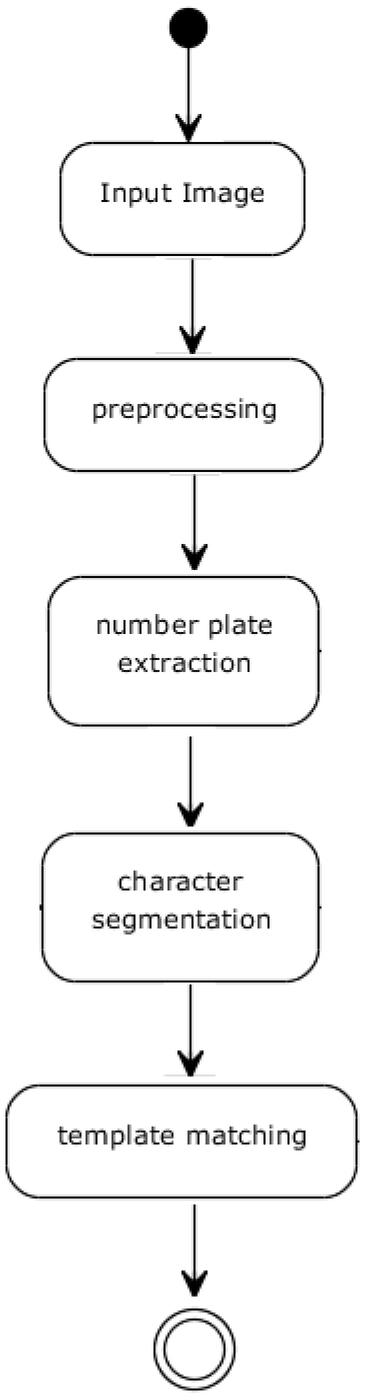
;

I Use Case:

