GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

OPTICAL CHARACTER RECOGNITION SYSTEM

A Project Report of Capstone Project 2

Submitted by

SIDDHANT GUPTA

(1613101735/ 16SCSE101546)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

Mrs. Vinny Sharma, Assistant Professor

APRIL / MAY- 2020

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report "OPTICAL CHARACTER RECOGNITION" is the bonafide work of "SIDDHANT GUPTA (1613101735)" who carried out the project work under my supervision.

SIGNATURE OF HEAD

School of Computing Science & Engineering

SIGNATURE OF SUPERVISOR

School of Computing Science & Engineering

ABSTRACT

The Optical Character Recognition (OCR) is one of the automatic identification techniques that fulfil the automation needs in various applications. A machine can read the information present in natural scenes or other materials in any form with OCR. The typed and printed character recognition is uncomplicated due to its well-defined size and shape. The handwriting of individuals differs in the above aspects. So, the handwritten OCR system faces complexity to learn this difference to recognize a character. In this report, we discussed the various stages in text recognition, handwritten OCR systems classification according to the text type as well as application oriented recent research in OCR.

TABLE OF CONTENT

TABLE OF FIGURES

INTRODUCTION

The goal of Optical Character Recognition (OCR) is to classify optical patterns (often contained in a digital image) corresponding to alphanumeric or other characters. The process of OCR involves several steps including segmentation, feature extraction, and classification. Each of these steps is a field unto itself, and is described briefly here in the context of a Matlab implementation of OCR.

One example of OCR is shown below

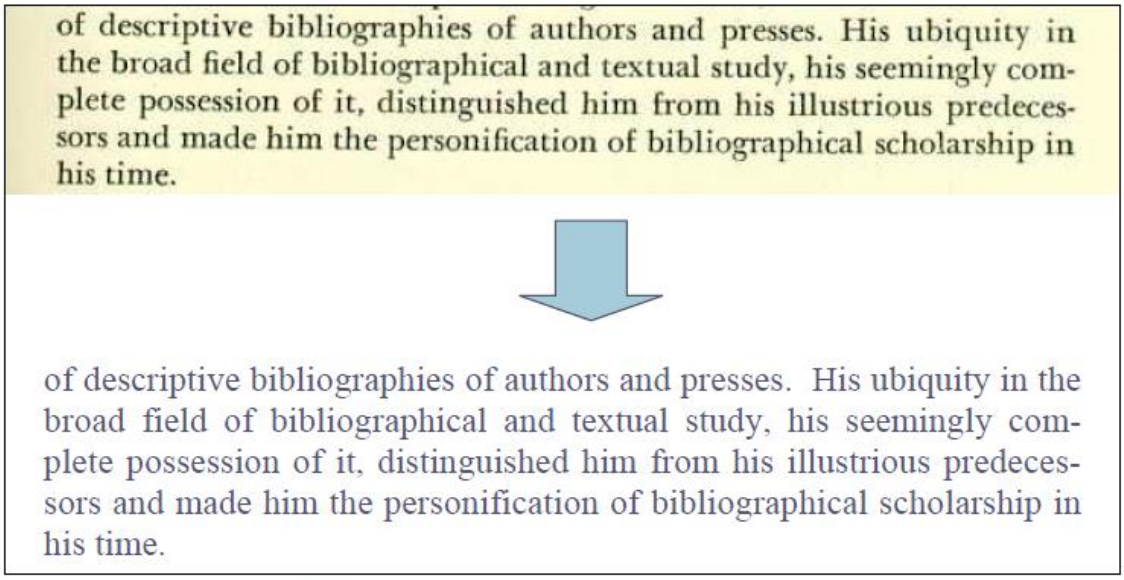

Figure 1: Scanned image of text and its corresponding recognized representation

Therefore, Text capture is a process to convert analogue text based resources into digitally recognisable text resources. These digital text resources can be represented in many ways such as searchable text in indexes to identify documents or page images, or as full text resources. An essential first stage in any text capture process from analogue to digital will be to create a scanned image of the page side. This will provide the base for all other processes. The next stage may then be to use a technology known as Optical Character Recognition to convert the text content into a machine readable format.

Optical Character Recognition (OCR) is a type of document image analysis where a scanned digital image that contains either machine printed or handwritten script is input into an OCR software engine and translating it into an editable machine readable digital text format (like ASCII text).

OCR works by first pre-processing the digital page image into its smallest component parts with layout analysis to find text blocks, sentence/line blocks, word blocks and character blocks. Other features such as lines, graphics, photographs etc are recognised and discarded. The character blocks are then further broken down into components parts, pattern recognized and compared to the OCR engines large dictionary of characters from various fonts and languages. Once a likely match is made then this is recorded and a set of characters in the word block are recognized until all likely characters have been found for the word block. The word is then compared to the OCR engine's large dictionary of complete words that exist for that language.

