**SYSTEM DESIGN DOCUMENT**

**1 INTRODUCTION**

In this running world ,there is growing demands for software system to recognize characters in computer system when information scanned through paper documents as we know we have number of newspapers and books which are in printed format related to different subjects.To recognize the characters from printed format we used technology called OCR (Optical Character Recognition).It is a widespread technology to recognise text inside images, such as scanned documents and photos. OCR technology is used to convert virtually any kind of images containing written text (typed, handwritten or printed) into machine-readable text data.OCR Technology became popular in the early 1990s while attempting to digitise historic newspapers. Since then the technology has undergone several improvements. Nowadays solutions deliver near to perfect [OCR accuracy](https://docparser.com/blog/improve-ocr-accuracy/). Advanced methods like [Zonal OCR](https://docparser.com/blog/zonal-ocr/) are used to automate complex document based workflows.

Probably the most well known use case for OCR is converting printed paper documents into machine-readable text documents. Once a scanned paper document goes through OCR processing, the text of the document can be edited with word processors like Microsoft Word or Google Docs. Before OCR technology was available, the only option to digitise printed paper documents was to manually re-typing the text. Not only was it massively time consuming, it also came with inaccuracy and typing errors.

OCR is often used as a “hidden” technology, powering many well known systems and services in our daily life. Less known, but as important, use cases for OCR technology include data entry automation, indexing documents for search engines, automatic number plate recognition, as well as assisting blind and visually impaired persons.OCR technology has proven immensely useful in digitising historic newspapers and texts that have now been converted into fully searchable formats and had made accessing those earlier texts easier and faster.

**1.1 Purpose and Scope**

Developing a Responsive Web Application that allow scanning for Handwritten question pages of content into a word template.the scope of the project is to provide efficient and enhanced software tool for users to perform document image analysis and document processing by reading and recognizing the characters in research,academics and business organizations that are having large pool of documents,scanned images and irrespective of size of documents and type of characters in documents,the product is recognizing them,searching them and processing them faster according to the need of environment.

**1.2 Project Executive Summary**

By using OCR technology we have execute process and determines our objectives as follows:

The primary objective is to speed up the character recognition in document processing.As a result the system can process a huge number of documents within less time and hence saves the time of execution.

Since the character recognition is based on grid infrastructure,it aims to recognize multiple heterogeneous characters that belong to multiple universal languages with different properties and alignments.

1.2.1 System Overview

In the running world there is a growing demand for the users to convert the printed documents into electronic documents for maintaining the security of their data. Hence the basic OCR system was invented to convert the data available on papers into computer process-able documents, So that the documents can be editable and reusable. The existing system/the previous system of OCR on a grid infrastructure is just OCR without grid functionality. That is the existing system deals with the homogeneous character recognition or character recognition of single languages.

.Our proposed system is OCR on grid infrastructure which is character recognition system that supports recognition of characters in multiple languages.This infrastructure is what we call grid infrastructure which eliminates the problems of heterogeneous characters recognition and performs multiple functionalities to be perform on the documents.The multiple functionalities include editing and searching too where as the existing system only editing of the documents.In this context the infrastructure means the group that supports group of specific set of language thus the OCR in grid infrastructure is multilingual.

1.2.2 Design Constraints

For developing OCR technology based web applications we have to use open source case tools such as Tesseract.

Tesseract is an open source optical character recognition engine . It was developed at HP in between 1984 to1994. It was modified and improved in 1995 with greater accuracy. In late 2005, HP released Tesseract for open source. It is now available at. It is highly portable.It is more focused towards providing less rejection than accuracy. Currently only the command base version is available. As of now Tesseract version 3.01 is released and available for use. HP never used it. Now it is developed and maintained by Google. It provides support for various languages.

1.2.3 Future Contingencies

There are several key issues to consider in deciding whether to use OCR at all orc choosing between different possible appropriate uses for the text output. The main factors to consider are a combination of accuracy, efficiency and the value gained from the process. If the accuracy is below 98% the considerations of the cost in terms of time and effort to proofread and correct the resource would have to be accounted for if a full text representation is to be made. For instance, see the EEBO production description for how the accuracy issue changed their potential approaches4. If the OCR engine is not capable to delivering the required accuracy then rekeying the text may become viable, but only if the intellectual value to be gained from having the rekeyed text matches the project's goals and budgets.Otherwise, OCR for indexing and retrieval may be the most viable option.

circumstances significant words and names are repeated which improves even more the chances of retrieval and can enable high retrieval rates for OCR accuracies measuring lower than 90%.

