TRANSLIFY: THE TEXT DETECTION AND TRANSLATION APPLICATION

A Minor project report submitted in fulfilment of the requirements for the award of the degree of

B.TECH (CSE) + M.TECH (ICT)

with specialization in

SOFTWARE ENGINEERING

Submitted by:

Radhika Gupta
Registration no.:15/ICS/067

Varnika Gupta
Registration no.:15/ICS/058

Kaavya Singh
Registration no.:15/ICS/075

Supervised by:
Dr. Pradeep Tomar
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY, GAUTAM BUDDHA UNIVERSITY, GREATER NOIDA-201312 GAUTAM BUDDHA NAGAR, UTTAR PRADESH, INDIA.

NOVEMBER, 2018

ACKNOWLEDGEMENTS

It gives us immense pleasure in presenting this project report on "**Translify : The Text Detection and Translation Application**". The writing of this project has been one of the significant academic challenges we have faced and without the support, patience and guidance of the people involved in this project, this would not have been completed. It is to them that we owe our deepest gratitude.

The success of this project is the outcome of sheer hard work and determination put in together by our team with the help of our project mentor and guide.

We hereby take this opportunity to add a special note of thanks to **Dr. Pradeep Tomar.** Without his insight and constant support it would not have been possible to kick start this project and to reach the completion of this report.

____

(Students' Signature)

TABLE OF CONTENTS

1. Introduction

For software products to be effectively usable by an audience, they must be localized, or translated to suit the target user group's culture and language. Multilingual software development is a vast topic and it connects all phases of software development lifecycle and it requires consideration of additional factors such as new roles and linguistic translation of content to one or more languages.

In the running world there is a growing demand for the software systems to recognize characters where information is scanned through paper documents as we know that we have a number of newspapers and books which are in printed formats related to different subjects. Similarly there persists one more demand to get the text translated into desired language. This is how this project came into existence.

2. Theoretical Background

Android

Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touchscreen mobile devices such as smartphones and tablets. [1]
In addition, Google has further developed Android TV for televisions, Android Auto for cars, and Wear OS for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.
The Android SDK (software development kit) is a set of development tools used to develop applications for Android platform. The Android SDK includes the following:

- Required libraries
- Debugger
- An emulator
- Relevant documentation for the Android application program interfaces (APIs)
- Sample source code
- Tutorials for the Android OS

Every time Google releases a new version of Android, a corresponding SDK is also released. To be able to write programs with the latest features, developers must download and install each version's SDK for the particular phone. The development platforms that are compatible with SDK include operating systems like Windows (XP or later), Linux (any recent Linux distribution) and Mac OS X (10.4.9 or later). The components of Android SDK can be downloaded separately. Third party add-ons are also available for download.
Although the SDK can be used to write Android programs in the command prompt, the most common method is by using an integrated development environment (IDE). The recommended IDE is Eclipse with

the Android Development Tools (ADT) plug-in. However, other IDEs, such as NetBeans or IntelliJ, will also work.
Most of these IDEs provide a graphical interface enabling developers to perform development tasks faster. Since Android applications are written in Java code, a user should have the Java Development Kit (JDK) installed.

API

In computer programming, an application programming interface (API) is a set of subroutine definitions, communication protocols, and tools for building software. In general terms, it is a set of clearly defined methods of communication among various components. A good API makes it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer. An API may be for a web-based system, operating system, database system, computer hardware, or software library. An API specification can take many forms, but often includes specifications for routines, data structures, object classes, variables, or remote calls. Windows API and ASPI are examples of different forms of APIs. Documentation for the API usually is provided to facilitate usage and implementation. [2]

Optical Character Recognition

OCR (optical character recognition) is the recognition of printed or written text characters by a computer. This involves photo scanning of the text character-by-character, analysis of the scanned-in image, and then translation of the character image into character codes, such as ASCII, commonly used in data processing as shown in *Figure 2(a)*.
In OCR processing, the scanned-in image or bitmap is analyzed for light and dark areas in order to identify each alphabetic letter or numeric digit. When a character is recognized, it is converted into an ASCII code. Special circuit boards and computer chips designed expressly for OCR are used to speed up the recognition process.
OCR is being used by libraries to digitize and preserve their holdings. OCR is also used to process checks and credit card slips and sort the mail. Billions of magazines and letters are sorted every day by OCR machines, considerably speeding up mail delivery.

Figure 2(a): OCR conversion

Mobile Vision

This application is using Mobile Vision for the text recognition. “Mobile Vision” is an interesting framework by Google *Figure 2(b)*. This framework has the capability to detect objects in photos and videos. This locates and describes objects in images and video frames. [3]

Currently, Android API can detect:

-Faces

-Barcodes

-Texts

Figure 2(b): Google Mobile vision

- Face Detection

The Face API finds human faces in photos, videos, or live streams as shown in *Figure 2(c)*. It also finds and tracks positions of facial landmarks such as the eyes, nose, and mouth.

The API also provides information about the state of facial features. With these technologies, you can edit photos and video, enhance video feeds with effects and decorations, create hands-free controls for games and apps, or react when a person winks or smiles.

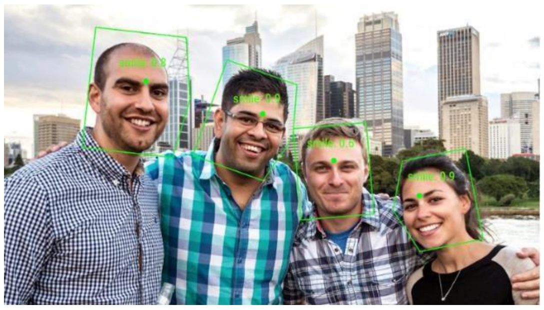


Figure 2(c): Face detection

- Scan Barcodes

The Barcode Scanner API detects barcodes in real time in any orientation. You can also detect and parse several barcodes in different formats at the same time. Refer the *Figure 2(d).*

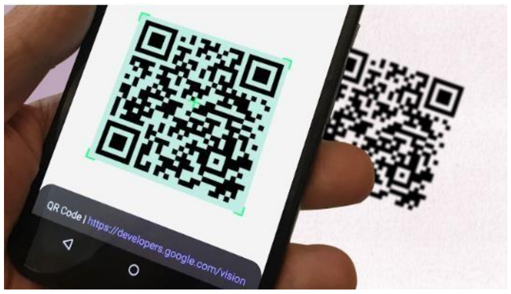


Figure 2(d): Scan barcodes

- Recognize Text

The Text Recognition feature recognizes text in any Latin based language as shown in *Figure 2(d)*.
It also represents the structure of recognized text, including paragraphs and lines.
Text Recognition can automate tedious data entry for credit cards, receipts, and business cards, as well as help organize photos, translate documents, or increase accessibility.
Recognized Languages
The Text feature can recognize text in any Latin based language. This includes, but is not limited to:

- Catalan
- Danish
- Dutch
- English
- Finnish
- French
- German
- Hungarian
- Italian
- Latin
- Norwegian
- Polish
- Portuguese
- Romanian
- Spanish
- Swedish
- Tagalog

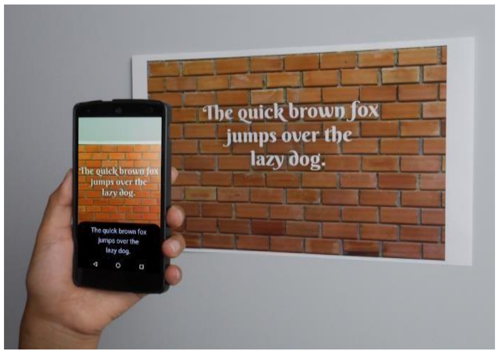

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.

Figure 2(d): Recognize text

Translation API

The Translation API provides a simple programmatic interface for translating an arbitrary string into any supported language.
It is highly responsive, so websites and applications can integrate with Translation API for fast, dynamic translation of source text from the source language to a target language (such as French to English). [4]

Language detection is also available in cases where the source language is unknown.
The underlying technology is updated constantly to include improvements from Google research teams, which results in better translations and new languages and language pairs.

- Helps in Fast and dynamic translation.
- Provides simple programmatic interface for the translating an arbitrary string into any supported language.
- Translates many languages: Translation API supports more than one hundred different languages, from Afrikaans to Zulu. Used in combination, this enables translation between thousands of language pairs.
- Performs language detection: Sometimes you don't know your source text language in advance. For example, user-generated content might not contain a language code. Translation API can automatically identify languages with high accuracy.
- Easy to integrate: Translation API is an easy-to-use Google REST API. You don't have to extract text from your document, just send it HTML documents and get back translated text.

Working

The workflow of the application will be as follows and refer *Figure 2(e)*:-

- Capture Image.
- Detect edges.
- Retrieve text using mobile vision text recognition feature.
- Choosing the desired language to get the text translated in.
- Translating the text using Translation API.
- Getting the translated text.

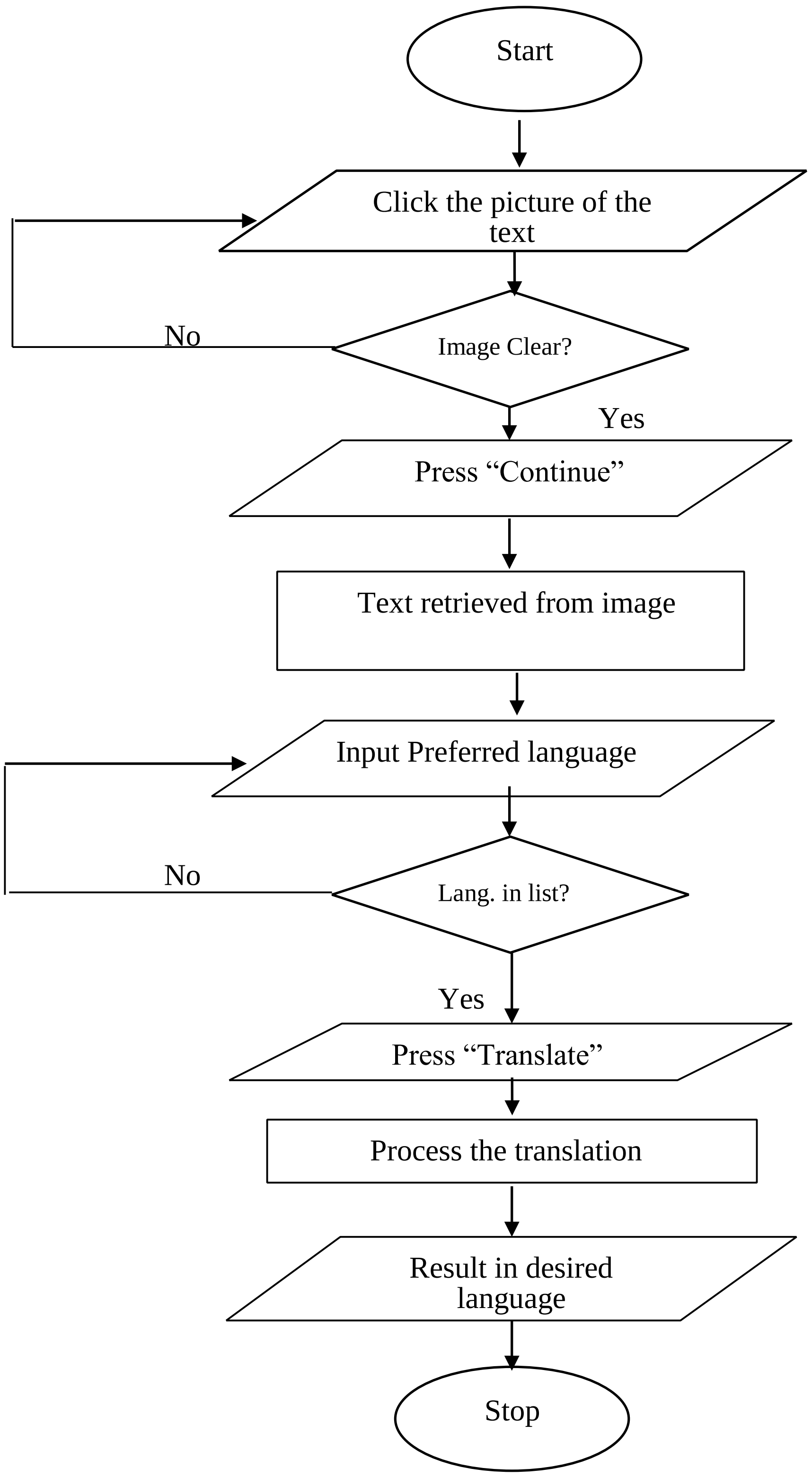


Figure 2(e): Flow chart

3. Design

This includes all the diagrams that help in transforming user requirements into a suitable form which makes the implementation simpler and efficient. It moves the concentration from problem domain to solution domain.