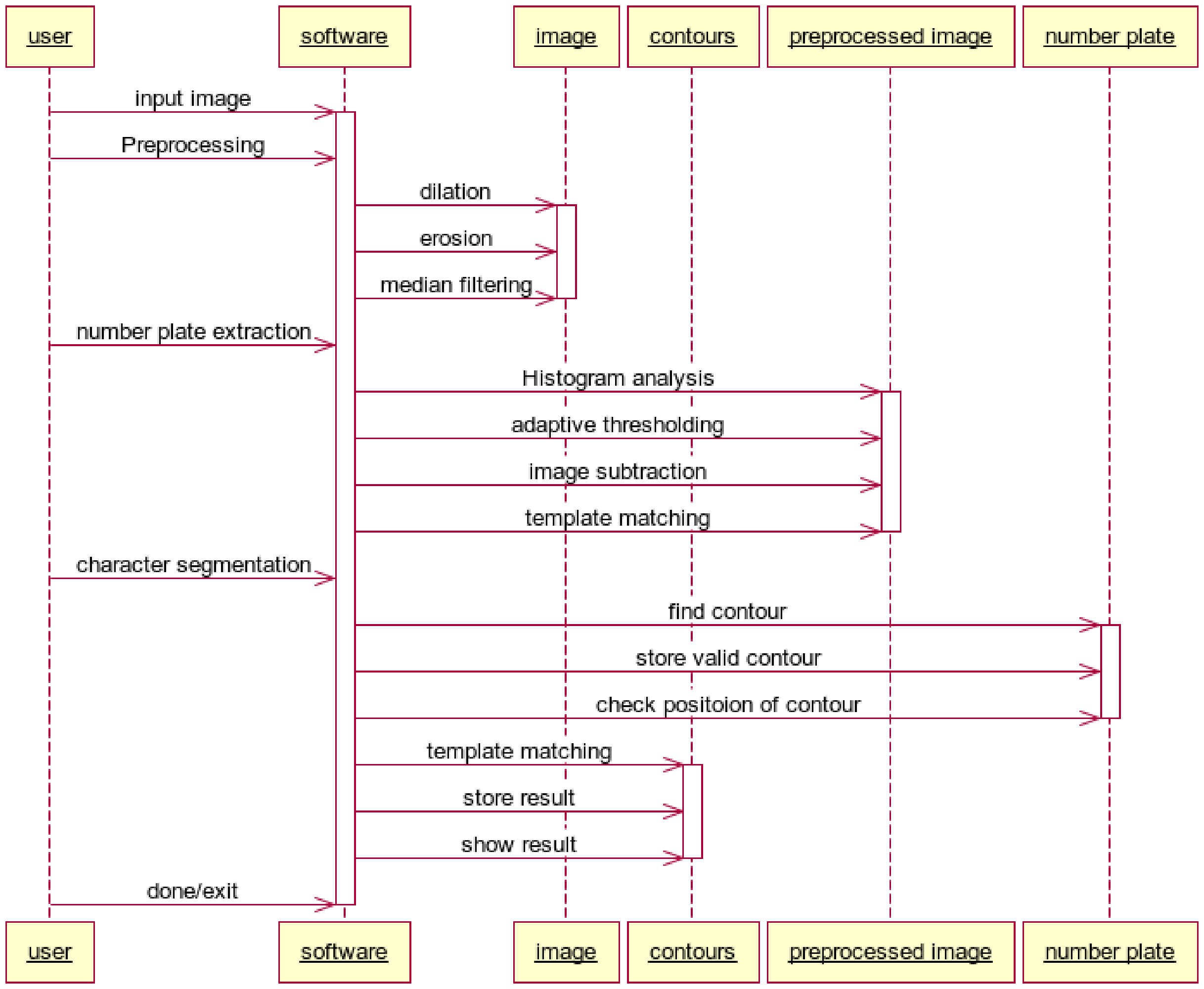

The Adminstrator is responsible for registering the vehicles and store details in the database
detect the vehicle
vehicle
sensor
activates the camera
register
Administrator
capture image
digital camera
Database
image processing
find plate region
compare in order to recognize
recognition
segmentation
plate extraction

II Activity Diagram

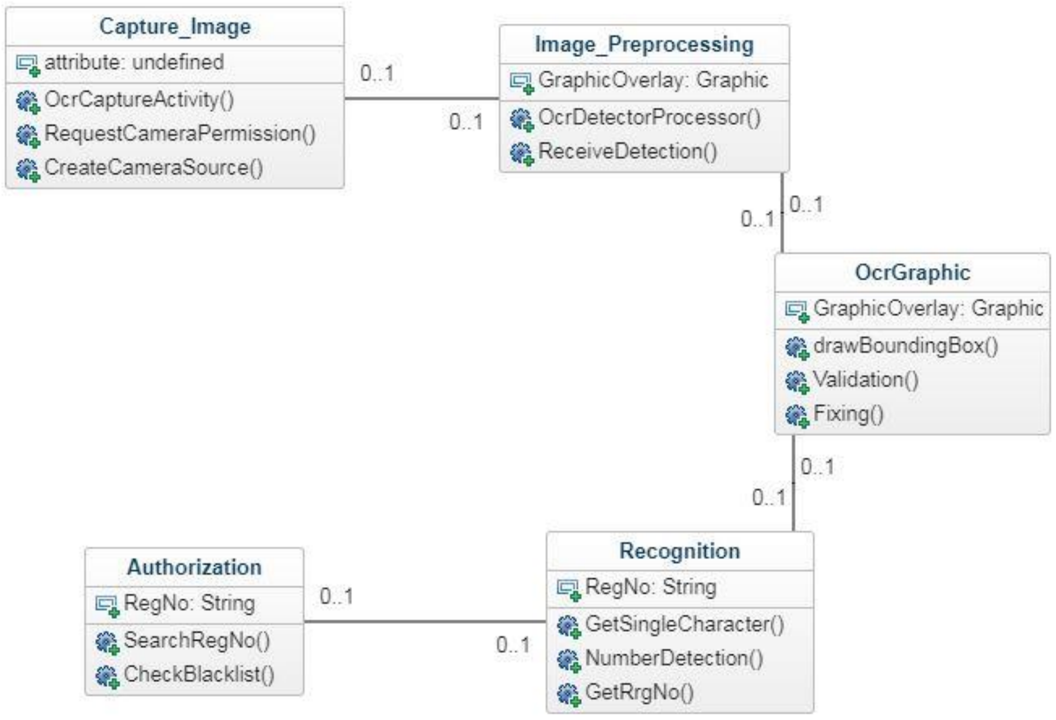
.



III Sequence Diagram:



IV Class Diagram:



V Code:

```
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.hardware.Camera;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.Snackbar;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Toast;
import android.widget.ToggleButton;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GoogleApiAvailability;
import
com.google.android.gms.samples.vision.ocrreader.ui.camera.CameraSource;
import
com.google.android.gms.samples.vision.ocrreader.ui.camera.CameraSourcePrev
iew;
import
com.google.android.gms.samples.vision.ocrreader.ui.camera.GraphicOverlay;
import com.google.android.gms.vision.text.TextRecognizer;

import java.io.IOException;

public final class OcrCaptureActivity extends AppCompatActivity {
    private static final String TAG = "OcrCaptureActivity";
    Context context;
    // Intent request code to handle updating play services if needed.
    private static final int RC_HANDLE_GMS = 9001;

    // Permission request codes need to be < 256
    private static final int RC_HANDLE_CAMERA_PERM = 2;
    // Constants used to pass extra data in the intent
    public static final String AutoFocus = "AutoFocus";
    public static final String UseFlash = "UseFlash";
    ToggleButton flash;
    boolean useFlash;

    private CameraSource mCameraSource;
    private CameraSourcePreview mPreview;
    private GraphicOverlay<OcrGraphic> mGraphicOverlay;

    // Helper objects for detecting taps and pinches.
    private ScaleGestureDetector scaleGestureDetector;

    @Override
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.ocr_capture);

        mPreview = (CameraSourcePreview) findViewById(R.id.preview);
        mGraphicOverlay = (GraphicOverlay<OcrGraphic>)
findViewById(R.id.graphicOverlay);
        flash = (ToggleButton)findViewById(R.id.flash);

        flash.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                if(useFlash)

mCameraSource.setFlashMode(Camera.Parameters.FLASH_MODE_OFF);
                else

mCameraSource.setFlashMode(Camera.Parameters.FLASH_MODE_TORCH);
                useFlash = !useFlash;

            }
        });

        // Set good defaults for capturing text.
        boolean autoFocus = true;

        // Check for the camera permission before accessing the camera.  If the
        // permission is not granted yet, request permission.
        int rc = ActivityCompat.checkSelfPermission(this,
Manifest.permission.CAMERA);
        if (rc == PackageManager.PERMISSION_GRANTED) {
            createCameraSource(autoFocus, false); //false is off by default, the
button use useFlash to toggle the flash
        } else {
            requestCameraPermission();
        }

        scaleGestureDetector = new ScaleGestureDetector(this, new
ScaleListener());

        Snackbar.make(mGraphicOverlay, "Capture number plate. Pinch/Stretch to
zoom",
                Snackbar.LENGTH_LONG)
                .show();
    }

    /**
     * Handles the requesting of the camera permission.  This includes
     * showing a "Snackbar" message of why the permission is needed then
     * sending the request.
     */
    private void requestCameraPermission() {
        Log.w(TAG, "Camera permission is not granted. Requesting permission");

        final String[] permissions = new String[]{Manifest.permission.CAMERA};

        if (!ActivityCompat.shouldShowRequestPermissionRationale(this,
                Manifest.permission.CAMERA)) {
            ActivityCompat.requestPermissions(this, permissions,
RC_HANDLE_CAMERA_PERM);
            return;
        }

        final Activity thisActivity = this;

        OnClickListener listener = new OnClickListener() {
            @Override
            public void onClick(View view) {
                ActivityCompat.requestPermissions(thisActivity, permissions,
                        RC_HANDLE_CAMERA_PERM);
            }
        };

        Snackbar.make(mGraphicOverlay, R.string.permission_camera_rationale,
                Snackbar.LENGTH_INDEFINITE)
                .setAction(R.string.ok, listener)
                .show();
    }

    @Override
    public boolean onTouchEvent(MotionEvent e) {
        boolean b = scaleGestureDetector.onTouchEvent(e);


        return b || super.onTouchEvent(e);
    }

    /**
     * Creates and starts the camera.  Note that this uses a higher resolution in
comparison
     * to other detection examples to enable the ocr detector to detect small text
samples
     * at long distances.
     *
     * Suppressing InlinedApi since there is a check that the minimum version is
met before using
     * the constant.
     */
    @SuppressLint("InlinedApi")
    private void createCameraSource(boolean autoFocus, boolean useFlash) {
        context = getApplicationContext();

        // A text recognizer is created to find text.  An associated multi-processor
instance
        // is set to receive the text recognition results, track the text, and maintain
        // graphics for each text block on screen.  The factory is used by the multi-
processor to
        // create a separate tracker instance for each text block.
        TextRecognizer textRecognizer = new
TextRecognizer.Builder(context).build();