These factors of characters and words recognised are the key to OCR accuracy – by combining them the OCR engine can deliver much higher levels of accuracy. Modern OCR engines extend this accuracy through more sophisticated pre-processing of source digital images and better algorithms for fuzzy matching, sounds-like matching and grammatical measurements to more accurately establish word accuracy.

OVERALL DESCRIPTION

Optical Character Recognition, or OCR, is a technology that enables us to convert different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera or phone into editable and searchable data.

This technology is very useful since it saves time without the need of retyping the document. It can perform the action in few minutes. It is able to recognize text in images and convert it into editable text by going throughout a simplified process as illustrated in figure 1.

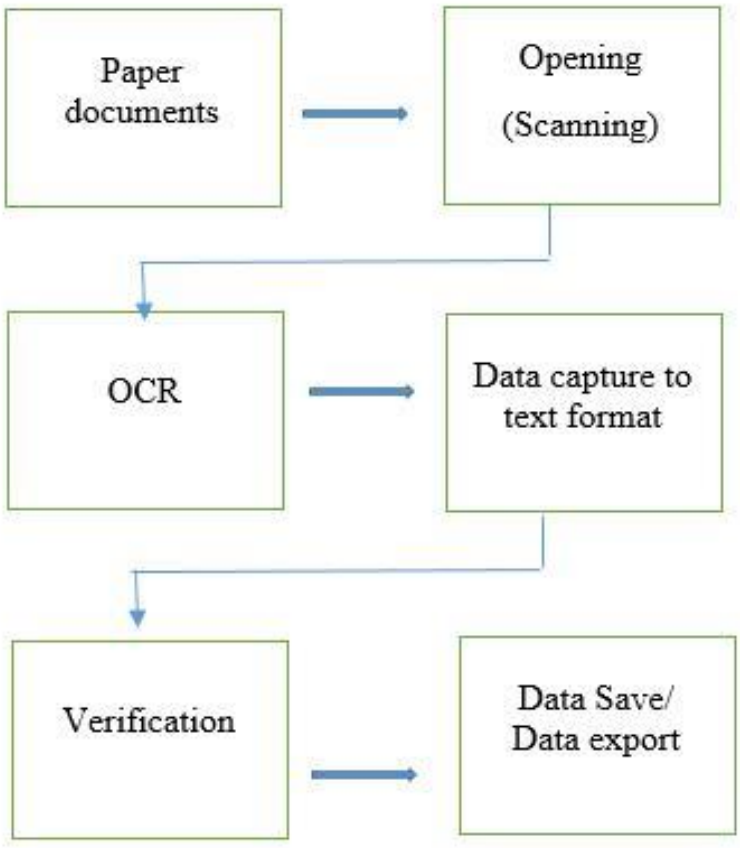

Figure 2: OCR Process

This process generally consists of three stages: Open (Scan) the document, Capture and recognize data and then Save in a convenient format.

Product Description:

OCR also called Optical Character Reader is a system that provides a full alphanumeric recognition of printed or handwritten characters at electronic speed by simply scanning the form. Forms containing characters images can be scanned through scanner and then recognition engine of the OCR system interpret the images and turn images of handwritten or printed characters into ASCII data (machine-readable characters).

Therefore, OCR allows users to quickly automate data capture from forms, eliminate keystrokes to reduce data entry costs and still maintain the high level of accuracy required in forms processing applications.

The technology provides a complete form processing and documents capture solution. Usually, OCR uses a modular architecture that is open, scalable and workflow controlled. It includes forms definition, scanning, image pre-processing, and recognition capabilities.

This project has two motivations, a desktop application and Android based mobile application.

Scanned documents, pictures stored in Android devices, and pictures taken by an Android device containing a text are scanned by this OCR. This application will allow its users to retrieve text from these documents to make it editable or to reuse it to perform other actions. For example, people can scan any ID Card, and store its data in their phones directly, or scan books or any important article found in a newspaper and retrieve the needed information and store them in their devices in few minutes.

Android Mobile Application Description:

Android mobile OCR is a software development kit (SDK) based on OCR technologies. Its powerful image processing algorithms enable mobile devices to perform highly accurate text recognition. Optimized for efficiency and featuring comprehensive language support, the application is ideal for wide range of individuals and users of android mobile phones who seek to convert their scanned paper documents, PDF files or images captured by their android mobile phones into editable and searchable data.