3.1 Data Flow Diagram

The following diagram shows the graphical representation of the flow of data through this information system, modelling its process aspects.

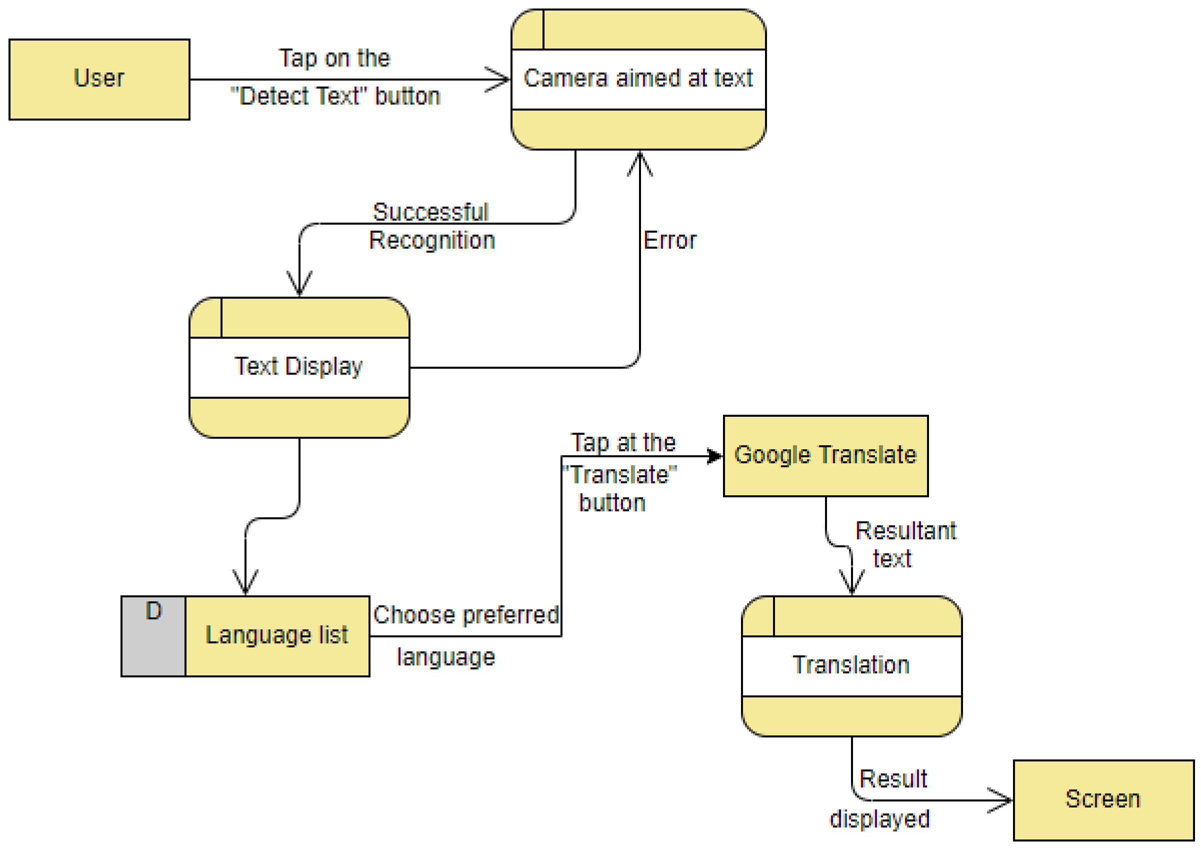


Figure 3(a): Data Flow Diagram

3.2 Sequence Diagram

The following sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

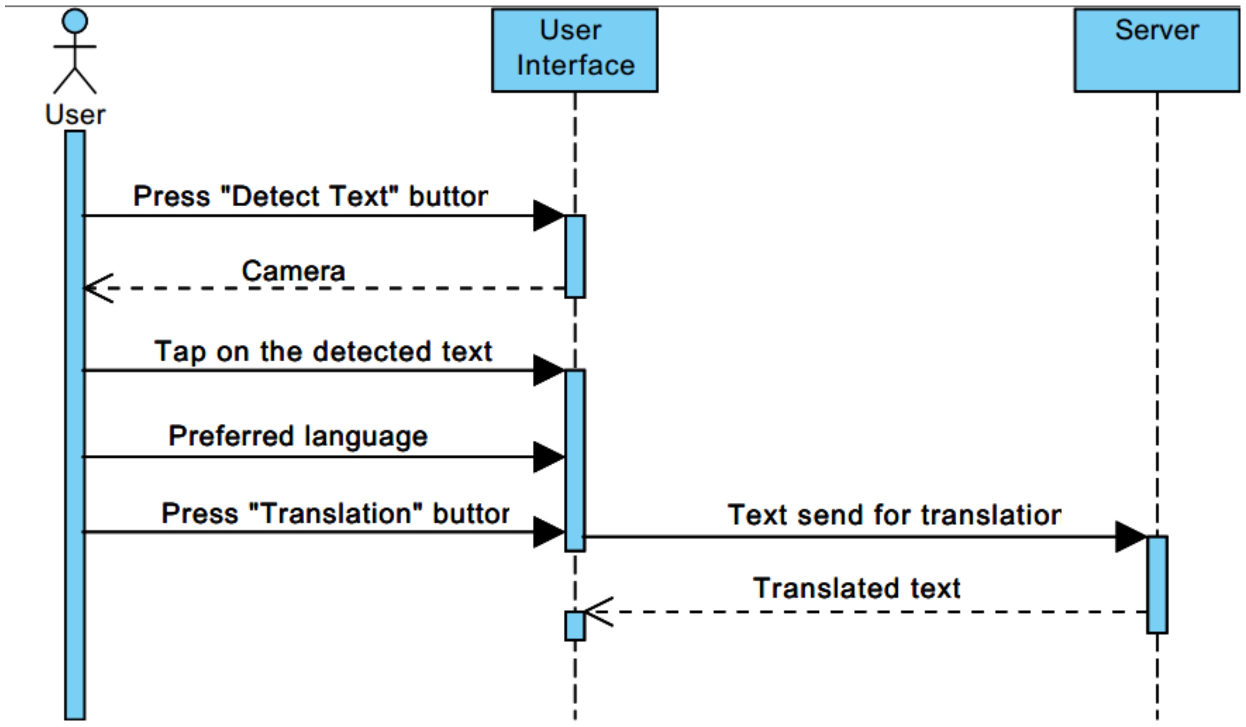
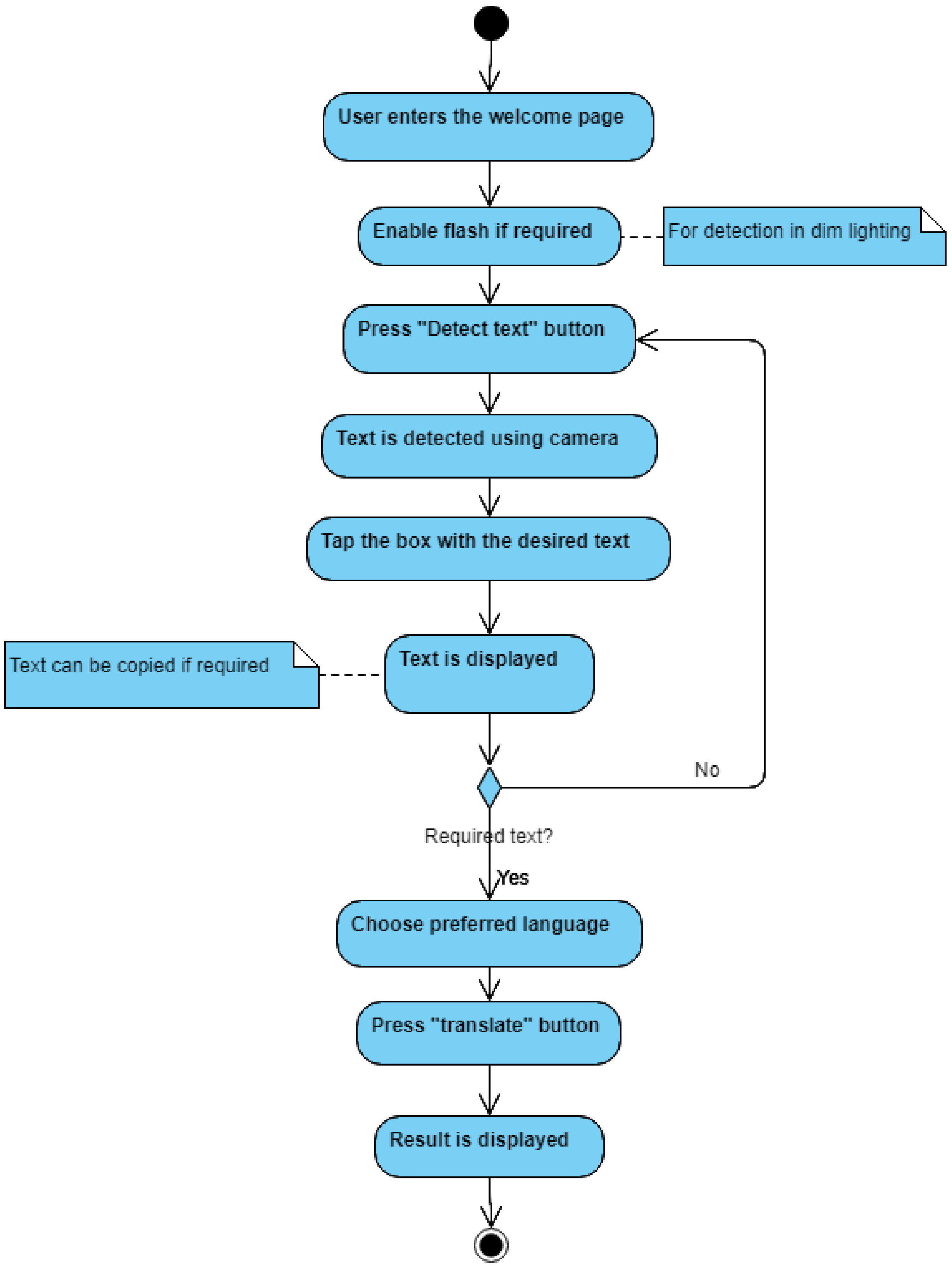


Figure 3(b) : Sequence Diagram

3.3 Activity Diagram

The following activity diagram represents the flow from one activity to another. The activity can be described as an operation of the system. The control flow is drawn from one operation to another.



User enters the welcome page
Enable flash if required
For detection in dim lighting
Press "Detect text" button
Text is detected using camera
Tap the box with the desired text
Text can be copied if required
Text is displayed
No
Required text?
Yes
Choose preferred language
Press "translate" button
Result is displayed

Figure 3(c) : Activity Diagram

Software Requirements Specification

for

Translify: The Text Detection and Translation Application

4.1 Introduction

The main aim of this project is to translate the text that is retrieved from an image through mobile vision. And this document enlists all the hardware, software requirements that would be needed by this project to function smoothly.

4.1.1 Purpose

The application focuses on retrieving the text from images captured by real time camera and then processing it and converting it into text format and further translating it into the desired language. It will recognize the text in the images, boards and hoardings instantly and will give the provision to change it to any preferred language as mentioned by the user.
The main purpose of this application is to perform Document Image Analysis, document formats converted from paper formats more effectively and efficiently.
This improves the accuracy of recognizing the characters during document processing compared to various existing available character recognition methods. The primary objective is to speed up the process of text recognition in document.

4.1.2 Intended Audience and Reading Suggestions

This application aims in helping the tourists and the people who are travelling from one place to other and can't understand the native language of that area. This SRS is the complete synopsis of the idea we want to accomplish. This document holds the workflow and the requirements of the application.

4.1.3 Product Scope

This product majorly finds its scope among users that are in dire need of comprehending the text that they are given to deal with. The main benefit of this application is that it works on android OS platform which is quite prevalent among users, these days.
The scope of our product is to provide an efficient and enhanced software tool for the users to perform Document Image Analysis, document processing by reading and recognizing the text in research and converting it in the desired language.
It supports functionalities such as editing, etc. It also adds benefit by providing heterogeneous characters recognition.

4.2 Overall Description

A detailed explanation of our application is given below.