        OcrDetectorProcessor ocrDetectorProcessor =  new
OcrDetectorProcessor(mGraphicOverlay);
        ocrDetectorProcessor.saveContext(OcrCaptureActivity.this);
        textRecognizer.setProcessor(ocrDetectorProcessor);
        if (!textRecognizer.isOperational()) {
            Log.w(TAG, "Detector dependencies are not yet available.");
            IntentFilter lowstorageFilter = new
IntentFilter(Intent.ACTION_DEVICE_STORAGE_LOW);
            boolean hasLowStorage = registerReceiver(null, lowstorageFilter) !=
null;

            if (hasLowStorage) {
                Toast.makeText(this, R.string.low_storage_error,
Toast.LENGTH_LONG).show();
                Log.w(TAG, getString(R.string.low_storage_error));
            }
        }

        mCameraSource =
                new CameraSource.Builder(getApplicationContext(), textRecognizer)
                        .setFacing(CameraSource.CAMERA_FACING_BACK)
                        .setRequestedPreviewSize(800, 600)
                        .setRequestedFps(30.0f)
                        .setFlashMode(useFlash ?
Camera.Parameters.FLASH_MODE_TORCH : null)
                        .setFocusMode(autoFocus ?
Camera.Parameters.FOCUS_MODE_CONTINUOUS_PICTURE : null)
                        .build();
    }

    /**
     * Restarts the camera.
     */
    @Override
    protected void onResume() {
        super.onResume();
        startCameraSource();
    }

    /**
     * Stops the camera.
     */
    @Override
    protected void onPause() {
        super.onPause();
        if (mPreview != null) {
            mPreview.stop();
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (mPreview != null) {
            mPreview.release();
        }
    }    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           @NonNull String[] permissions,
                                           @NonNull int[] grantResults) {
        if (requestCode != RC_HANDLE_CAMERA_PERM) {
            Log.d(TAG, "Got unexpected permission result: " + requestCode);
            super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
            return;
        }

        if (grantResults.length != 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            Log.d(TAG, "Camera permission granted - initialize the camera
source");
            // we have permission, so create the camerasource
            boolean autoFocus = getIntent().getBooleanExtra(AutoFocus,false);
            boolean useFlash = getIntent().getBooleanExtra(UseFlash, false);
            createCameraSource(autoFocus, useFlash);
            return;
        }

        Log.e(TAG, "Permission not granted: results len = " + grantResults.length
+
                " Result code = " + (grantResults.length > 0 ? grantResults[0] :
"(empty)"));

        DialogInterface.OnClickListener listener = new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                finish();
            }
        };
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Multitracker sample")
                .setMessage(R.string.no_camera_permission)
                .setPositiveButton(R.string.ok, listener)
                .show();
    }

    /**
     * Starts or restarts the camera source, if it exists.  If the camera source doesn't
exist yet
     * (e.g., because onResume was called before the camera source was created),
this will be called
     * again when the camera source is created.
     */
    private void startCameraSource() throws SecurityException {
        // check that the device has play services available.
        int code =
GoogleApiAvailability.getInstance().isGooglePlayServicesAvailable(
                getApplicationContext());
        if (code != ConnectionResult.SUCCESS) {
            Dialog dlg =
                    GoogleApiAvailability.getInstance().getErrorDialog(this, code,
RC_HANDLE_GMS);
            dlg.show();
        }

        if (mCameraSource != null) {
            try {
                mPreview.start(mCameraSource, mGraphicOverlay);
            } catch (IOException e) {
                Log.e(TAG, "Unable to start camera source.", e);
                mCameraSource.release();
                mCameraSource = null;
            }
        }
    }

    private class ScaleListener implements
ScaleGestureDetector.OnScaleGestureListener {
        @Override
        public boolean onScale(ScaleGestureDetector detector) {
            return false;
        }        @Override
        public boolean onScaleBegin(ScaleGestureDetector detector) {
            return true;
        }        @Override
        public void onScaleEnd(ScaleGestureDetector detector) {
            if (mCameraSource != null) {
                mCameraSource.doZoom(detector.getScaleFactor());
            }
        }
    }
}
```

```
package com.google.android.gms.samples.vision.ocrreader;

import android.content.Context;
import android.util.Log;
import android.util.SparseArray;

import
com.google.android.gms.samples.vision.ocrreader.ui.camera.GraphicOverlay;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.text.TextBlock;

/**
 * A very simple Processor which gets detected TextBlocks and adds them to the
overlay
```