**1.3 Points of Contact**

For details please contact on following :

Email: [harshadnimsarkar06@gmail.com](mailto:harshadnimsarkar06@gmail.com)

**1.4 Project References**

1.<https://www.geeksforgeeks.org/tesseract-ocr-with-java-with-examples/>

2.<https://en.wikipedia.org/wiki/Optical_character_recognition>

3.<https://docs.oracle.com/javaee/7/tutorial/index.html>

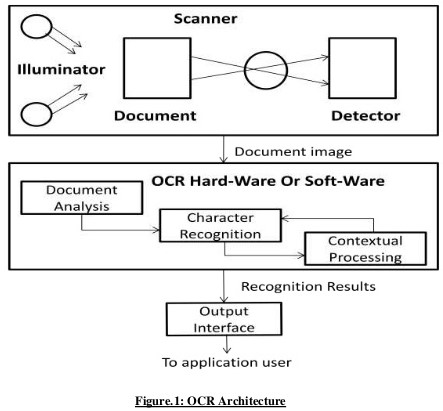
4.<https://docs.oracle.com/javase/tutorial/>

5.https://developer.android.com

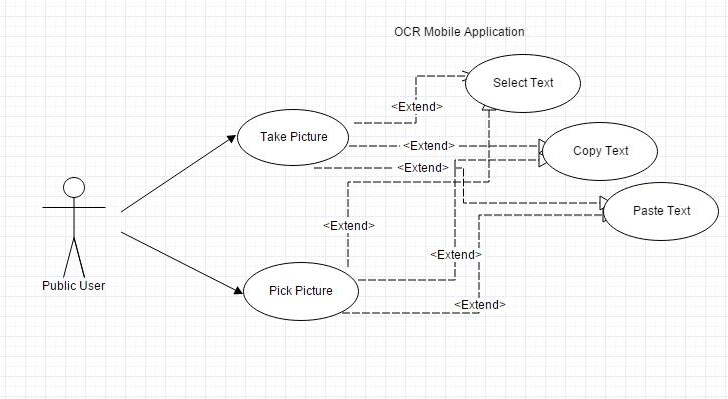
.

**1.5 Glossary**

Optical Character Recognition (OCR) is a technology that allows dots or pixels representing machine generated characters in a [raster image](http://www.digitizationguidelines.gov/term.php?term=rasterimage) to be converted into digitally coded text. In addition to recognizing and coding text, OCR programs attempt to recognize and code the structural elements of a document page, such as columns and non-text graphical elements. Intelligent Character Recognition [(ICR)](http://www.digitizationguidelines.gov/term.php?term=ICR) is a related technology designed to recognize handwritten characters.OCR is generally part of a workflow that begins with the scanning documents. Scanned images may be further processed or "cleaned" (for example, see [contrast stretching](http://www.digitizationguidelines.gov/term.php?term=contraststretching)) prior to OCR to improve accuracy of the recognition process. Modern OCR applications are capable of producing multiple output formats such as ASCII, RTF, Microsoft Word or PDF. While some hardware applications for OCR exist, the vast majority of OCR is performed by software applications.



**1.6 Use Case Diagrams**

****

Use case Specification:

1. Scan Photo Use Case

Use case name: Take a Photo

Use case ID: 001

Description: The user takes a valid photo of the text that needs to be digitized

Actor: User of the application

Pre: Application is waiting for the user either to choose a photo from the gallery or to take one.

Main flow: 1. The use case starts when a user takes a photo

Post: the photo is taken , and ready to be used .

2. Pick a Photo

Use case name: Pick a Photo

Use case ID: 002

Description: The user chooses a photo that contains the text from the directory to be converted.

Actor: User of the application

Pre: Application is waiting for the user either to choose a photo from the directory or to take one.

Main flow: 1. The use case starts when a user chooses a photo

Post: the photo is chosen , and ready to be used .

3. Select text

Use case name: Select text

Use case ID: 003

Description: The user selects the text that was converted from the image.

Actor: User of the application

Pre: The application has processed the image and extracted its text to that can be selected.

Main flow: 1. The use case starts when a user takes or chooses a photo

2. The user gets the text from the application and selects it

Post: the text from the photo is selected.

4. Copy text

Use case name: Copy text

Use case ID: 004

Description: The user copies the text that was converted from the image.

Actor: User of the application

Pre: The User has selected the text or the part of the text that wants to copy

Main flow:

1. The use case starts when a user takes or chooses a photo

2. The user gets the text from the application and selects it

3. The user copies the text or part of the text that he\she needs

Post: the text from the photo is copied and ready to use

.

4. Paste text

Use case name: paste text

Use case ID: 005

Description: The user pastes the text that he\she selected and copied.

Actor: User of the application

Pre: The User either choosed or took a photo.

Main flow:

1. The use case starts when a user takes or chooses a photo

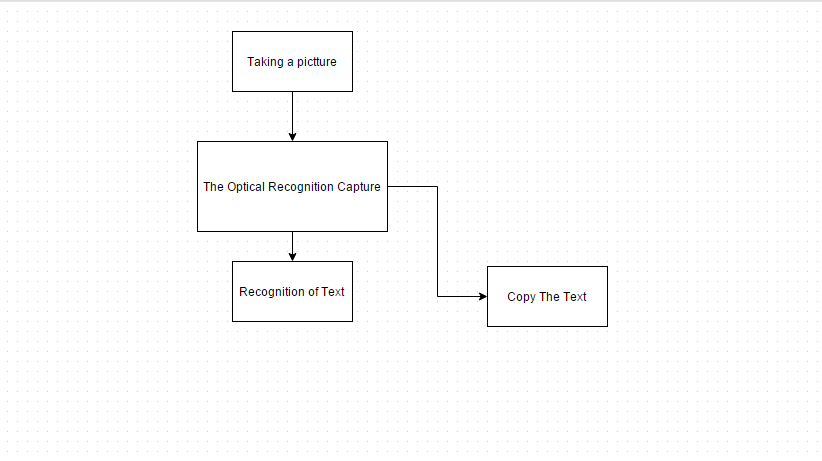
2. The application gets text from the selected photo