4.2.1 Product Perspective

It is a self-contained application that is loosely based on the concepts of Optical Character Recognition but most of its formation is done through Mobile Vision API. Refer the image *Figure 3(a).*

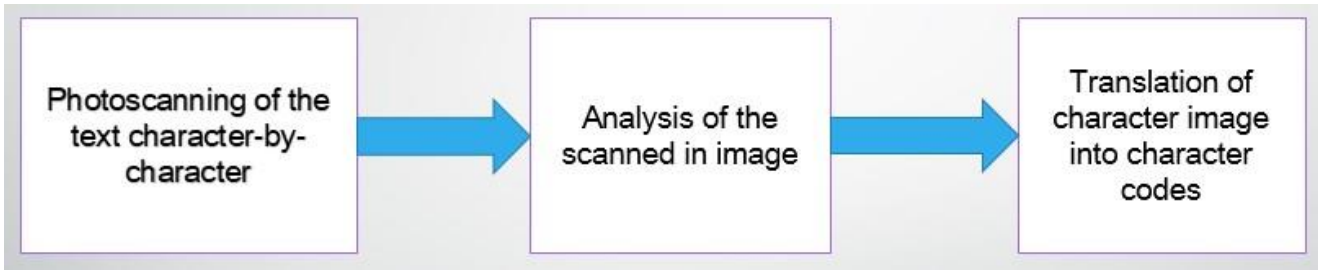

Photoscanning of the text character-by-character
Analysis of the scanned in image
Translation of character image into character codes

Figure 3(a): Concept of the product

4.2.2 Product Functions

Mobile Vision gives developers a powerful and reliable OCR capability that works with most Android devices. Translation API will provide a simple programmatic interface for translating an arbitrary string into any supported language.

4.2.3 User Classes and Characteristics

This will be a very useful app for anyone who lives overseas or travels regularly, and it just got even smarter on mobile. The translation works by translating an entire sentence at a time instead of breaking it first. This allows it to understand the context of a sentence better, resulting in translations that sound more accurate and natural. Another important feature of this application is instant visual translation. It allows users to put the camera integrated with the app over a word or phrase and the translation will be displayed on the phone’s screen, after choosing the desired language.

4.2.4 Operating Environment

Software requirements:

- Android SDK
- Android Studio
- Mobile vision
- Translation API
- Windows Operating system

Hardware requirements:

- Smart phone with a working camera

4.2.5 Design and Implementation Constraints

The application has been developed for Android based devices with high camera's resolution. Also, the printed text without skew is the mainly focus of the application because of time constraint, but it can be added later on.

4.2.5.1 Operating System

The application shall be constrained to the android operating system, and as such no guarantee shall be given for the program being able to run under any other operating system.

4.2.5.2 Graphical User Interface

The application shall be written to be executed on the Android OS. As such the application shall not be optimized for any other operating system.

4.2.5.3 Data Output

The output shall be a visual output that will allow the user to see the text that is to be translated and shall display it translated in the language specified by the user.

4.2.6 Assumptions and Dependencies

- Target platform should be only android smart phones.
- There should be a stable Internet connection.
- Operating system of the device should be up-to-date.
- Smart phone should have modern RAM and CPU.

4.3 External Interface Requirements

This describes the logical and physical characteristics of each interface between the software product and the hardware components of the system.

4.3.1 Hardware Interfaces

For the product to run, you shall need an Android device. To get accurate results, your device should have a high resolution camera. Also, since image processing is resource-hungry, your device should have modern RAM and CPU.
Since neither the mobile application nor the web portal have any designated hardware, it does not have any direct hardware interfaces. The physical camera is managed by the camera interface in the smart phone and the hardware connection to the Internet is managed by the underlying operating system on the mobile phone and the web server.

4.3.2 Software Interfaces

As it has already been stated, product shall run on only Android devices, so it shall not need any specific software from the user's side other than it. The mobile application communicates with the web server in order to get required results.

4.3.3 Communications Interfaces

As it has already been stated, Internet connection is obviously necessary for a complete operation but, you do not need any special configuration for your network adapter.

4.4 Other Non-functional Requirements

Non-functional requirements specify the criteria that can be used to judge the operation of a system.

4.4.1 Performance Requirements

Operation of the application shall not cause other applications to halt/hang or exit prematurely.

Application should be compatible with Android OS.

4.4.2 Security Requirements

This mobile application does not directly deal with any personal information that requires a secure application it will never ask for personal information, in any form.

4.4.3 Software Quality Attributes

Quality Assurance activities are oriented towards prevention of introduction of defects.

4.4.3.1 Availability

This application will be available with a stable internet connection.

4.4.3.2 Maintainability

This application will be guaranteed for a certain Android OS version. Older versions do not have a guarantee. Future updates for complete compatibility with future OS releases are not guaranteed.

4.4.3.3 Transferability/Portability

This app is specifically saved to the device it is installed on, and does not give any cloud storage. Upon reinstalling, no settings are guaranteed to be saved. This application is specifically written for the Android OS, and therefore no plans are made or guaranteed for a port to iOS.

5. Implementation and Results

The implementation results shows the end result of the project. It is an Android based device application that fulfils the requirements set. These are some snapshots of the application with their description.

- The ***Figure 4(a)*** represents the home page of the application.

:

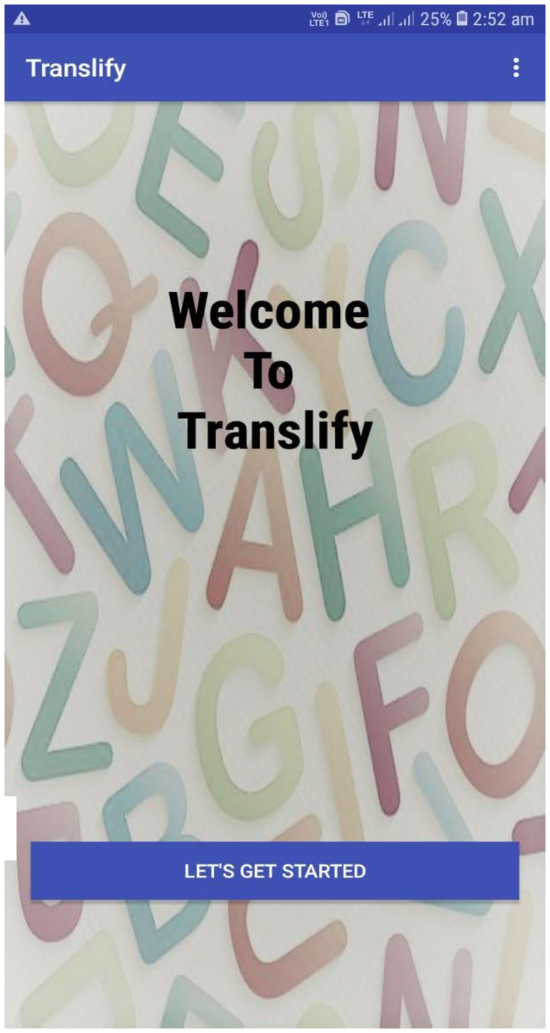


Figure 4(a): Welcome page

- The ***Figure 4(b)*** represents the window from which the user gets the option of text detection through the "DETECT TEXT" button and the resultant machine encoded text is displayed in the text box.

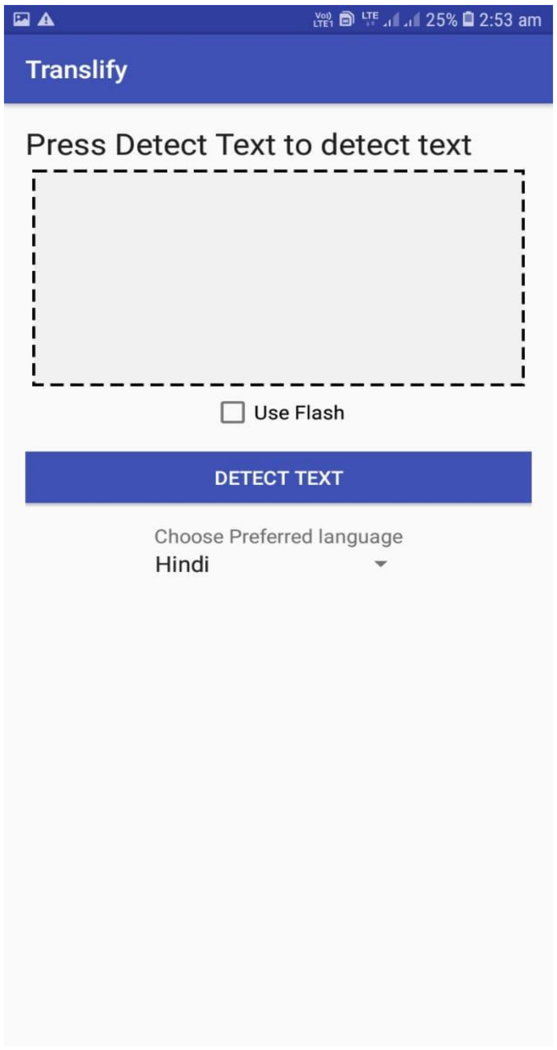


Figure 4(b): Before text detection

- The ***Figure 4(c)*** shows the detection of text through the camera. The user aims the camera at the text which is to be translated. From there only, the user gets suggestions as to what the text is supposed to be and the moment the desired text appears, the user has to tap on the screen.

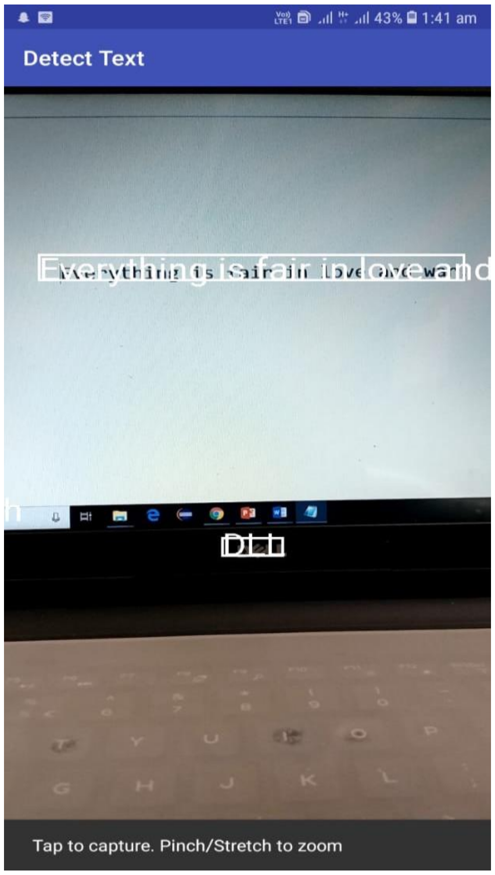
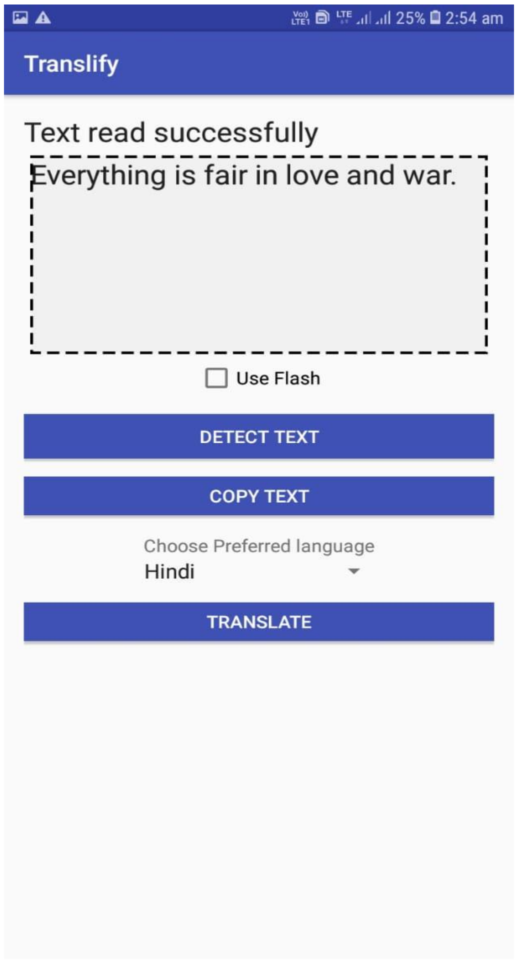

43% 1:41 am
Detect Text
Everything is fair in love and
Tap to capture. Pinch/Stretch to zoom

Figure 4(c): Text Detection

- The ***Figure 4(d)*** shows the detected text in the text box. If this is not the required text, then the user has to repeat the process through the “DETECT TEXT” button.