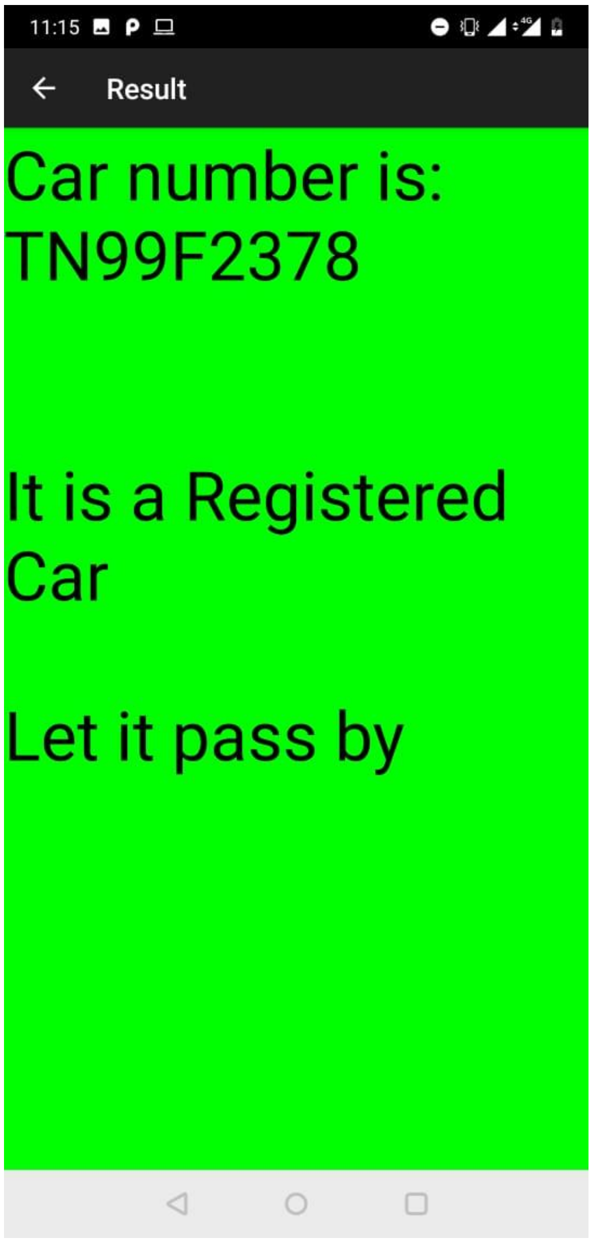
```
 * as OcrGraphics.
 */
public class OcrDetectorProcessor implements Detector.Processor<TextBlock>
{

    private GraphicOverlay<OcrGraphic> mGraphicOverlay;
    private Context context ;
    OcrDetectorProcessor(GraphicOverlay<OcrGraphic> ocrGraphicOverlay) {
        mGraphicOverlay = ocrGraphicOverlay;
    }

    /**
     * Called by the detector to deliver detection results.
     * If your application called for it, this could be a place to check for
     * equivalent detections by tracking TextBlocks that are similar in location
and content from
     * previous frames, or reduce noise by eliminating TextBlocks that have not
persisted through
     * multiple detections.
     */
    @Override
    public void receiveDetections(Detector.Detections<TextBlock> detections) {
        mGraphicOverlay.clear();
        SparseArray<TextBlock> items = detections.getDetectedItems();
        for (int i = 0; i < items.size(); ++i) {
            TextBlock item = items.valueAt(i);
            if (item != null && item.getValue() != null) {
                Log.d("OcrDetectorProcessor", "Text detected! " + item.getValue());
            }
            OcrGraphic graphic = new OcrGraphic(mGraphicOverlay, item);
            //mGraphicOverlay.saveContext(context);
            graphic.saveContext(context);
            mGraphicOverlay.add(graphic);
        }
    }

    /**
     * Frees the resources associated with this detection processor.
     */
    @Override
    public void release() {
        mGraphicOverlay.clear();
    }
    public void saveContext(Context con){
        context = con;
    }
}

/*
 * Copyright (C) The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.google.android.gms.samples.vision.ocrreader;
import android.content.Context;
import android.content.Intent;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.RectF;

import
com.google.android.gms.samples.vision.ocrreader.ui.camera.GraphicOverlay;
import com.google.android.gms.vision.text.TextBlock;

import java.util.regex.Matcher;
import java.util.regex.Pattern;
/**
 * Graphic instance for rendering TextBlock position, size, and ID within an
associated graphic
 * overlay view.
 */
class OcrGraphic extends GraphicOverlay.Graphic {
    private static final int TEXT_COLOR = Color.WHITE;

    private static Paint sRectPaint;
    private static Paint sTextPaint;
    private final TextBlock mText;
    public String text;
    Context context;
    Intent myIntent;

    OcrGraphic(GraphicOverlay overlay, TextBlock text) {
        super(overlay);

        mText = text;

        if (sRectPaint == null) {
            sRectPaint = new Paint();
            sRectPaint.setColor(TEXT_COLOR);
            sRectPaint.setStyle(Paint.Style.STROKE);
            sRectPaint.setStrokeWidth(4.0f);
        }

        if (sTextPaint == null) {
            sTextPaint = new Paint();
            sTextPaint.setColor(TEXT_COLOR);
            sTextPaint.setTextSize(30.0f);
        }
        // Redraw the overlay, as this graphic has been added.
        postInvalidate();
    }

    public void saveContext(Context con){
        context = con;
    }
    /**
     * Draws the text block annotations for position, size, and raw value on the
supplied canvas.
     */
    @Override
    public void draw(Canvas canvas) {
        if (mText == null) {
            return;
        }
        // Draws the bounding box around the TextBlock.
        RectF rect = new RectF(mText.getBoundingBox());
        rect.left = translateX(rect.left);
        rect.top = translateY(rect.top);
        rect.right = translateX(rect.right);
        rect.bottom = translateY(rect.bottom);
        canvas.drawRect(rect, sRectPaint);

        //validation setting
        String REGEX = "^[A-Z]{1,4}\\s*[0-9]{0,3}\\s*[A-Z]{1,2}\\s*[0-
9]{2,4}$"; //regular expression
        Pattern number; //a pattern of compiled regex
        Matcher matcher; //helps in matching the regex
        text = mText.getValue();

        //fixing
        Matcher m = Pattern.compile("[-][0-9]{2}[-]|[-]|[\n]").matcher(text);
        text = m.replaceAll("");
        m = Pattern.compile("IND").matcher(text);
        text = m.replaceAll("");

        //final touch
        text = Pattern.compile("\\s[0-9]{2}\\s").matcher(text).replaceAll("");
        text = text.replaceAll("( +)", "").trim();

        //number detection
        number = Pattern.compile(REGEX);
        matcher = number.matcher(text);
        if (matcher.matches()) { //print if valid
            canvas.drawText(text, rect.centerX(), rect.bottom, sTextPaint); //draw on
screen

            myIntent = new Intent(context, Result.class);
            myIntent.putExtra("result", text); //Optional parameters
            context.startActivity(myIntent);
            //setting up the intent and passing data from this Ocr activity to Result
Activity

            /*
            NOTE:   After a long time searching the web about the problem, I think
the problem is with intent because this file
                    is NOT an Activity file, the activity file is OcrCaptureActivity.
There's a possibility that this might be a
```

```