3. The user copies the test and pastes it

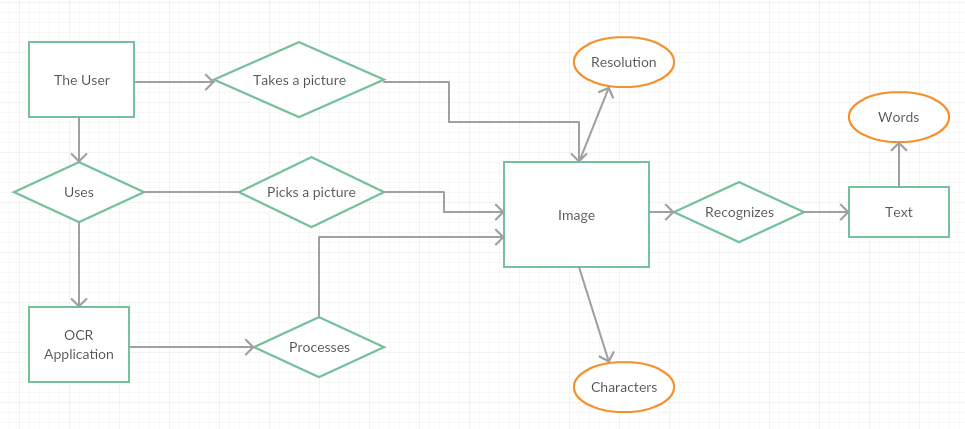
Post: the text from the photo is provided

**2 SYSTEM ARCHITECTURE**

At this level, we were able to identify the most important modules that we designed as follows:

The interface of the system is separated from its logic by using a Model View Controller. The Model contains the data while the interface constructs the view of the architecture. The controller matches between the model and the view. All communications between the model and the view will go through the controller. Moreover, the interface doesn't follow the logic of the application which makes the architecture of the system easier. 21 Also, for the user class which has many functions such as presscaptureImageBotton() and pressImportImageBotton() handles the interface's process without dealing with the application's logic. Then, the system's controller contains handlers for the system's functionality which is composed into modules. In other words, the handler of CaptureImage is used in the Capture image module, the handler of ImportImage is used inImport Picture module, and the RecognizeText is used in the UserApplication module. This controller calls the functions, but does not process them. Finally, the application's model processes the functions of all invoked functions. This model is made of CaptureImage, ImportImage, RecognizeText, and UserApplication classes that summarize the logic of the application. These classes have different functionalities and process requests of the aforementioned view and controller, retrieve the result, and give it back to the controller.

**2.2 Architecture Diagram**

****

How OCR works:

Generally OCR works as follows:

1. Pre-process image data, for example: convert to grayscale, smooth, de-skew, filter.
2. Detect lines, words and characters.
3. Produce a ranked list of candidate characters based on a trained data set. (here the setDataPath() method is used for setting path of trainer data)
4. Post process recognized characters, choose best characters based on confidence from previous step and language data. Language data includes dictionaries, grammar rules, etc.