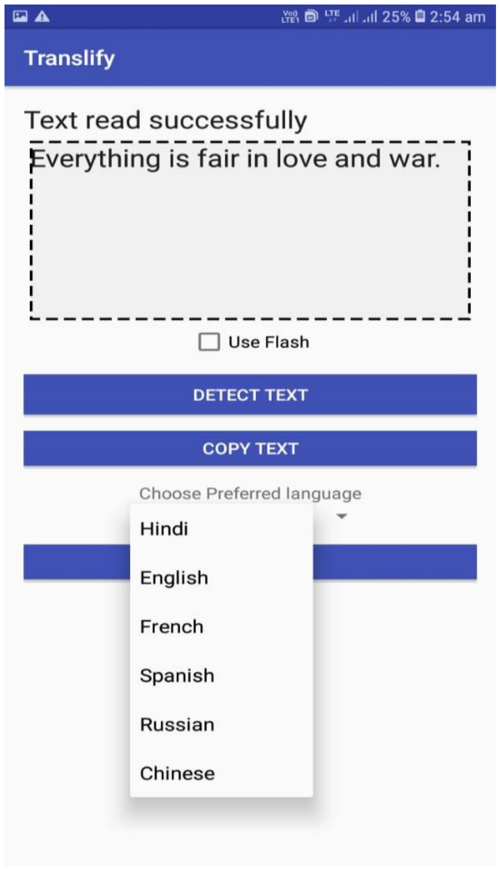


Text read successfully

Everything is fair in love and war.

Figure 4(d): After text detection

- The ***Figure 4(e)*** shows the choices of languages from which the user has to choose the preferred language.

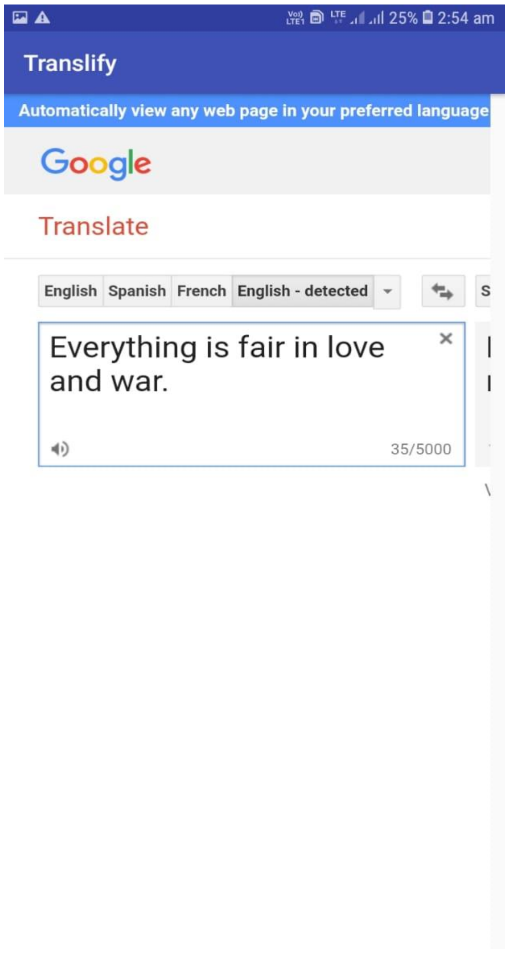


Text read successfully

Everything is fair in love and war.

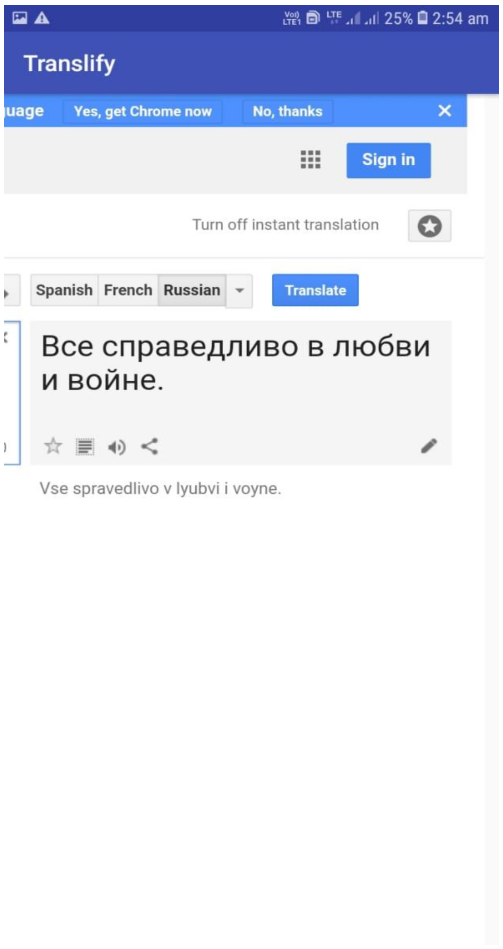
Figure 4(e): Choosing preferred language

- The ***Figure (f)*** *and* ***Figure (g)*** shows the translation of the desired text using Google Translate API.


2:54 am
Translify
Automatically view any web page in your preferred language
Google
Translate
English Spanish French English - detected

Everything is fair in love and war.

Figure (f): Text to be translated


2:54 am
Translify
uage Yes, get Chrome now No, thanks
Sign in
Turn off instant translation
Spanish French Russian
Translate

Все справедливо в любви и войне.

Vse spravedlivo v lyubvi i voyne.

Figure (g): Text after translation

- The ***Figure 4(h) Figure 4(i)*** represents the home screen from where the user gets the "About us" option.

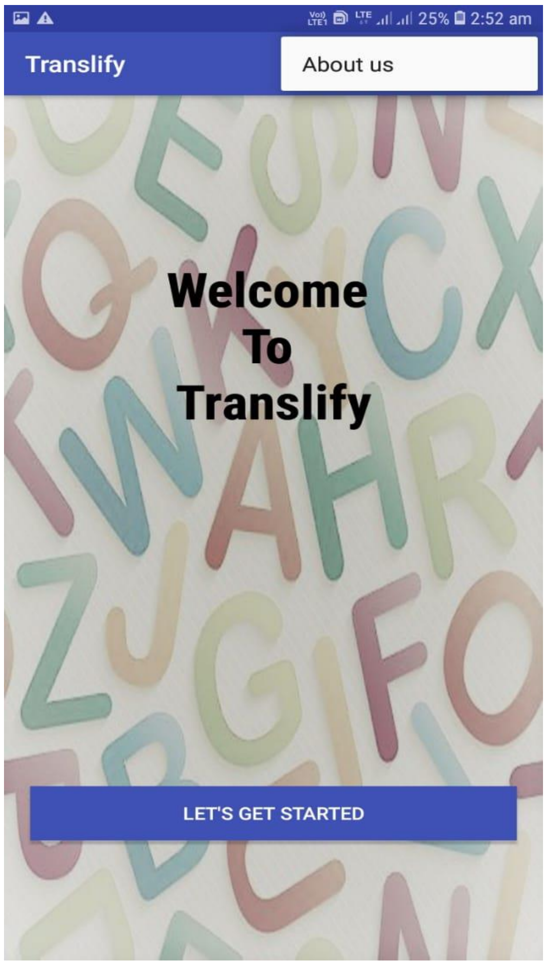
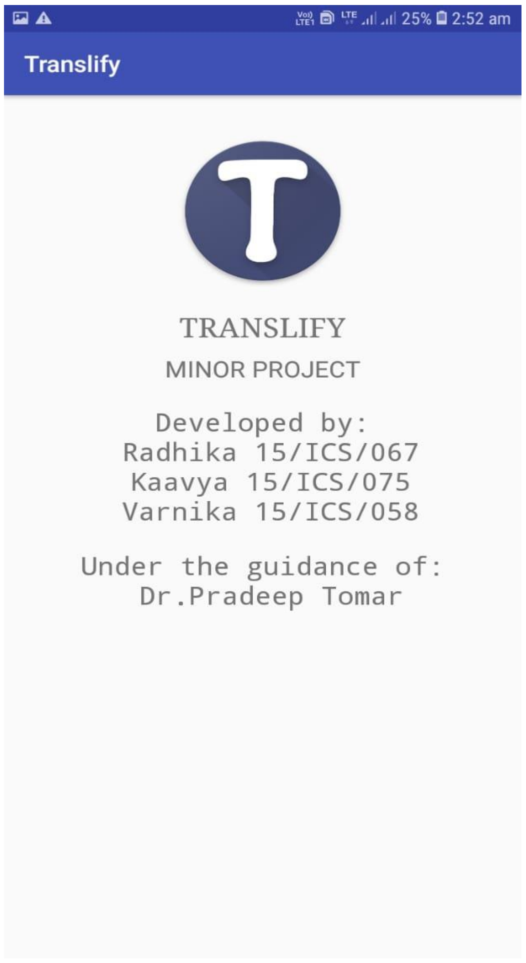


Figure 4(h): "About us" button on the Home screen



TRANSLIFY
MINOR PROJECT

Developed by:
Radhika 15/ICS/067
Kaavya 15/ICS/075
Varnika 15/ICS/058

Under the guidance of:
Dr.Pradeep Tomar

Figure 4(i): About us

6. Scope of

- This app comes handy in places following entirely different linguistic skills in order to facilitate communication for tourists visiting the particular location.
- It instantly recognizes texts. Hence it is highly beneficial in reading street boards and hoardings.
- It deciphers the script written along with road signs as well.

7. Limitations

- Image taken by mobile camera should be of good quality.
- Mobile should be of high specifications.
- For the translation of the extracted text, internet connection is required.

8. References

1. https://www.android.com/
2. https://www.mulesoft.com/resources/api/what-is-an-api
3. https://developers.google.com/vision/
4. https://cloud.google.com/translate/docs/

Appendix: Source Code

Front End

Activity_welcome.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:background="@drawable/a"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.kvr.translify.activities.Welcome"
    android:weightSum="1">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="304dp"
        android:layout_gravity="center_horizontal"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:layout_marginTop="20dp"
        android:layout_marginBottom="60dp"
        android:textStyle="bold"
        android:fontFamily="sans-serif-condensed"
        android:gravity="center"
        android:text="Welcome \nTo \nTranslify"
        android:textColor="@color/common_google_signin_btn_text_dark_focused"
        android:textSize="42dp"
        android:layout_weight="0.48" />


    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="130dp"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"

        android:textAlignment="center"
        android:background="@color/colorPrimary"
        android:text="Let's get started"
        android:textColor="@android:color/white" />
</LinearLayout>
```

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
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:fitsSystemWindows="true"
    tools:context="com.kvr.translify.activities.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
```

```
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main"
        android:layout_height="463dp"
        />

</android.support.design.widget.CoordinatorLayout>
```

content_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    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/content_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scrollbars="none"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:fillViewport="true"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.kvr.translify.activities.MainActivity"
    tools:showIn="@layout/activity_main">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:weightSum="1">

        <TextView
            android:id="@+id/status_message"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Press Detect Text to detect text" />

        <TextView
            android:id="@+id/text_value"
            android:layout_width="match_parent"
            android:layout_height="176dp"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:minHeight="@dimen/text_value_min_height"
            android:layout_margin="@dimen/small_margin"
            android:background="@drawable/dashed_border" />

        <CheckBox
            android:id="@+id/use_flash"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/use_flash"
            android:layout_gravity="center"
            android:checked="false" />

        <Button
            android:id="@+id/read_text_button"
            android:layout_width="match_parent"
            android:layout_height="34dp"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="@dimen/button_margin_top"
            android:background="@color/colorPrimary"
            android:text="Detect Text"
            android:textColor="@android:color/white"
            android:layout_weight="0.02" />

        <Button
            android:id="@+id/copy_text_button"
            android:layout_width="match_parent"
            android:layout_height="34dp"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="@dimen/button_margin_top"
            android:background="@color/colorPrimary"
            android:text="Copy Text"
            android:textColor="@android:color/white" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="186dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/button_margin_top"
            android:layout_gravity="center_horizontal"
            android:text="Choose Preferred language" />

        <Spinner
            android:id="@+id/spinner2"
            android:layout_width="201dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal" />

        <Button
            android:id="@+id/mail_text_button"
            android:layout_width="match_parent"
            android:layout_height="34dp"
            android:layout_marginTop="@dimen/button_margin_top"
            android:background="@color/colorPrimary"
            android:text="Translate"
            android:textColor="@android:color/white" />

    </LinearLayout>
</ScrollView>
```

Activity_translate.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"
    tools:context="com.kvr.translify.activities.TranslateActivity">

 <WebView

        android:layout_width="match_parent"
        android:id="@+id/webView"
        android:layout_height="match_parent"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_marginRight="12dp"
        android:layout_marginEnd="12dp">
 </WebView>

</RelativeLayout>
```

ocr_capture.xml

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

<LinearLayout
```

```xml
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:keepScreenOn="true">