                    problem. Also, you may need to check the .xml files, I had a hard
time with those.
             */

        }
    }
}


package com.google.android.gms.samples.vision.ocrreader;
import android.content.Context;
import android.os.Build;
import android.os.Vibrator;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.LinearLayout;
import android.widget.TextView;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

import java.io.IOException;
import java.util.ArrayList;


public class Result extends AppCompatActivity {
    Intent intent;
    String value;
    TextView text;
    ArrayList<String> members = new ArrayList<String>();
    ArrayList<String> blacklist = new ArrayList<String>();

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

        intent = getIntent();
        value = intent.getStringExtra("result"); //if it's a string you stored.
        text = (TextView) findViewById(R.id.textView);
        members.add("TN99F2378");
        members.add("DL49AK49");
        blacklist.add("MH20EJ0365");
        blacklist.add("MH12DE1433");
        blacklist.add("MH20V314");
        blacklist.add("DL2CJ1459");
        blacklist.add("DL3CC0524");
        blacklist.add("DL5SM2443");
        blacklist.add("HR38G6020");
        blacklist.add("MH01EA6837");
        blacklist.add("MHO1EA6837");


        String result;

        try {
            result = "Car number is: "+value+"\n\n\n"+find(value);
        }
        catch (Exception e) {
            result = "";
        }

        if(result.isEmpty() || result == null)
            text.setText("Sorry no record found");
        else
            text.setText( result );
    }
    //function for searching the number plate
    public String find(String number) throws IOException {

        StringBuffer string = new StringBuffer();

        if(members.contains(number))
        {
            string.append("It is a Registered Car\n\nLet it pass by");
            LinearLayout l= (LinearLayout) findViewById(R.id.ll);
            l.setBackgroundColor(Color.GREEN);
            return string.toString();


        }
```

```
        if(blacklist.contains(number))
        {
            Vibrator v = (Vibrator)
getSystemService(Context.VIBRATOR_SERVICE);

                //deprecated in API 26
                v.vibrate(1500);
                for(int j=0;j<100000000;j++);
                v.vibrate(1500);

            string.append("It is a blacklisted car\n\nDeport ASAP");
            LinearLayout l= (LinearLayout) findViewById(R.id.ll);
            l.setBackgroundColor(Color.RED);
            return string.toString();

        }



        string.append("Not found in Records");
        LinearLayout l= (LinearLayout) findViewById(R.id.ll);
        l.setBackgroundColor(Color.WHITE);
        return string.toString();
    }
}


//Manifest

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.android.gms.samples.vision.ocrreader"
    android:installLocation="auto">

    <uses-feature android:name="android.hardware.camera" />
    <uses-permission android:name="android.permission.VIBRATE"/>

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:fullBackupContent="false"
        android:hardwareAccelerated="true"
        android:icon="@drawable/icon"
        android:label="RoadwayIntel"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat">
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
        <meta-data
            android:name="com.google.android.gms.vision.DEPENDENCIES"
            android:value="ocr" />

        <activity
            android:name=".OcrCaptureActivity"
            android:label="RoadwayIntel">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:label="@string/title_activity_result"
            android:name=".Result"
            android:parentActivityName=".OcrCaptureActivity">
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.INFO" />
            </intent-filter>
        </activity>
    </application>

</manifest>



//Activity_result

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:id="@+id/ll"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.google.android.gms.samples.vision.ocrreader.Result">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

android:textColor="@color/common_google_signin_btn_text_dark_focused"
        android:textSize="48sp" />
</LinearLayout>


//OCRCapture.xml


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

<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/topLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:keepScreenOn="true">

    <ToggleButton

        android:id="@+id/flash"
        android:layout_width="50dp"
        android:layout_height="40dp"
        android:text="Flash"
        android:layout_alignParentRight="true"
        />




<com.google.android.gms.samples.vision.ocrreader.ui.camera.CameraSourcePre
view
        android:id="@+id/preview"
        android:layout_width="match_parent"
        android:layout_height="match_parent">



<com.google.android.gms.samples.vision.ocrreader.ui.camera.GraphicOverlay
        android:id="@+id/graphicOverlay"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />


</com.google.android.gms.samples.vision.ocrreader.ui.camera.CameraSourcePr
eview>




</RelativeLayout>
```