Using the code OCR technology, highly accurate text recognition functionality can be included in applications for tablets, Smart phones and other mobile devices using android. Mobile devices using iOS is out of our scope and can be done as a future work.

For the Android, it needs to use the procedures of the Android operating system. In other words, in order to have some tasks done, the system's implementation uses some libraries, so the application will interact with these libraries as well.

The Communication Description:

This application doesn't require any network connection. It will work as a device in-house application. This is very useful since sometimes people need to extract data from documents while internet is not working, so this application will allow them to so.

Main Actors

Public Users:

The application will be used to read texts and cards. For instance, it can be used to retrieve information from a business card and save it in the phone directly, or retrieve text from a text image taken by an Android device and store it to further reuse.

Moreover, this application will also facilitate retrieving text from books in order to edit it and to convert any images such as slides into notes. Also, instead of taking notes in class, students can directly take a picture of whatever it is written on the board and convert it to text and save it.

Finally, the application can also help its users such as foreigners to take a picture of any text image written in any language, retrieve it in order translate it to their native language in few minutes.

Constraints of the Implementation and Design:

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.

REQUIREMENT SPECIFICATIONS

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 on 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:

Interface requirements:

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:

1. Taking a picture using the phone's camera or choosing one from the phone's directory.
2. Recognition of the text.
3. Retrieving the text and make it editable.

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

Functional Requirements of the 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.

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.

a. Android Mobile Phone.

a. Text Image.

b. Images containing text.

Recognition of the text:

Description:

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.

- Recognition of the text from the image.
- Ready to be used.

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.

- Copy the text from the text from the image and modify it.
- Retrieve data from the text image and store it on the phone.

Non-Functional requirements:

After the functional requirements, My teammate and I have been able to classify the non-functional requirements as follows:

Product Requirements:

Usability Requirements:

The application shall be used friendly and doesn't require any guidance to be used. In other words, the application has to be as simple as possible, so its users shall use it easily. Actually, the interface is quite simple and straight forward so that anyone can understand it. The user should only click on the take or pick picture button and then directly click on the button extract text without any complications.

Reliability Requirements:

The application should not have any unexpected failure. In order to avoid any failure's occurrence, the specifications have been respected and followed correctly.

The only problem that may occur in some cases is that the application does not get 100% of the characters in the picture.

Efficiency Requirements:

Performance:

The application response time shall be adequate and sufficient enough, that's why the time required for this application to response to its user's actions has to been managed and controlled. But in order to maintain the performance of the application, the user has to follow the required steps to get the desired result.

Portability Requirements:

The application should be compatible with different version of Android, so if the version of Android is upgraded, the application should be upgraded as well.

Organizational Requirements:

Delivery Requirements:

I agreed with my client, to deliver the mobile based application by the beginning of December, and exactly on December 4th, 2015.

Implementation Requirements:

I used Java as a programming language for the implementation of the project.

Standards Requirements:

The application shall follow the AUI standard form.

External Requirements:

Ethical Requirements:

This application should protect the confidentiality of the user's personal information and any personal data stored on his\her mobile phone.

Legislative Requirements:

Security:

The security signature and certificate of the application is required as in any mobile application.

Privacy:

The application shall protect the user's data and make sure to keep it confidential. The device can be protected by a pin code or finger prints in order to ensure the privacy.

Attributes of the Software:

Maintainability:

The application shall respond to any change on the requirements.

Adaptability:

The application shall be compatible to any Android OS version.

Availability:

The application shall be available on the store whenever users want to download it.

Flexibility:

The architecture shall be flexible to any change of the requirements.

Analysis:

To analyze our requirements, my teammate and I used a use case diagram that will show the possible actors, use cases, and the relationships between the actors and the use cases and that is illustrated in figure 2

Use case Diagram:

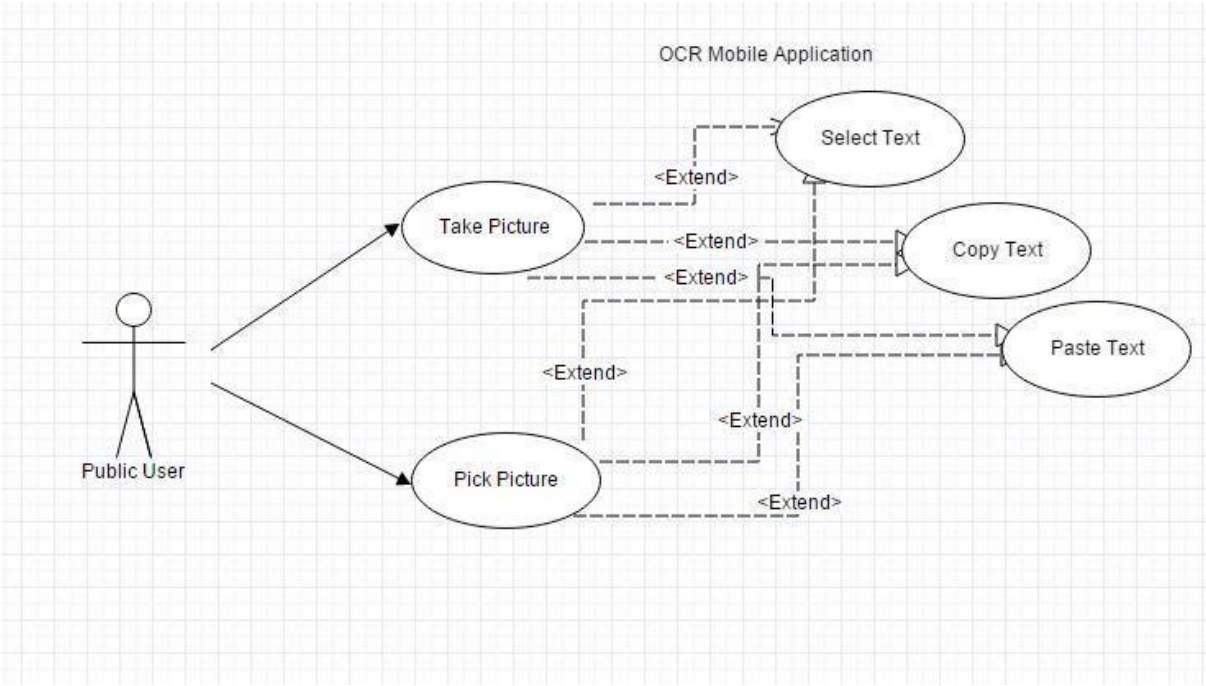

Figure 3: OCR Mobile Application Use Case Diagram

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 digitalized
Actor: User of the application
Pre: Application is wating 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 wating 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 choosen , 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: 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.

5. **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.	

APPLICATION DESIGN

The design of the application is highly important in order to fulfil the requirements and functionalities of the project.

Architecture of the System:

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

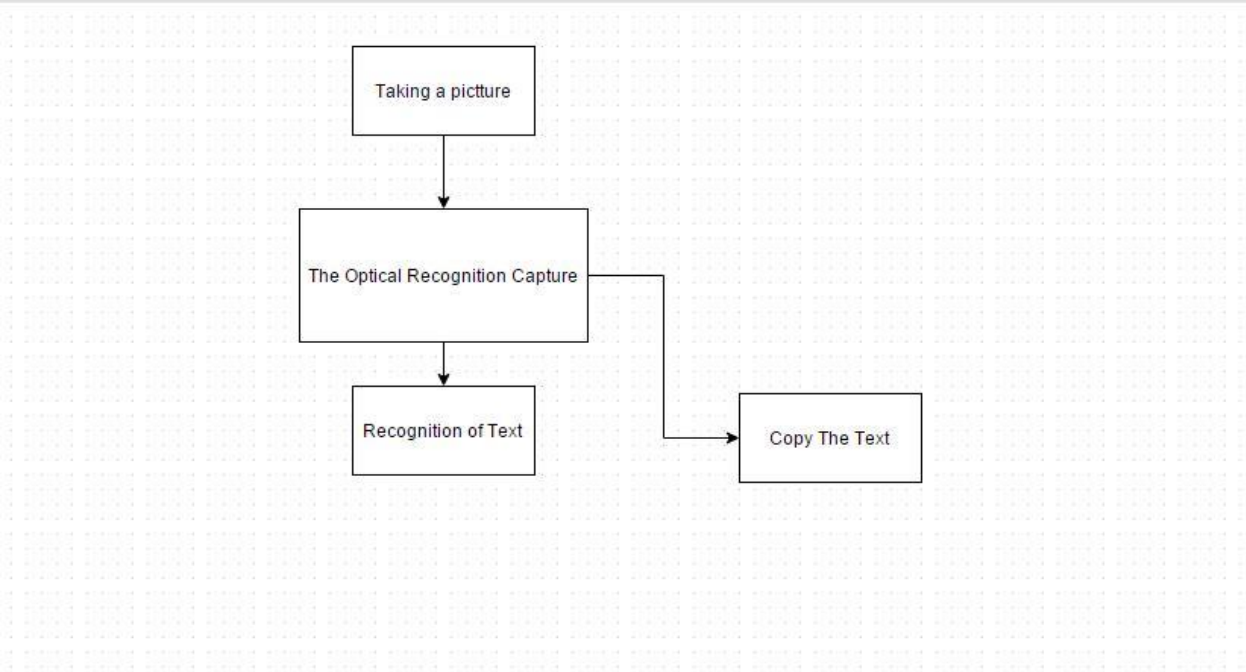

Figure 4: Structure design of OCR

The system is modelled 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.