    <com.kvr.translify.ui.camera.CameraSourcePreview
        android:id="@+id/preview"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <com.kvr.translify.ui.camera.GraphicOverlay
            android:id="@+id/graphicOverlay"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

    </com.kvr.translify.ui.camera.CameraSourcePreview>

</LinearLayout>
```

keys.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="keys">
        <item>Hindi</item>
        <item>English</item>
        <item>French</item>
        <item>Spanish</item>
        <item>Russian</item>
        <item>Chinese</item>
        <item>Italian</item>
        <item>Portuguese</item>
        <item>Swedish</item>
        <item>German</item>
        <item>Arabic</item>
        <item>Bengali</item>
    </string-array>
</resources>
```

AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.kvr.translify"
    android:installLocation="auto">

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

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

    <application
        android:allowBackup="true"
        android:fullBackupContent="false"
        android:hardwareAccelerated="true"
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <meta-data
            android:name="com.google.android.gms.vision.DEPENDENCIES"
            android:value="ocr" />

        <activity
            android:name=".activities.OcrCaptureActivity"
            android:label="Detect Text" />
        <activity
            android:name=".activities.Welcome"
            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=".activities.MainActivity"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity android:name=".activities.TranslateActivity" />
        <activity android:name=".activities.activity_aboutus"></activity>
    </application>

</manifest>
```

Back End

Welcome.java

```java
package com.kvr.translify.activities;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.kvr.translify.R;

public class Welcome extends AppCompatActivity {


    public Button  btnCamera;

    public Button abtus;


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu, menu);
        return true;
    }
    public boolean onOptionsItemSelected(MenuItem item) {

      int id=item.getItemId();
        if(id==R.id.aboutus){
            switch (item.getItemId()) {
                case R.id.aboutus:
                    startActivity(new Intent(this, activity_aboutus.class));
                    return true;
                default:
                    return super.onOptionsItemSelected(item);
            }
        }
 return
         super.onOptionsItemSelected(item);
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_welcome);
        btnCamera  = (Button)findViewById(R.id.button1);
        btnCamera.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i= new Intent(Welcome.this, MainActivity.class);
                startActivity(i);
            }
        });

    }
}
```

MainActivity.java

```
package com.kvr.translify.activities;

import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.common.api.CommonStatusCodes;
import com.kvr.translify.R;

import java.util.ArrayList;
import java.util.List;

import static com.kvr.translify.R.id.spinner2;

public class MainActivity extends AppCompatActivity implements
AdapterView.OnItemSelectedListener {

    private CompoundButton useFlash;
    private TextView statusMessage;
    private TextView textValue;
    private Button copyButton;
    private Button mailTextButton;
    public  Spinner spinner2;
    public String keyValue;




    private static final int RC_OCR_CAPTURE = 9003;
    private static final String TAG = "MainActivity";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        statusMessage = (TextView)findViewById(R.id.status_message);
        textValue = (TextView)findViewById(R.id.text_value);
        useFlash = (CompoundButton) findViewById(R.id.use_flash);

        Button readTextButton = (Button) findViewById(R.id.read_text_button);
        readTextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Intent intent = new Intent(getApplicationContext(),
OcrCaptureActivity.class);
                intent.putExtra(OcrCaptureActivity.AutoFocus, true);
                intent.putExtra(OcrCaptureActivity.UseFlash, useFlash.isChecked());

                startActivityForResult(intent, RC_OCR_CAPTURE);
            }
        });

        copyButton = (Button) findViewById(R.id.copy_text_button);
        copyButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(Build.VERSION.SDK_INT < Build.VERSION_CODES.HONEYCOMB) {
                    android.text.ClipboardManager clipboard =
                            (android.text.ClipboardManager)
getSystemService(Context.CLIPBOARD_SERVICE);
                    clipboard.setText(textValue.getText().toString());
                } else {
                    android.content.ClipboardManager clipboard =
                            (android.content.ClipboardManager)
getSystemService(Context.CLIPBOARD_SERVICE);
                    android.content.ClipData clip =
android.content.ClipData.newPlainText("Copied Text", textValue.getText().toString());
                    clipboard.setPrimaryClip(clip);
                }
                Toast.makeText(getApplicationContext(),
R.string.clipboard_copy_successful_message, Toast.LENGTH_SHORT).show();
            }
        });

        mailTextButton = (Button) findViewById(R.id.mail_text_button);
        mailTextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i = new Intent(MainActivity.this, TranslateActivity.class);
                //i.putExtra("link",
"https://translate.google.com/m/translate#view=home&op=translate&sl=auto&tl="+
keyValue.replace("English", "en").replace("Hindi", "hi").replace("French",
"fr").replace("Spanish","es").replace("Chinese","zh-CN").replace("Russian","ru") +
"&text=" + textValue.getText().toString().replace("\n", " ").replace(" ", "+"));
                    i.putExtra("textDetected",
textValue.getText().toString().replace("\n", " "));
                i.putExtra("key",  keyValue.replace("English", "en").replace("Hindi",
"hi").replace("French", "fr").replace("Spanish","es").replace("Chinese","zh-
CN").replace("Russian","ru").replace("Italian","it").replace("Portuguese","pt").replace("
Swedish","sv").replace("German","de").replace("Arabic","ar").replace("Bengali","bn"));
                     startActivity(i);

                     //Uri.parse("http://translate.google.com/#auto/hi/" +
textValue.getText().toString().replace(" ", "+")));

               //

                /*Intent i = new Intent(Intent.ACTION_SEND);
                i.setType("message/rfc822");
                i.putExtra(Intent.EXTRA_SUBJECT, "Text Read");
                i.putExtra(Intent.EXTRA_TEXT, textValue.getText().toString());
                try {
                    startActivity(Intent.createChooser(i,
getString(R.string.mail_intent_chooser_text)));
                } catch (android.content.ActivityNotFoundException ex) {
                    Toast.makeText(getApplicationContext(),
                            R.string.no_email_client_error, Toast.LENGTH_SHORT).show();
                }*/
            }
        });

        Spinner spinner2 =(Spinner)findViewById(R.id.spinner2);

        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this,R.array.keys, android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

        spinner2.setAdapter(adapter);

        spinner2.setOnItemSelectedListener(this);



    }


    @Override
    protected void onResume() {
        super.onResume();
        if (textValue.getText().toString().isEmpty()) {
            copyButton.setVisibility(View.GONE);
            mailTextButton.setVisibility(View.GONE);
        } else {
            copyButton.setVisibility(View.VISIBLE);
            mailTextButton.setVisibility(View.VISIBLE);
        }
    }

        @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if(requestCode == RC_OCR_CAPTURE) {
            if (resultCode == CommonStatusCodes.SUCCESS) {
                if (data != null) {
                    String text =
data.getStringExtra(OcrCaptureActivity.TextBlockObject);
                    statusMessage.setText(R.string.ocr_success);
                    textValue.setText(text);
                    Log.d(TAG, "Text read: " + text);
                } else {
                    statusMessage.setText(R.string.ocr_failure);
                    Log.d(TAG, "No Text captured, intent data is null");
                }
            } else {
                statusMessage.setText(String.format(getString(R.string.ocr_error),
                        CommonStatusCodes.getStatusCodeString(resultCode)));
            }
        }
        else {
            super.onActivityResult(requestCode, resultCode, data);
        }
    }


    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        keyValue=parent.getItemAtPosition(position).toString();
        Toast.makeText(this,"Seclected Language"+ keyValue,Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
}
```

OcrCaptureActivity.java

```
package com.kvr.translify.activities;
```

```
import android.Manifest;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
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.GestureDetector;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
import android.widget.Toast;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GoogleApiAvailability;
import com.google.android.gms.common.api.CommonStatusCodes;
import com.google.android.gms.vision.text.TextBlock;
import com.google.android.gms.vision.text.TextRecognizer;
import com.kvr.translify.R;
import com.kvr.translify.ui.camera.CameraSource;
import com.kvr.translify.ui.camera.CameraSourcePreview;
import com.kvr.translify.ui.camera.GraphicOverlay;
import com.kvr.translify.utilities.OcrDetectorProcessor;
import com.kvr.translify.utilities.OcrGraphic;

import java.io.IOException;

/**
 * Activity for the multi-tracker app.  This app detects text and displays the value with
the
 * rear facing camera. During detection overlay graphics are drawn to indicate the
position,
 * size, and contents of each TextBlock.
 */
public final class OcrCaptureActivity extends AppCompatActivity {
    private static final String TAG = "OcrCaptureActivity";

    // 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";
    public static final String TextBlockObject = "String";

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

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

    /**
     * Initializes the UI and creates the detector pipeline.
     */
    @Override
```

```java
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.ocr_capture);

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

    // read parameters from the intent used to launch the activity.
    boolean autoFocus = getIntent().getBooleanExtra(AutoFocus, false);
    boolean useFlash = getIntent().getBooleanExtra(UseFlash, false);

    // 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, useFlash);
    } else {
        requestCameraPermission();
    }

    gestureDetector = new GestureDetector(this, new CaptureGestureListener());
    scaleGestureDetector = new ScaleGestureDetector(this, new ScaleListener());

    Snackbar.make(mGraphicOverlay, "Tap to capture. Pinch/Stretch to zoom",
            Snackbar.LENGTH_INDEFINITE)
            .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;

    View.OnClickListener listener = new View.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);

    boolean c = gestureDetector.onTouchEvent(e);

    return b || c || 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 context = getApplicationContext();

        // A text recognizer is created to find text.  An associated processor instance
        // is set to receive the text recognition results and display graphics for each
text block
        // on screen.
        TextRecognizer textRecognizer = new TextRecognizer.Builder(context).build();
        textRecognizer.setProcessor(new OcrDetectorProcessor(mGraphicOverlay));

        if (!textRecognizer.isOperational()) {
            // Note: The first time that an app using a Vision API is installed on a
            // device, GMS will download a native libraries to the device in order to do
detection.
            // Usually this completes before the app is run for the first time.  But if
that
            // download has not yet completed, then the above call will not detect any
text,
            // barcodes, or faces.
            //
            // isOperational() can be used to check if the required native libraries are
currently
            // available.  The detectors will automatically become operational once the
library
            // downloads complete on device.
            Log.w(TAG, "Detector dependencies are not yet available.");

            // Check for low storage.  If there is low storage, the native library will
not be
            // downloaded, so detection will not become operational.
            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));
            }
        }

        // Creates and starts the camera.  Note that this uses a higher resolution in
comparison
        // to other detection examples to enable the text recognizer to detect small
pieces of text.
        mCameraSource =
                new CameraSource.Builder(getApplicationContext(), textRecognizer)
                .setFacing(CameraSource.CAMERA_FACING_BACK)
                .setRequestedPreviewSize(1280, 1024)
                .setRequestedFps(2.0f)
                .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();
        }
    }

    /**
     * Releases the resources associated with the camera source, the associated
detectors, and the
     * rest of the processing pipeline.
     */
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (mPreview != null) {
            mPreview.release();
        }
    }

    /**
     * Callback for the result from requesting permissions. This method
     * is invoked for every call on {@link #requestPermissions(String[], int)}.
     * <p>
     * <strong>Note:</strong> It is possible that the permissions request interaction
     * with the user is interrupted. In this case you will receive empty permissions
     * and results arrays which should be treated as a cancellation.
     * </p>
     *
     * @param requestCode  The request code passed in {@link
#requestPermissions(String[], int)}.
     * @param permissions  The requested permissions. Never null.
     * @param grantResults The grant results for the corresponding permissions
     *                     which is either {@link PackageManager#PERMISSION_GRANTED}
     *                     or {@link PackageManager#PERMISSION_DENIED}. Never null.
     * @see #requestPermissions(String[], int)
     */
    @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;
            }
        }
    }

    /**
     * onTap is called to capture the first TextBlock under the tap location and return
it to
     * the Initializing Activity.
     *
     * @param rawX - the raw position of the tap
     * @param rawY - the raw position of the tap.
     * @return true if the activity is ending.
     */
    private boolean onTap(float rawX, float rawY) {
        OcrGraphic graphic = mGraphicOverlay.getGraphicAtLocation(rawX, rawY);
        TextBlock text = null;
        if (graphic != null) {
            text = graphic.getTextBlock();
            if (text != null && text.getValue() != null) {
                Intent data = new Intent();
                data.putExtra(TextBlockObject, text.getValue());
                setResult(CommonStatusCodes.SUCCESS, data);
                finish();
            }
            else {
                Log.d(TAG, "text data is null");
            }
        }
        else {
            Log.d(TAG,"no text detected");
        }
        return text != null;
    }

    private class CaptureGestureListener extends GestureDetector.SimpleOnGestureListener
{