VI Sample UI:



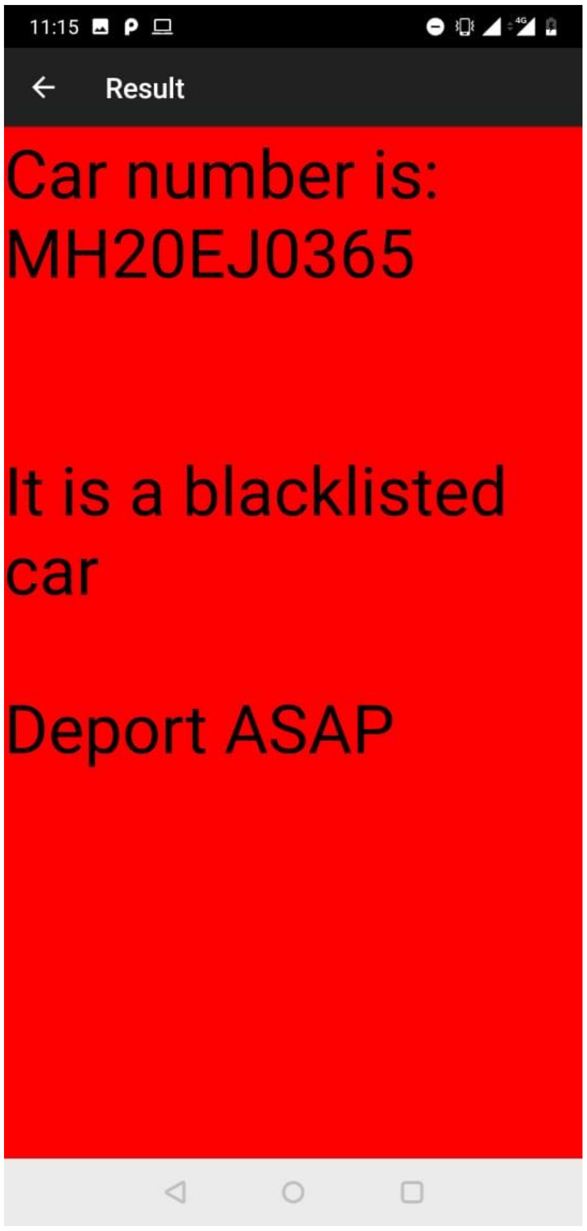
11:15

Result

Car number is: TN99F2378

It is a Registered Car

Let it pass by



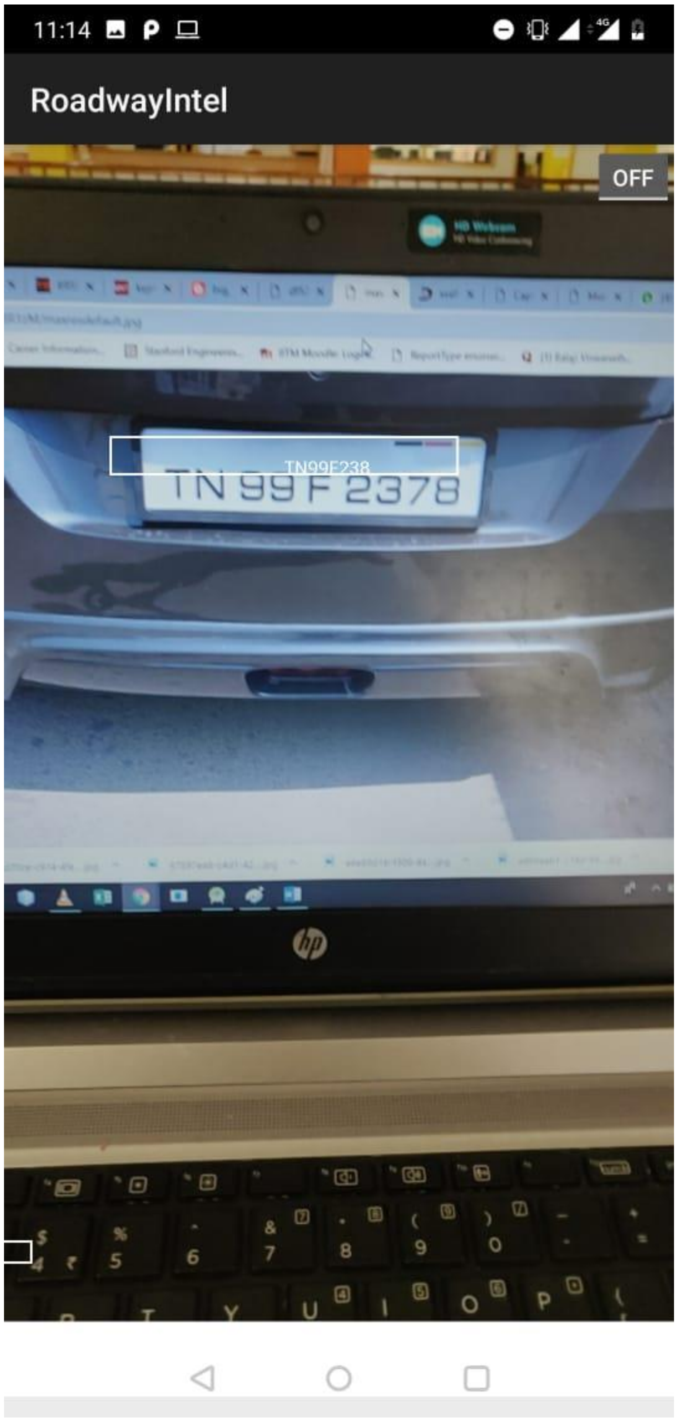