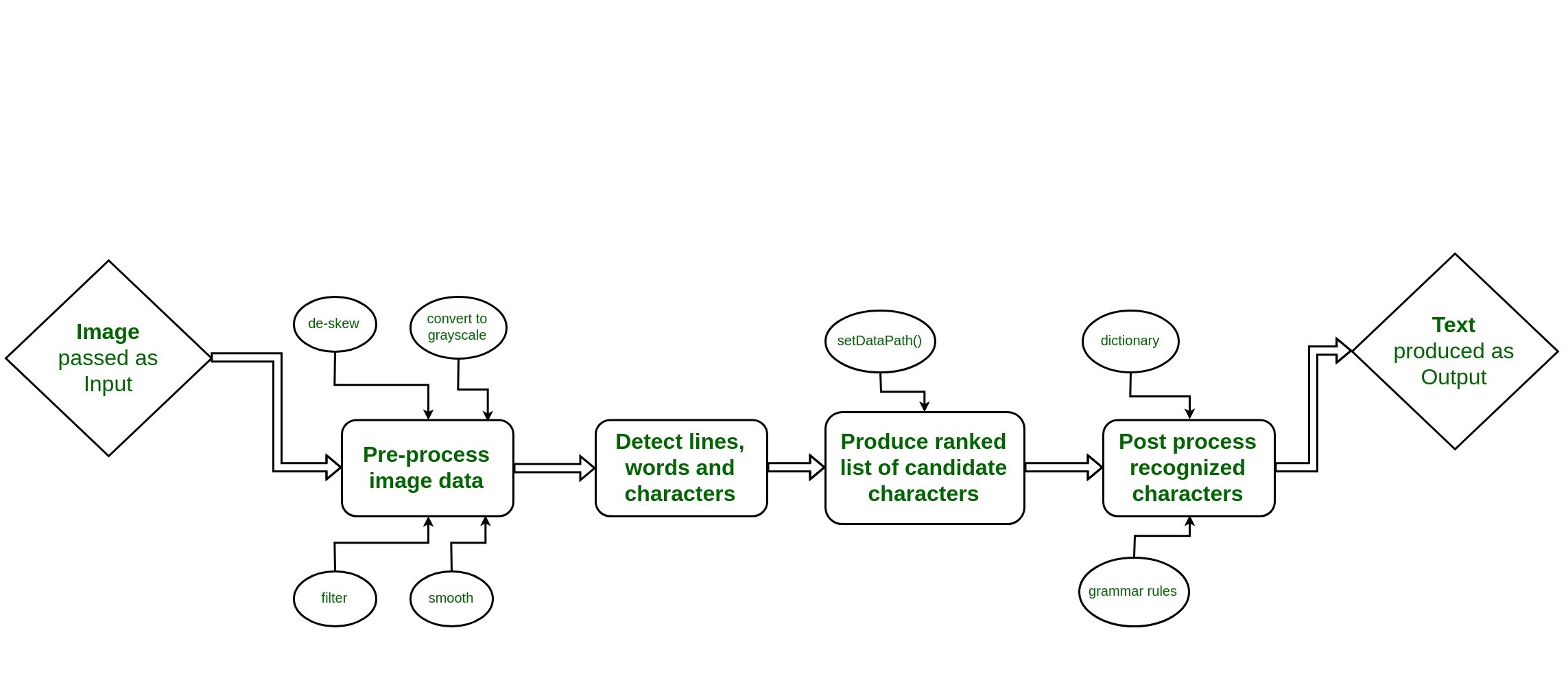


Fig:General working of OCR

.

**2.3 Tesseract OCR**

Tesseract OCR is an optical character reading engine developed by HP laboratories in 1985 and open sourced in 2005. Since 2006 it is developed by Google. Tesseract has Unicode (UTF-8) support and can recognize more than 100 languages “out of the box” and thus can be used for building different language scanning software also. Latest Tesseract version is Tesseract 4. It adds a new neural net (LSTM) based OCR engine which is focused on line recognition but also still supports the legacy Tesseract OCR engine which works by recognizing character patterns.

**3 REQUIREMENT SPECIFICATION**

As aforementioned, we need to retrieve text from scanned documents or any text image and make it editable to reuse it and read it word by word. For instance, there are plenty of books that are only available in printed format, so even if we scan them, they will be stored only as images. With the use of the Optical Capture Recognition (OCR), these scanned documents will be available for later editing and can be reused by the user.

After the feasibility study and a deep understanding of the functionalities of the project, and after looking at the tools that will be needed to develop and realize the application, My teammate and I have been able to collect the main user classes and the different requirements and classify them as follows:

**3.1 INTERFACE REQUIREMENT**

User: must use an Android mobile phone. He/she will take a picture of the desired text or choose one from the phone's directory. The OCR will ignore the non textual region of the picture and will print only the text. Also, the user has to follow the required steps in order to avoid any error while using the application.

The application will work as follow:

* Taking a picture using the phone's camera or choosing one from the phone's directory.
* Recognition of the text.
* Retrieving the text and making it editable.

With a simple click, the user can take advantage of this application and perform many actions in a few minutes. By using the mobile's camera, different text images can be scanned, copied, and saved.

**3.2 FUNCTIONS REQUIREMENT OF SYSTEM**

We have classified these functional requirements as follow:

1. Taking/ choosing the desired text image.

2. Recognition of the text.

3. Copying the text for different uses.

3.2.1. Taking/ choosing the desired text image:

Description:

For the mobile application:

The most important thing here is the use of an Android mobile phone and its camera. The user can take a picture of a text image or choose one from the mobile's directory.

The user must use a camera of typical resolution and take a picture of a text image or choose one from existing ones in his phone.

1: Android Mobile Phone.

2: Text Image.

3: Images containing text.

3.2.2. Recognition of the text:

Description:

14

The text will be recognized from the image taken by the mobile's camera or from any chosen image from the phone directory.

The text will be recognized and ready to be used.

1: Recognition of the text from the image.

2: Ready to be used.

3.2.3. Copying the text for different uses:

Description:

Once the text is recognized and ready to be used, the user will be able to copy, edit, and modify it. He/she may also be able to retrieve the data from the image and store it directly on the phone such as the contact information taken from a Business Card.

The recognized text may be retrieved to make it editable or store it directly on the phone.

1: Copy the text from the text from the image and modify it.

2: Retrieve data from the text image and store it on the phone

**4 HUMAN-MACHINE INTERFACE**

As we know OCR performs on the basis of image quality and sharpness and therefore for our concern we have to perform the OCR in different conditions such as follows:

1.OCR on clear image

2**.**.OCR on unclear image

**4.1 Performing OCR on clear Image**

Now that you have linked the jar file, we can get started with our coding part. The following code reads an image file and performs OCR and displays text on the console.

Code:

import java.io.File;

import net.sourceforge.tess4j.Tesseract;

import net.sourceforge.tess4j.TesseractException;

public class Test {

public static void main(String[] args)

{

Tesseract tesseract = new Tesseract();

try {

tesseract.setDatapath("D:/Tess4J/tessdata");

// the path of your tess data folder

// inside the extracted file

String text

= tesseract.doOCR(new File("image.jpg"));

// path of your image file

System.out.print(text);