        @Override
        public boolean onSingleTapConfirmed(MotionEvent e) {
            return onTap(e.getRawX(), e.getRawY()) || super.onSingleTapConfirmed(e);
        }
    }

    private class ScaleListener implements ScaleGestureDetector.OnScaleGestureListener {

        /**
         * Responds to scaling events for a gesture in progress.
         * Reported by pointer motion.
         *
         * @param detector The detector reporting the event - use this to
         *                 retrieve extended info about event state.
         * @return Whether or not the detector should consider this event
         * as handled. If an event was not handled, the detector
         * will continue to accumulate movement until an event is
         * handled. This can be useful if an application, for example,
         * only wants to update scaling factors if the change is
         * greater than 0.01.
         */
        @Override
        public boolean onScale(ScaleGestureDetector detector) {
            return false;
        }

        /**
         * Responds to the beginning of a scaling gesture. Reported by
         * new pointers going down.
         *
         * @param detector The detector reporting the event - use this to
         *                 retrieve extended info about event state.
         * @return Whether or not the detector should continue recognizing
         * this gesture. For example, if a gesture is beginning
         * with a focal point outside of a region where it makes
         * sense, onScaleBegin() may return false to ignore the
         * rest of the gesture.
         */
        @Override
        public boolean onScaleBegin(ScaleGestureDetector detector) {
            return true;
        }

        /**
         * Responds to the end of a scale gesture. Reported by existing
         * pointers going up.
         * <p/>
         * Once a scale has ended, {@link ScaleGestureDetector#getFocusX()}
         * and {@link ScaleGestureDetector#getFocusY()} will return focal point
         * of the pointers remaining on the screen.
         *
         * @param detector The detector reporting the event - use this to
         *                 retrieve extended info about event state.
         */
        @Override
        public void onScaleEnd(ScaleGestureDetector detector) {
            mCameraSource.doZoom(detector.getScaleFactor());
        }
    }
}
```

TranslateActivity.java

```
package com.kvr.translify.activities;

import android.content.Intent;
import android.graphics.Bitmap;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.webkit.WebChromeClient;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.ProgressBar;

import com.kvr.translify.R;

public class TranslateActivity extends AppCompatActivity {


    public WebView translate;
    public String link;


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


        Bundle bundle = getIntent().getExtras();
        String textDetected = bundle.getString("textDetected");
        textDetected.replace(" " , "+");
        link = bundle.getString("link");

        String key = bundle.getString("key");
        String newUA= "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.4) Gecko/20100101
Firefox/4.0";

        translate = (WebView)findViewById(R.id.webView);
        translate.getSettings().setJavaScriptEnabled(true);
        translate.setWebChromeClient(new WebChromeClient());
        translate.getSettings().setPluginState(WebSettings.PluginState.ON);
        translate.clearHistory();
        translate.clearCache(true);
        translate.reload();
        translate.getSettings().setUserAgentString(newUA);
        //translate.setWebViewClient(new WebViewClient());

//translate.loadUrl("https://translate.google.com/m/translate#view=home&op=translate&sl=a
uto&tl="+ key + "&text=" + textDetected);
        translate.loadUrl("https://translate.google.com/#auto/" + key + "/" +
textDetected);
        translate.loadUrl( "javascript:window.location.reload( true )" );

    }

    @Override
    public void onResume() {
        super.onResume();
        translate.reload();
        translate.clearCache(true);
        translate.clearHistory();
        translate.loadUrl(link);
    }

    @Override
    public void onStart() {
        super.onStart();
        translate.reload();
        translate.clearCache(true);
        translate.clearHistory();
        translate.loadUrl(link);
    }

    @Override
    public void onBackPressed() {
        if (translate.canGoBack()) {
            translate.goBack();
        } else {
            super.onBackPressed();
            translate.destroy();
        }

    }

    @Override
    public void onStop() {
        super.onStop();
        translate.destroy();
        translate.clearHistory();
        translate.clearCache(true);
    }

}
```

CameraSource.java

```
package com.kvr.translify.ui.camera;

import android.Manifest;
import android.annotation.SuppressLint;
import android.annotation.TargetApi;
import android.content.Context;
import android.graphics.ImageFormat;
import android.graphics.SurfaceTexture;
import android.hardware.Camera;
import android.hardware.Camera.CameraInfo;
import android.os.Build;
import android.os.SystemClock;
import android.support.annotation.Nullable;
import android.support.annotation.RequiresPermission;
import android.support.annotation.StringDef;
import android.util.Log;
import android.view.Surface;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.WindowManager;

import com.google.android.gms.common.images.Size;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.Frame;

import java.io.IOException;
import java.lang.Thread.State;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@SuppressWarnings("deprecation")
public class CameraSource {
    @SuppressLint("InlinedApi")
    public static final int CAMERA_FACING_BACK = CameraInfo.CAMERA_FACING_BACK;
    @SuppressLint("InlinedApi")
    public static final int CAMERA_FACING_FRONT = CameraInfo.CAMERA_FACING_FRONT;
```

```
private static final String TAG = "OpenCameraSource";

    private static final int DUMMY_TEXTURE_NAME = 100;


private static final float ASPECT_RATIO_TOLERANCE = 0.01f;

@StringDef({
    Camera.Parameters.FOCUS_MODE_CONTINUOUS_PICTURE,
    Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO,
    Camera.Parameters.FOCUS_MODE_AUTO,
    Camera.Parameters.FOCUS_MODE_EDOF,
    Camera.Parameters.FOCUS_MODE_FIXED,
    Camera.Parameters.FOCUS_MODE_INFINITY,
    Camera.Parameters.FOCUS_MODE_MACRO
})
@Retention(RetentionPolicy.SOURCE)
private @interface FocusMode {}

@StringDef({
    Camera.Parameters.FLASH_MODE_ON,
    Camera.Parameters.FLASH_MODE_OFF,
    Camera.Parameters.FLASH_MODE_AUTO,
    Camera.Parameters.FLASH_MODE_RED_EYE,
    Camera.Parameters.FLASH_MODE_TORCH
})
@Retention(RetentionPolicy.SOURCE)
private @interface FlashMode {}

private Context mContext;

private final Object mCameraLock = new Object();

// Guarded by mCameraLock
private Camera mCamera;

private int mFacing = CAMERA_FACING_BACK;
private int mRotation;
private Size mPreviewSize;
private float mRequestedFps = 30.0f;
private int mRequestedPreviewWidth = 1024;
private int mRequestedPreviewHeight = 768;
private String mFocusMode = null;
private String mFlashMode = null;
private SurfaceView mDummySurfaceView;
private SurfaceTexture mDummySurfaceTexture;
private Thread mProcessingThread;
private FrameProcessingRunnable mFrameProcessor;
private Map<byte[], ByteBuffer> mBytesToByteBuffer = new HashMap<>();
public static class Builder {
    private final Detector<?> mDetector;
    private CameraSource mCameraSource = new CameraSource();

    public Builder(Context context, Detector<?> detector) {
        if (context == null) {
            throw new IllegalArgumentException("No context supplied.");
        }
        if (detector == null) {
            throw new IllegalArgumentException("No detector supplied.");
        }

        mDetector = detector;
        mCameraSource.mContext = context;
    }

    public Builder setRequestedFps(float fps) {
        if (fps <= 0) {
            throw new IllegalArgumentException("Invalid fps: " + fps);
        }
```

```
            mCameraSource.mRequestedFps = fps;
            return this;
        }

        public Builder setFocusMode(@FocusMode String mode) {
            mCameraSource.mFocusMode = mode;
            return this;
        }

        public Builder setFlashMode(@FlashMode String mode) {
            mCameraSource.mFlashMode = mode;
            return this;
        }
        public Builder setRequestedPreviewSize(int width, int height) {
                        final int MAX = 1000000;
            if ((width <= 0) || (width > MAX) || (height <= 0) || (height > MAX)) {
                throw new IllegalArgumentException("Invalid preview size: " + width + "x"
+ height);
            }
            mCameraSource.mRequestedPreviewWidth = width;
            mCameraSource.mRequestedPreviewHeight = height;
            return this;
        }

        public Builder setFacing(int facing) {
            if ((facing != CAMERA_FACING_BACK) && (facing != CAMERA_FACING_FRONT)) {
                throw new IllegalArgumentException("Invalid camera: " + facing);
            }
            mCameraSource.mFacing = facing;
            return this;
        }

                public CameraSource build() {
            mCameraSource.mFrameProcessor = mCameraSource.new
FrameProcessingRunnable(mDetector);
            return mCameraSource;
        }
    }


         public interface ShutterCallback {
                void onShutter();
    }


    public interface PictureCallback {

        void onPictureTaken(byte[] data);
    }


    public interface AutoFocusCallback {
        void onAutoFocus(boolean success);
    }

        public interface AutoFocusMoveCallback {

        void onAutoFocusMoving(boolean start);
    }

    public void release() {
        synchronized (mCameraLock) {
            stop();
            mFrameProcessor.release();
        }
    }

        @RequiresPermission(Manifest.permission.CAMERA)
    public CameraSource start() throws IOException {
        synchronized (mCameraLock) {
```

```
        if (mCamera != null) {
            return this;
        }

        mCamera = createCamera();

                    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
            mDummySurfaceTexture = new SurfaceTexture(DUMMY_TEXTURE_NAME);
            mCamera.setPreviewTexture(mDummySurfaceTexture);
        } else {
            mDummySurfaceView = new SurfaceView(mContext);
            mCamera.setPreviewDisplay(mDummySurfaceView.getHolder());
        }
        mCamera.startPreview();

        mProcessingThread = new Thread(mFrameProcessor);
        mFrameProcessor.setActive(true);
        mProcessingThread.start();
    }
    return this;
}

@RequiresPermission(Manifest.permission.CAMERA)
public CameraSource start(SurfaceHolder surfaceHolder) throws IOException {
    synchronized (mCameraLock) {
        if (mCamera != null) {
            return this;
        }

        mCamera = createCamera();
        mCamera.setPreviewDisplay(surfaceHolder);
        mCamera.startPreview();

        mProcessingThread = new Thread(mFrameProcessor);
        mFrameProcessor.setActive(true);
        mProcessingThread.start();
    }
    return this;
}


public void stop() {
    synchronized (mCameraLock) {
        mFrameProcessor.setActive(false);
        if (mProcessingThread != null) {
            try {
                                mProcessingThread.join();
            } catch (InterruptedException e) {
                Log.d(TAG, "Frame processing thread interrupted on release.");
            }
            mProcessingThread = null;
        }


        mBytesToByteBuffer.clear();

        if (mCamera != null) {
            mCamera.stopPreview();
            mCamera.setPreviewCallbackWithBuffer(null);
            try {

                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
                    mCamera.setPreviewTexture(null);

                } else {
                    mCamera.setPreviewDisplay(null);
                }
            } catch (Exception e) {
                Log.e(TAG, "Failed to clear camera preview: " + e);
            }
            mCamera.release();
            mCamera = null;
        }
    }
}

    public Size getPreviewSize() {
    return mPreviewSize;
}

public int getCameraFacing() {
    return mFacing;
}

public int doZoom(float scale) {
    synchronized (mCameraLock) {
        if (mCamera == null) {
            return 0;
        }
        int currentZoom = 0;
        int maxZoom;
        Camera.Parameters parameters = mCamera.getParameters();
        if (!parameters.isZoomSupported()) {
            Log.w(TAG, "Zoom is not supported on this device");
            return currentZoom;
        }
        maxZoom = parameters.getMaxZoom();

        currentZoom = parameters.getZoom() + 1;
        float newZoom;
        if (scale > 1) {
            newZoom = currentZoom + scale * (maxZoom / 10);
        } else {
            newZoom = currentZoom * scale;
        }
        currentZoom = Math.round(newZoom) - 1;
        if (currentZoom < 0) {
            currentZoom = 0;
        } else if (currentZoom > maxZoom) {
            currentZoom = maxZoom;
        }
        parameters.setZoom(currentZoom);
        mCamera.setParameters(parameters);
        return currentZoom;
    }
}

public void takePicture(ShutterCallback shutter, PictureCallback jpeg) {
    synchronized (mCameraLock) {
        if (mCamera != null) {
            PictureStartCallback startCallback = new PictureStartCallback();
            startCallback.mDelegate = shutter;
            PictureDoneCallback doneCallback = new PictureDoneCallback();
            doneCallback.mDelegate = jpeg;
            mCamera.takePicture(startCallback, null, null, doneCallback);
        }
    }
}