11:15

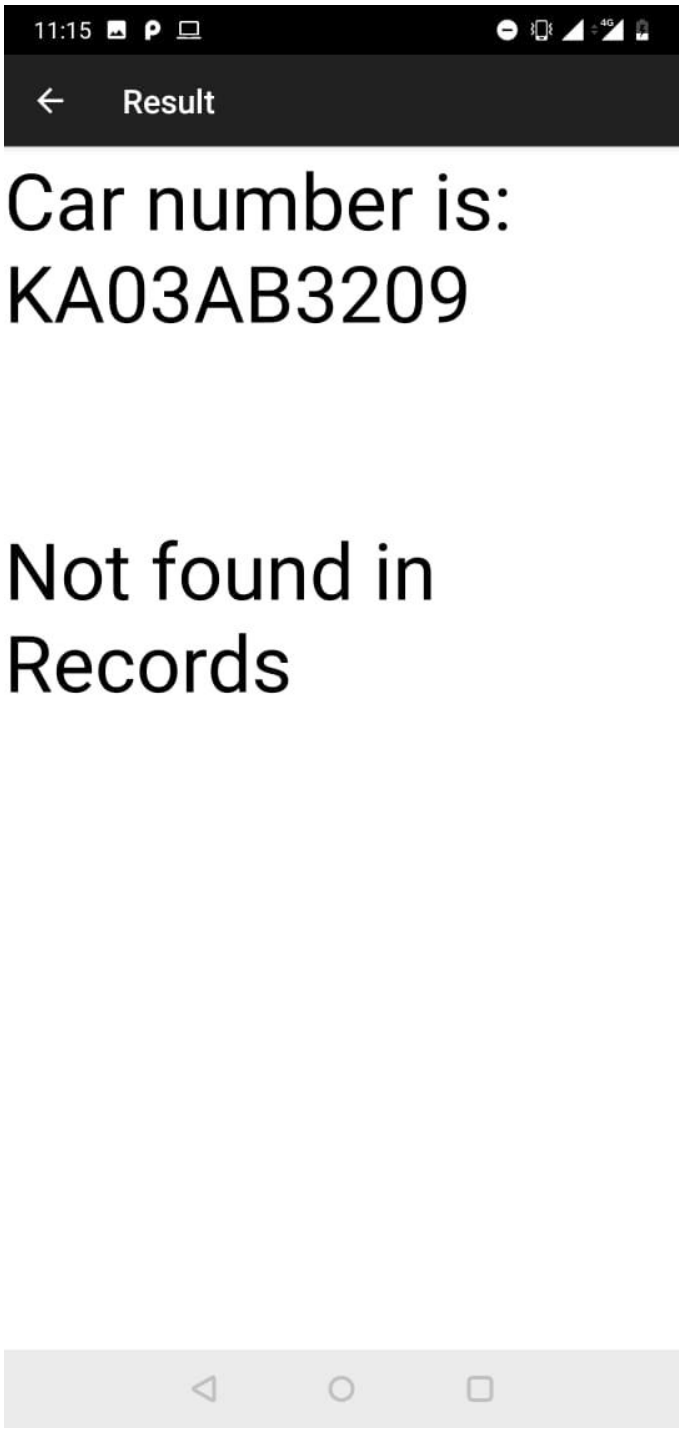
Result

Car number is: MH20EJ0365

It is a blacklisted car

Deport ASAP





Car number is:
KA03AB3209

Not found in
Records