}

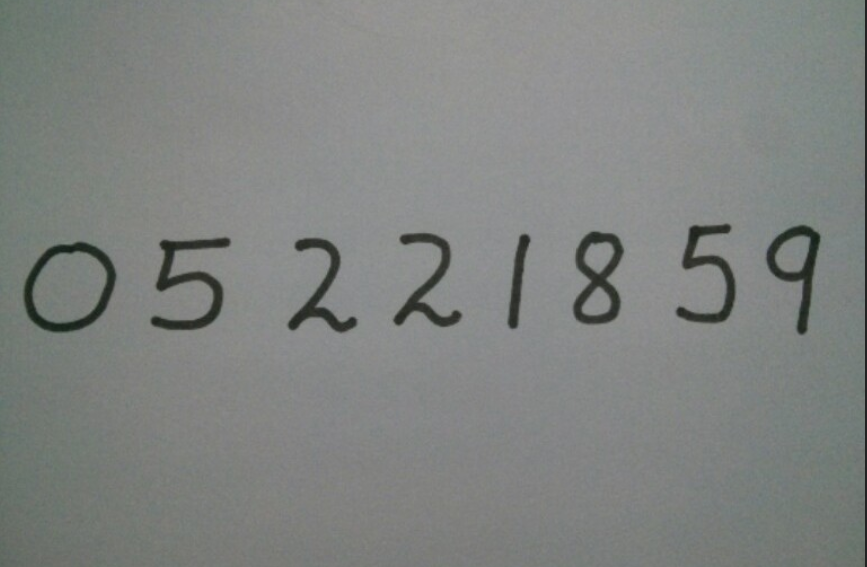
catch (TesseractException e) {

e.printStackTrace();

}

}

Input:



Output:*05221859*

**4.2 Performing OCR on unclear Image**

Tesseract works best when there is a very clean segmentation of the foreground text from the background. In practice, it can be extremely challenging to guarantee good segmentation. There are a variety of reasons you might not get good quality output from Tesseract if the image has noise on the background. Noise removal from images comes in the part of image processing. For this, we need to know in what way an image should be processed.

To implement the same in JAVA, we will make a small intelligence-based model which will scan the RGB content of the image and then convert it into the gray scaled content and also we will perform some zooming effect on the image too.

The below example is a sample code on how the image can be grayscaled based on its RGB content. So if images are very dark then they become brighter and clearer and if in case the images are whitish then they are scaled to little dark contrast so that text is visible.

Code:

import java.awt.Graphics2D;

import net.sourceforge.tess4j.\*;

import java.awt.Image;

import java.awt.image.\*;

import java.io.\*;

import javax.imageio.ImageIO;

public class ScanedImage {

public static void

processImg(BufferedImage ipimage,

float scaleFactor,

float offset)

throws IOException, TesseractException

{

// Making an empty image buffer

// to store image later

// ipimage is an image buffer

// of input image

BufferedImage opimage

= new BufferedImage(1050,

1024,

ipimage.getType());

// creating a 2D platform

// on the buffer image

// for drawing the new image

Graphics2D graphic

= opimage.createGraphics();

// drawing new image starting from 0 0

// of size 1050 x 1024 (zoomed images)

// null is the ImageObserver class object

graphic.drawImage(ipimage, 0, 0,

1050, 1024, null);

graphic.dispose();

// rescale OP object

// for gray scaling images

RescaleOp rescale

= new RescaleOp(scaleFactor, offset, null);

// performing scaling

// and writing on a .png file

BufferedImage fopimage

= rescale.filter(opimage, null);

ImageIO

.write(fopimage,

"jpg",

new File("D:\\Tess4J\\Testing and learning\\output.png"));

// Instantiating the Tesseract class

// which is used to perform OCR

Tesseract it = new Tesseract();

it.setDatapath("D:\\Program Files\\Workspace\\Tess4J");

// doing OCR on the image

// and storing result in string str

String str = it.doOCR(fopimage);

System.out.println(str);

}

public static void main(String args[]) throws Exception

{

File f

= new File(

"D:\\Tess4J\\Testing and learning\\Final Learning Results\\input.jpg");

BufferedImage ipimage = ImageIO.read(f);

// getting RGB content of the whole image file

double d

= ipimage

.getRGB(ipimage.getTileWidth() / 2,

ipimage.getTileHeight() / 2);

// comparing the values

// and setting new scaling values

// that are later on used by RescaleOP

if (d >= -1.4211511E7 && d < -7254228) {

processImg(ipimage, 3f, -10f);

}

else if (d >= -7254228 && d < -2171170) {

processImg(ipimage, 1.455f, -47f);

}

else if (d >= -2171170 && d < -1907998) {

processImg(ipimage, 1.35f, -10f);

}

else if (d >= -1907998 && d < -257) {

processImg(ipimage, 1.19f, 0.5f);

}

else if (d >= -257 && d < -1) {

processImg(ipimage, 1f, 0.5f);

}

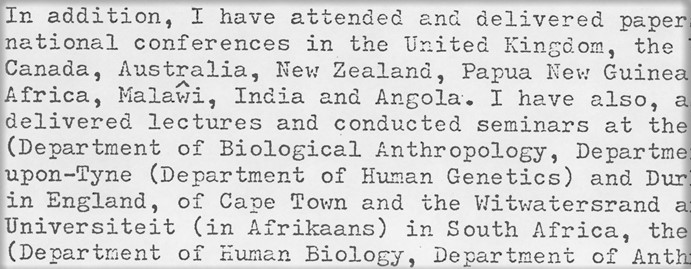
else if (d >= -1 && d < 2) {

processImg(ipimage, 1f, 0.35f);