@Nullable
@FocusMode
public String getFocusMode() {
    return mFocusMode;
}

    public boolean setFocusMode(@FocusMode String mode) {
    synchronized (mCameraLock) {
        if (mCamera != null && mode != null) {
            Camera.Parameters parameters = mCamera.getParameters();
            if (parameters.getSupportedFocusModes().contains(mode)) {
                parameters.setFocusMode(mode);
                mCamera.setParameters(parameters);
                mFocusMode = mode;
                return true;
            }
        }

        return false;
    }
}

@Nullable
@FlashMode
public String getFlashMode() {
    return mFlashMode;
}

public boolean setFlashMode(@FlashMode String mode) {
    synchronized (mCameraLock) {
        if (mCamera != null && mode != null) {
            Camera.Parameters parameters = mCamera.getParameters();
            if (parameters.getSupportedFlashModes().contains(mode)) {
                parameters.setFlashMode(mode);
                mCamera.setParameters(parameters);
                mFlashMode = mode;
                return true;
            }
        }

        return false;
    }
}

    public void autoFocus(@Nullable AutoFocusCallback cb) {
    synchronized (mCameraLock) {
        if (mCamera != null) {
            CameraAutoFocusCallback autoFocusCallback = null;
            if (cb != null) {
                autoFocusCallback = new CameraAutoFocusCallback();
                autoFocusCallback.mDelegate = cb;
            }
            mCamera.autoFocus(autoFocusCallback);
        }
    }
}

    public void cancelAutoFocus() {
    synchronized (mCameraLock) {
        if (mCamera != null) {
            mCamera.cancelAutoFocus();
        }
    }
}

    @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
public boolean setAutoFocusMoveCallback(@Nullable AutoFocusMoveCallback cb) {
    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.JELLY_BEAN) {
        return false;
    }

    synchronized (mCameraLock) {
        if (mCamera != null) {
            CameraAutoFocusMoveCallback autoFocusMoveCallback = null;
            if (cb != null) {
                autoFocusMoveCallback = new CameraAutoFocusMoveCallback();
                autoFocusMoveCallback.mDelegate = cb;
            }
            mCamera.setAutoFocusMoveCallback(autoFocusMoveCallback);
        }
    }
```

```
        return true;
    }

    private CameraSource() {
    }


    private class PictureStartCallback implements Camera.ShutterCallback {
        private ShutterCallback mDelegate;

        @Override
        public void onShutter() {
            if (mDelegate != null) {
                mDelegate.onShutter();
            }
        }
    }

        private class PictureDoneCallback implements Camera.PictureCallback {
        private PictureCallback mDelegate;

        @Override
        public void onPictureTaken(byte[] data, Camera camera) {
            if (mDelegate != null) {
                mDelegate.onPictureTaken(data);
            }
            synchronized (mCameraLock) {
                if (mCamera != null) {
                    mCamera.startPreview();
                }
            }
        }
    }

    private class CameraAutoFocusCallback implements Camera.AutoFocusCallback {
        private AutoFocusCallback mDelegate;

        @Override
        public void onAutoFocus(boolean success, Camera camera) {
            if (mDelegate != null) {
                mDelegate.onAutoFocus(success);
            }
        }
    }

    @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
    private class CameraAutoFocusMoveCallback implements Camera.AutoFocusMoveCallback {
        private AutoFocusMoveCallback mDelegate;

        @Override
        public void onAutoFocusMoving(boolean start, Camera camera) {
            if (mDelegate != null) {
                mDelegate.onAutoFocusMoving(start);
            }
        }
    }

        @SuppressLint("InlinedApi")
    private Camera createCamera() {
        int requestedCameraId = getIdForRequestedCamera(mFacing);
        if (requestedCameraId == -1) {
            throw new RuntimeException("Could not find requested camera.");
        }
        Camera camera = Camera.open(requestedCameraId);

        SizePair sizePair = selectSizePair(camera, mRequestedPreviewWidth,
mRequestedPreviewHeight);
        if (sizePair == null) {
            throw new RuntimeException("Could not find suitable preview size.");
        }
        Size pictureSize = sizePair.pictureSize();
        mPreviewSize = sizePair.previewSize();

        int[] previewFpsRange = selectPreviewFpsRange(camera, mRequestedFps);
        if (previewFpsRange == null) {
            throw new RuntimeException("Could not find suitable preview frames per second
range.");
        }

        Camera.Parameters parameters = camera.getParameters();

        if (pictureSize != null) {
            parameters.setPictureSize(pictureSize.getWidth(), pictureSize.getHeight());
        }

        parameters.setPreviewSize(mPreviewSize.getWidth(), mPreviewSize.getHeight());
        parameters.setPreviewFpsRange(
                previewFpsRange[Camera.Parameters.PREVIEW_FPS_MIN_INDEX],
                previewFpsRange[Camera.Parameters.PREVIEW_FPS_MAX_INDEX]);
        parameters.setPreviewFormat(ImageFormat.NV21);

        setRotation(camera, parameters, requestedCameraId);

        if (mFocusMode != null) {
            if (parameters.getSupportedFocusModes().contains(
                    mFocusMode)) {
                parameters.setFocusMode(mFocusMode);
            } else {
                Log.i(TAG, "Camera focus mode: " + mFocusMode +
                    " is not supported on this device.");
            }
        }


        mFocusMode = parameters.getFocusMode();

        if (mFlashMode != null) {
            if (parameters.getSupportedFlashModes().contains(
                    mFlashMode)) {
                parameters.setFlashMode(mFlashMode);
            } else {
                Log.i(TAG, "Camera flash mode: " + mFlashMode +
                    " is not supported on this device.");
            }
        }


        mFlashMode = parameters.getFlashMode();

        camera.setParameters(parameters);

                camera.setPreviewCallbackWithBuffer(new CameraPreviewCallback());
        camera.addCallbackBuffer(createPreviewBuffer(mPreviewSize));
        camera.addCallbackBuffer(createPreviewBuffer(mPreviewSize));
        camera.addCallbackBuffer(createPreviewBuffer(mPreviewSize));
        camera.addCallbackBuffer(createPreviewBuffer(mPreviewSize));

        return camera;
    }

        private static int getIdForRequestedCamera(int facing) {
        CameraInfo cameraInfo = new CameraInfo();
        for (int i = 0; i < Camera.getNumberOfCameras(); ++i) {
            Camera.getCameraInfo(i, cameraInfo);
            if (cameraInfo.facing == facing) {
                return i;
            }
        }
        return -1;
    }

    private static SizePair selectSizePair(Camera camera, int desiredWidth, int
desiredHeight) {
        List<SizePair> validPreviewSizes = generateValidPreviewSizeList(camera);

                SizePair selectedPair = null;
        int minDiff = Integer.MAX_VALUE;
        for (SizePair sizePair : validPreviewSizes) {
            Size size = sizePair.previewSize();
            int diff = Math.abs(size.getWidth() - desiredWidth) +
                    Math.abs(size.getHeight() - desiredHeight);
            if (diff < minDiff) {
                selectedPair = sizePair;
                minDiff = diff;
            }
        }

        return selectedPair;
    }


    private static class SizePair {
        private Size mPreview;
        private Size mPicture;

        public SizePair(android.hardware.Camera.Size previewSize,
                        android.hardware.Camera.Size pictureSize) {
            mPreview = new Size(previewSize.width, previewSize.height);
            if (pictureSize != null) {
                mPicture = new Size(pictureSize.width, pictureSize.height);
            }
        }

        public Size previewSize() {
            return mPreview;
        }

        @SuppressWarnings("unused")
        public Size pictureSize() {
            return mPicture;
        }
    }
    private static List<SizePair> generateValidPreviewSizeList(Camera camera) {
        Camera.Parameters parameters = camera.getParameters();
        List<android.hardware.Camera.Size> supportedPreviewSizes =
                parameters.getSupportedPreviewSizes();
        List<android.hardware.Camera.Size> supportedPictureSizes =
                parameters.getSupportedPictureSizes();
        List<SizePair> validPreviewSizes = new ArrayList<>();
        for (android.hardware.Camera.Size previewSize : supportedPreviewSizes) {
            float previewAspectRatio = (float) previewSize.width / (float)
previewSize.height;

            for (android.hardware.Camera.Size pictureSize : supportedPictureSizes) {
                float pictureAspectRatio = (float) pictureSize.width / (float)
pictureSize.height;
                if (Math.abs(previewAspectRatio - pictureAspectRatio) <
ASPECT_RATIO_TOLERANCE) {
                    validPreviewSizes.add(new SizePair(previewSize, pictureSize));
                    break;
                }
            }
        }


        if (validPreviewSizes.size() == 0) {
            Log.w(TAG, "No preview sizes have a corresponding same-aspect-ratio picture
size");
            for (android.hardware.Camera.Size previewSize : supportedPreviewSizes) {
```

```
                                validPreviewSizes.add(new SizePair(previewSize, null));
                }
            }

            return validPreviewSizes;
        }

            private int[] selectPreviewFpsRange(Camera camera, float desiredPreviewFps) {
    int desiredPreviewFpsScaled = (int) (desiredPreviewFps * 1000.0f);

                    int[] selectedFpsRange = null;
            int minDiff = Integer.MAX_VALUE;
            List<int[]> previewFpsRangeList =
    camera.getParameters().getSupportedPreviewFpsRange();
            for (int[] range : previewFpsRangeList) {
                int deltaMin = desiredPreviewFpsScaled -
    range[Camera.Parameters.PREVIEW_FPS_MIN_INDEX];
                int deltaMax = desiredPreviewFpsScaled -
    range[Camera.Parameters.PREVIEW_FPS_MAX_INDEX];
                int diff = Math.abs(deltaMin) + Math.abs(deltaMax);
                if (diff < minDiff) {
                    selectedFpsRange = range;
                    minDiff = diff;
                }
            }
            return selectedFpsRange;
        }

            private void setRotation(Camera camera, Camera.Parameters parameters, int
    cameraId) {
            WindowManager windowManager =
                    (WindowManager) mContext.getSystemService(Context.WINDOW_SERVICE);
            int degrees = 0;
            int rotation = windowManager.getDefaultDisplay().getRotation();
            switch (rotation) {
                case Surface.ROTATION_0:
                    degrees = 0;
                    break;
                case Surface.ROTATION_90:
                    degrees = 90;
                    break;
                case Surface.ROTATION_180:
                    degrees = 180;
                    break;
                case Surface.ROTATION_270:
                    degrees = 270;
                    break;
                default:
                    Log.e(TAG, "Bad rotation value: " + rotation);
            }

            CameraInfo cameraInfo = new CameraInfo();
            Camera.getCameraInfo(cameraId, cameraInfo);

            int angle;
            int displayAngle;
            if (cameraInfo.facing == Camera.CameraInfo.CAMERA_FACING_FRONT) {
                angle = (cameraInfo.orientation + degrees) % 360;
                displayAngle = (360 - angle); // compensate for it being mirrored
            } else {  // back-facing
                angle = (cameraInfo.orientation - degrees + 360) % 360;
                displayAngle = angle;
            }


            mRotation = angle / 90;

            camera.setDisplayOrientation(displayAngle);
            parameters.setRotation(angle);
        }
        private byte[] createPreviewBuffer(Size previewSize) {
            int bitsPerPixel = ImageFormat.getBitsPerPixel(ImageFormat.NV21);
            long sizeInBits = previewSize.getHeight() * previewSize.getWidth() *
    bitsPerPixel;
            int bufferSize = (int) Math.ceil(sizeInBits / 8.0d) + 1;


            byte[] byteArray = new byte[bufferSize];
            ByteBuffer buffer = ByteBuffer.wrap(byteArray);
            if (!buffer.hasArray() || (buffer.array() != byteArray)) {
                            throw new IllegalStateException("Failed to create valid buffer
    for camera source.");
            }

            mBytesToByteBuffer.put(byteArray, buffer);
            return byteArray;
        }

        private class CameraPreviewCallback implements Camera.PreviewCallback {
            @Override
            public void onPreviewFrame(byte[] data, Camera camera) {
                mFrameProcessor.setNextFrame(data, camera);
            }
        }

            private class FrameProcessingRunnable implements Runnable {
            private Detector<?> mDetector;
            private long mStartTimeMillis = SystemClock.elapsedRealtime();


            private final Object mLock = new Object();
            private boolean mActive = true;

            private long mPendingTimeMillis;
            private int mPendingFrameId = 0;
            private ByteBuffer mPendingFrameData;