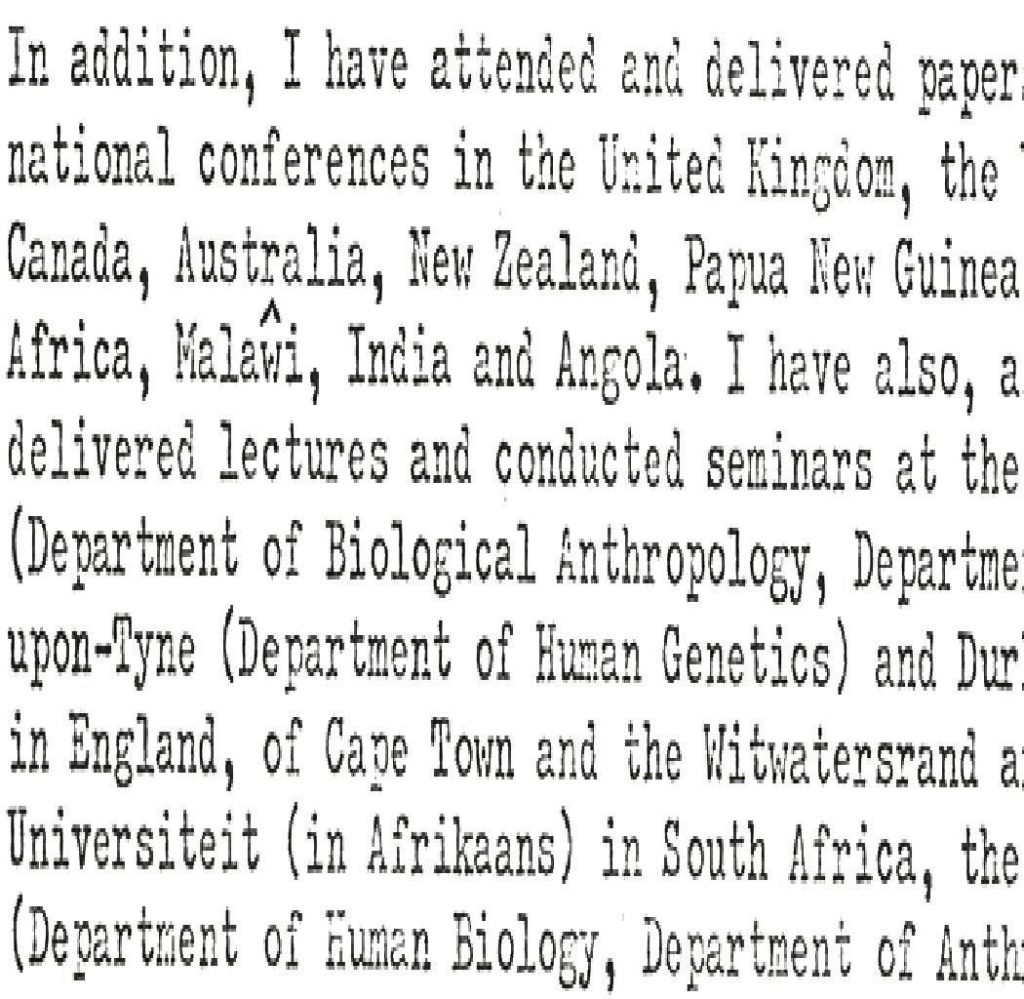
}

}}

Input:



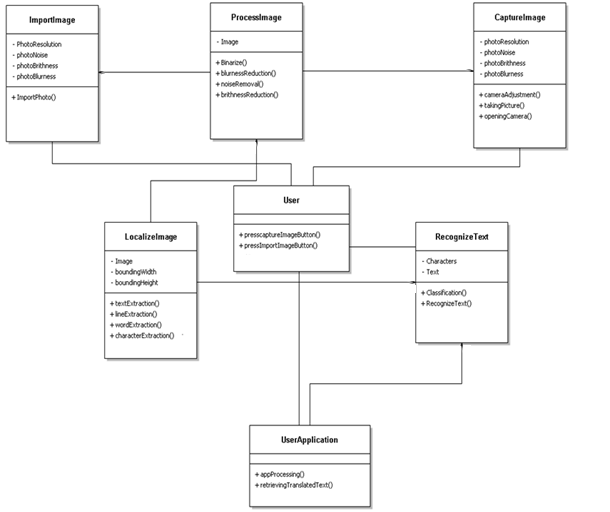
Output:



**5 DETAILED DESIGN**

After a deep understanding of the requirements and functionalities of the Optical Capture Recognition (OCR) application, and I have been able to define entities and their relationships and ended up with this following architecture diagram:

**5.1 Class Diagram**

****

After the specification of the requirements of the application and their analysis, my teammate and I have been able to get our class diagram. This above diagram characterizes the classes of the application. We ended up with seven classes that hold all the information required to build and run the application. This class diagram describes the relationships among classes and shows their attributes and methods. Moreover, this class diagram has been made after several meetings with our supervisor who helped us to end up with this above final class diagram.

**5.2 Class Description**

User: This class refers to the user of the application. This class contains the presscaptureImageButton() and pressImportImageButton() methods that invoke the CaptureImage and ImportImage classes respectively and that are responsible for the processes of each user's request.

CaptureImage: This class contains PhotoResolution, PhotoNoise, PhotoBrightness, and PhotoBlurness as attributes. These attributes are related to the photo taken by the user, meaning that the photo has to be taken by a high quality device with a high resolution. Also, the class contains cameraAdjustment(), takingPicture(), openingCamera() as methods that are exclusively responsible for the capturing of the picture.

ProcessImage: This class's attribute is the image. It is mainly responsible of the pre-processing of the image which is done by these following methods: Binarize(), blurnessReduction(), noiseRemoval, and brightnessReduction(). This processImage is done in order to get the image from the CaptureImage class that meets the needed criteria.

ImportImage: This class contains PhotoResolution, PhotoNoise, PhotoBrightness, and PhotoBlurness as attributes. These attributes are related to the photo imported or picked from the phone's directory or from the computer storage disk.. Also, the class contains ImportPhoto() as a method that solely performs the import of the picture from the computer or the phone's directory.

LocalizeImage: This class is responsible for localizing the text in the image imported or taken by the user. It contains the image, boundingWidth, and boundingHeight as attributes, meaning that the image must be in a certain range and contains text as well in order to be processed. Moreover, this class contains the textExtraction(), lineExtraction(), wordExtraction(), and characterExtraction() which are the methods that enable the localization of the text respectively. The image uses in this class is the one processed by the ProcessImage class.

RecognizeText: The attributes of this class are characters and text. It mainly matches the characters of the LocalizeImage and classifies them using the classification() method. In order words, it recognizes only Latin Alphabet which is sent to the UserApplication class by the method RecognizeText(), else it returns garbage data.

UserApplication: This class contains the end result of these aforementioned processes of the application which is handled by the appProcessing() method, and then this retrieved text is handled by the retrievingTranslatedText() method that sends it to the user.

**6 Software Component And Technology Used**

The software components and technology used in this project are:

1. Technology enablers:

Java as an object oriented programming language.

Android SDK( Software development Kit)

Android Studio

.

2. Operating Systems:

Windows

Android OS

3. Hardware Components:

Personal Computers

Android based mobile phone with high resolution.