            FrameProcessingRunnable(Detector<?> detector) {
                mDetector = detector;
            }
            @SuppressLint("Assert")
            void release() {
                assert (mProcessingThread.getState() == State.TERMINATED);
                mDetector.release();
                mDetector = null;
            }


            void setActive(boolean active) {
                synchronized (mLock) {
                    mActive = active;
                    mLock.notifyAll();
                }
            }

                    void setNextFrame(byte[] data, Camera camera) {
                synchronized (mLock) {
                    if (mPendingFrameData != null) {
                        camera.addCallbackBuffer(mPendingFrameData.array());
                        mPendingFrameData = null;
                    }

                    if (!mBytesToByteBuffer.containsKey(data)) {
                        Log.d(TAG,
                            "Skipping frame.  Could not find ByteBuffer associated with the
    image " +
                            "data from the camera.");
                        return;
                    }
                    mPendingTimeMillis = SystemClock.elapsedRealtime() - mStartTimeMillis;
                    mPendingFrameId++;
                    mPendingFrameData = mBytesToByteBuffer.get(data);

                                    mLock.notifyAll();
                }
            }

            @Override
            public void run() {
                Frame outputFrame;
                ByteBuffer data;

                while (true) {
                    synchronized (mLock) {
                        while (mActive && (mPendingFrameData == null)) {
                            try {

                                mLock.wait();
                            } catch (InterruptedException e) {
                                Log.d(TAG, "Frame processing loop terminated.", e);
                                return;
                            }
                        }

                        if (!mActive) {
                                                    return;
                        }

                        outputFrame = new Frame.Builder()
                                .setImageData(mPendingFrameData, mPreviewSize.getWidth(),
                                        mPreviewSize.getHeight(), ImageFormat.NV21)
                                .setId(mPendingFrameId)
                                .setTimestampMillis(mPendingTimeMillis)
                                .setRotation(mRotation)
                                .build();
                        data = mPendingFrameData;
                        mPendingFrameData = null;
                    }



                    try {
                        mDetector.receiveFrame(outputFrame);
                    } catch (Throwable t) {
                        Log.e(TAG, "Exception thrown from receiver.", t);
                    } finally {
                        mCamera.addCallbackBuffer(data.array());
                    }
                }
            }
        }
    }
```

CameraSourcePreview.java

```
package com.kvr.translify.ui.camera;

import android.Manifest;
import android.content.Context;
import android.content.res.Configuration;
import android.support.annotation.RequiresPermission;
import android.util.AttributeSet;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.ViewGroup;
import com.google.android.gms.common.images.Size;

import java.io.IOException;

public class CameraSourcePreview extends ViewGroup {
    private static final String TAG = "CameraSourcePreview";

    private Context mContext;
    private SurfaceView mSurfaceView;
    private boolean mStartRequested;
    private boolean mSurfaceAvailable;
    private CameraSource mCameraSource;

    private GraphicOverlay mOverlay;

    public CameraSourcePreview(Context context, AttributeSet attrs) {
        super(context, attrs);
        mContext = context;
        mStartRequested = false;
        mSurfaceAvailable = false;

        mSurfaceView = new SurfaceView(context);
        mSurfaceView.getHolder().addCallback(new SurfaceCallback());
        addView(mSurfaceView);
    }

    @RequiresPermission(Manifest.permission.CAMERA)
    public void start(CameraSource cameraSource) throws IOException, SecurityException {
        if (cameraSource == null) {
            stop();
        }

        mCameraSource = cameraSource;

        if (mCameraSource != null) {
            mStartRequested = true;
            startIfReady();
        }
    }

    @RequiresPermission(Manifest.permission.CAMERA)
    public void start(CameraSource cameraSource, GraphicOverlay overlay) throws
IOException, SecurityException {
        mOverlay = overlay;
        start(cameraSource);
    }

    public void stop() {
        if (mCameraSource != null) {
            mCameraSource.stop();
        }
    }

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

    @RequiresPermission(Manifest.permission.CAMERA)
    private void startIfReady() throws IOException, SecurityException {
        if (mStartRequested && mSurfaceAvailable) {
            mCameraSource.start(mSurfaceView.getHolder());
            if (mOverlay != null) {
                Size size = mCameraSource.getPreviewSize();
                int min = Math.min(size.getWidth(), size.getHeight());
                int max = Math.max(size.getWidth(), size.getHeight());
                if (isPortraitMode()) {
                    // Swap width and height sizes when in portrait, since it will be
rotated by
                    // 90 degrees
                    mOverlay.setCameraInfo(min, max, mCameraSource.getCameraFacing());
                } else {
                    mOverlay.setCameraInfo(max, min, mCameraSource.getCameraFacing());
                }
                mOverlay.clear();
            }
            mStartRequested = false;
        }
    }

    private class SurfaceCallback implements SurfaceHolder.Callback {
        @Override
        public void surfaceCreated(SurfaceHolder surface) {
            mSurfaceAvailable = true;
            try {
                startIfReady();
            } catch (SecurityException se) {
                Log.e(TAG,"Do not have permission to start the camera", se);
            } catch (IOException e) {
                Log.e(TAG, "Could not start camera source.", e);
            }
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder surface) {
            mSurfaceAvailable = false;
        }

        @Override
        public void surfaceChanged(SurfaceHolder holder, int format, int width, int
height) {
        }
    }

    @Override
    protected void onLayout(boolean changed, int left, int top, int right, int bottom) {
        int previewWidth = 320;
        int previewHeight = 240;
        if (mCameraSource != null) {
            Size size = mCameraSource.getPreviewSize();
            if (size != null) {
                previewWidth = size.getWidth();
                previewHeight = size.getHeight();
            }
        }


        if (isPortraitMode()) {
            int tmp = previewWidth;
            previewWidth = previewHeight;
            previewHeight = tmp;
        }

        final int viewWidth = right - left;
        final int viewHeight = bottom - top;

        int childWidth;
        int childHeight;
        int childXOffset = 0;
        int childYOffset = 0;
        float widthRatio = (float) viewWidth / (float) previewWidth;
        float heightRatio = (float) viewHeight / (float) previewHeight;


        if (widthRatio > heightRatio) {
            childWidth = viewWidth;
            childHeight = (int) ((float) previewHeight * widthRatio);
            childYOffset = (childHeight - viewHeight) / 2;
        } else {
            childWidth = (int) ((float) previewWidth * heightRatio);
            childHeight = viewHeight;
            childXOffset = (childWidth - viewWidth) / 2;
        }

        for (int i = 0; i < getChildCount(); ++i) {
            // One dimension will be cropped.  We shift child over or up by this offset
and adjust
            // the size to maintain the proper aspect ratio.
            getChildAt(i).layout(
                    -1 * childXOffset, -1 * childYOffset,
                    childWidth - childXOffset, childHeight - childYOffset);
        }

        try {
            startIfReady();
        } catch (SecurityException se) {
            Log.e(TAG,"Do not have permission to start the camera", se);
        } catch (IOException e) {
            Log.e(TAG, "Could not start camera source.", e);
        }
    }

    private boolean isPortraitMode() {
        int orientation = mContext.getResources().getConfiguration().orientation;
        if (orientation == Configuration.ORIENTATION_LANDSCAPE) {
            return false;
        }
        if (orientation == Configuration.ORIENTATION_PORTRAIT) {
            return true;
        }

        Log.d(TAG, "isPortraitMode returning false by default");
        return false;
    }
}
```

GraphicOverlay.java

```
package com.kvr.translify.ui.camera;

import android.content.Context;
import android.graphics.Canvas;
import android.util.AttributeSet;
import android.view.View;
import com.google.android.gms.vision.CameraSource;
import java.util.HashSet;
import java.util.Set;
public class GraphicOverlay<T extends GraphicOverlay.Graphic> extends View {
    private final Object mLock = new Object();
    private int mPreviewWidth;
    private float mWidthScaleFactor = 1.0f;
    private int mPreviewHeight;
    private float mHeightScaleFactor = 1.0f;
    private int mFacing = CameraSource.CAMERA_FACING_BACK;
    private Set<T> mGraphics = new HashSet<>();
    public static abstract class Graphic {
        private GraphicOverlay mOverlay;

        public Graphic(GraphicOverlay overlay) {
            mOverlay = overlay;
        }

                public abstract void draw(Canvas canvas);

                public abstract boolean contains(float x, float y);

                public float scaleX(float horizontal) {
```

```
            return horizontal * mOverlay.mWidthScaleFactor;
        }


        public float scaleY(float vertical) {
            return vertical * mOverlay.mHeightScaleFactor;
        }

        public float translateX(float x) {
            if (mOverlay.mFacing == CameraSource.CAMERA_FACING_FRONT) {
                return mOverlay.getWidth() - scaleX(x);
            } else {
                return scaleX(x);
            }
        }

                public float translateY(float y) {
            return scaleY(y);
        }

        public void postInvalidate()
mOverlay.postInvalidate();
        }
    }

    public GraphicOverlay(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

        public void clear() {
        synchronized (mLock) {
            mGraphics.clear();
        }
        postInvalidate();
    }

        public void add(T graphic) {
        synchronized (mLock) {
            mGraphics.add(graphic);
        }
        postInvalidate();
    }

        public void remove(T graphic) {
        synchronized (mLock) {
            mGraphics.remove(graphic);
        }
        postInvalidate();
    }

        public T getGraphicAtLocation(float rawX, float rawY) {
        synchronized (mLock) {

            int[] location = new int[2];
            this.getLocationOnScreen(location);
            for (T graphic : mGraphics) {
                if (graphic.contains(rawX - location[0], rawY - location[1])) {
                    return graphic;
                }
            }
            return null;
        }
    }

        public void setCameraInfo(int previewWidth, int previewHeight, int facing) {
        synchronized (mLock) {
            mPreviewWidth = previewWidth;
            mPreviewHeight = previewHeight;
            mFacing = facing;
        }
        postInvalidate();
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        synchronized (mLock) {
            if ((mPreviewWidth != 0) && (mPreviewHeight != 0)) {
                mWidthScaleFactor = (float) canvas.getWidth() / (float) mPreviewWidth;
                mHeightScaleFactor = (float) canvas.getHeight() / (float) mPreviewHeight;
            }

            for (Graphic graphic : mGraphics) {
                graphic.draw(canvas);  }}}}
```

OcrDetectorProcessor.java

```
package com.kvr.translify.utilities;

import android.util.SparseArray;

import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.text.TextBlock;
import com.kvr.translify.ui.camera.GraphicOverlay;

public class OcrDetectorProcessor implements Detector.Processor<TextBlock> {

    private GraphicOverlay<OcrGraphic> mGraphicOverlay;

    public OcrDetectorProcessor(GraphicOverlay<OcrGraphic> ocrGraphicOverlay) {
        mGraphicOverlay = ocrGraphicOverlay;
    }

        @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);
            OcrGraphic graphic = new OcrGraphic(mGraphicOverlay, item);
            mGraphicOverlay.add(graphic);
        }
    }

        @Override
    public void release() {
        mGraphicOverlay.clear();
    }
}
```

OcrGraphic.java

```
/*
 * 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.kvr.translify.utilities;
```

```
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.RectF;

import com.google.android.gms.vision.text.Text;
import com.google.android.gms.vision.text.TextBlock;
import com.kvr.translify.ui.camera.GraphicOverlay;

import java.util.List;

public class OcrGraphic extends GraphicOverlay.Graphic {

    private int mId;

    private static final int TEXT_COLOR = Color.WHITE;

    private static Paint sRectPaint;
    private static Paint sTextPaint;
    private final TextBlock mText;

    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(54.0f);
        }

        postInvalidate();
    }

    public int getId() {
        return mId;
    }

    public void setId(int id) {
        this.mId = id;
    }

    public TextBlock getTextBlock() {
        return mText;
    }

        public boolean contains(float x, float y) {
        TextBlock text = mText;
        if (text == null) {
            return false;
        }
        RectF rect = new RectF(text.getBoundingBox());
        rect.left = translateX(rect.left);
        rect.top = translateY(rect.top);
        rect.right = translateX(rect.right);
        rect.bottom = translateY(rect.bottom);
        return (rect.left < x && rect.right > x && rect.top < y && rect.bottom > y);
    }
    @Override
    public void draw(Canvas canvas) {
        TextBlock text = mText;
        if (text == null) {
            return;
        }


        RectF rect = new RectF(text.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);


        List<? extends Text> textComponents = text.getComponents();
        for(Text currentText : textComponents) {
            float left = translateX(currentText.getBoundingBox().left);
            float bottom = translateY(currentText.getBoundingBox().bottom);
            canvas.drawText(currentText.getValue(), left, bottom, sTextPaint);
        }
    }
}
```