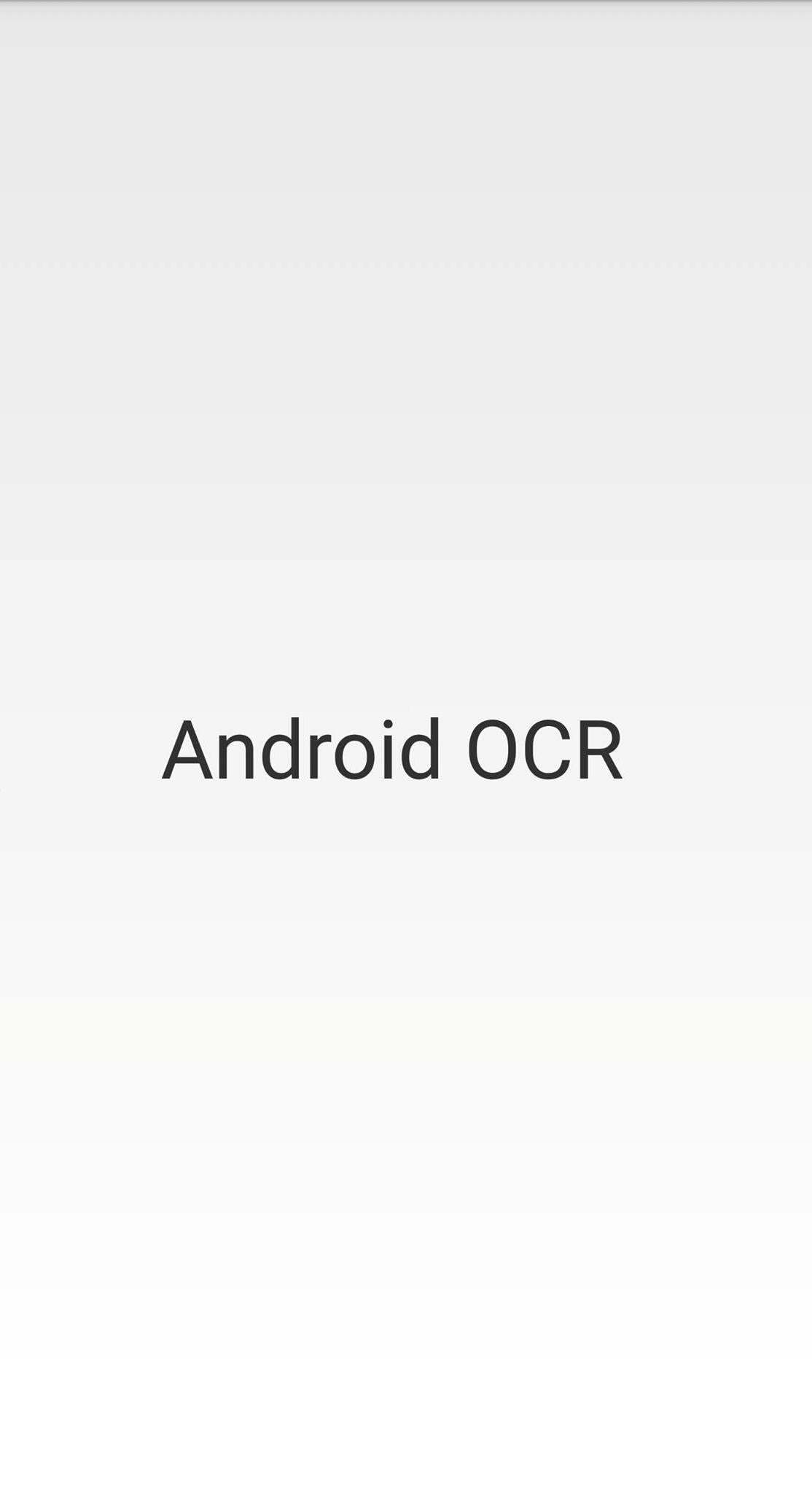
**7 Testing**

Testing is an important step that helps to detect errors. Testing is a process of finding faults that might occur during the implementation phase. It is also a way to test if the product fulfills the requirements and to check the components functionalities. There exists many ways of testing where each one of them has a distinct requirement, but the only testing that we made is the acceptance testing.

I have tested the application, Android based device, with the supervisor using acceptance testing strategy

**8 Implementation Result**

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



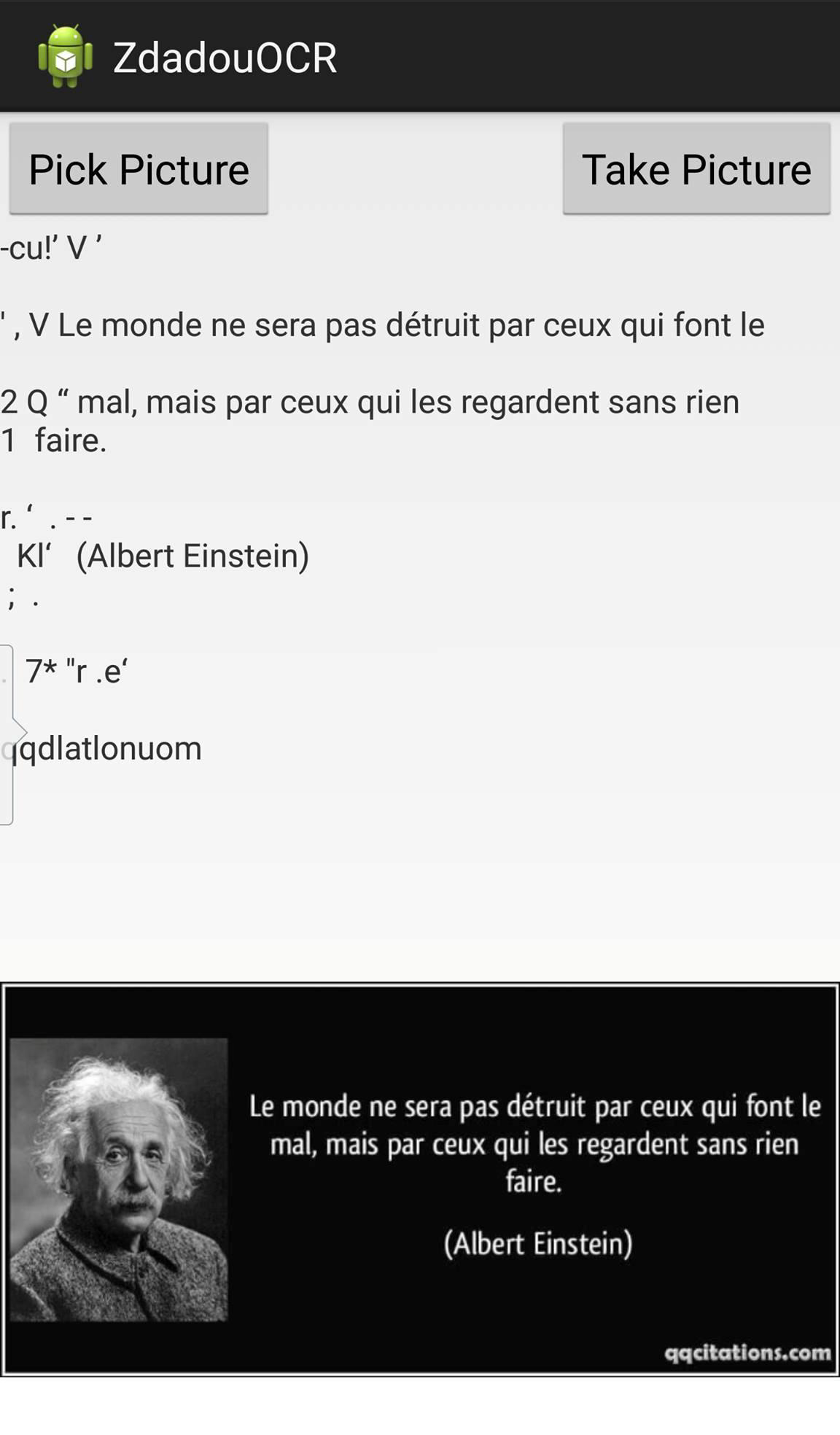
This above figure represents the Home page of the Optical Recognition Application.



This above figure represents the activities allowed to the user which are picking a picture from the phone's directory, or taking a picture by the phone.



This above figure shows the output of the application where the user picks a picture containing Latin alphabet.



This above picture shows the output of the application where the user picks a picture containing a text in French and it shows also the garbage data gotten from the non textual regions.