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 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 do 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.

Architecture Diagram:

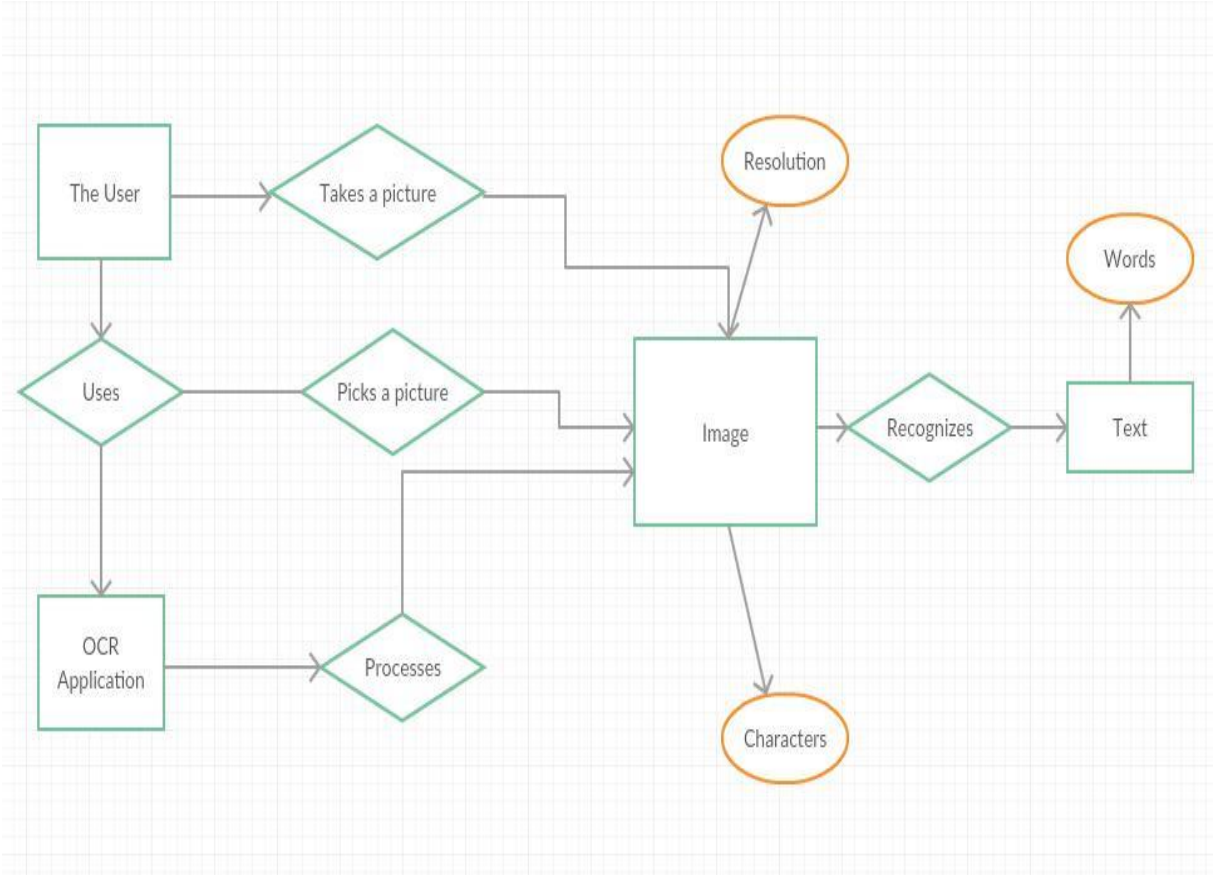

Figure 5: Architecture of OCR system

Description of the Attributes:

1. User: The one that will interacts with the Optical Capture Recognition (OCR).

2. The Image:

- **Taken:** must have a certain resolution and must contain text.
- **Selected from the phone:** must contain text.

3. The Text: must have words.

4. Words: English Alphabet.

5. Optical Capture Recognition Application.

Description of the relationships between the modules:

1. **User-OCR Application:** It is one to one relationship because only one user interacts with the application at a time.
2. **OCR Application-Image:** The application is able to process one image at a time.
3. **User-Image:** The user can either take a picture or choose one from the phone directory.
4. **Image-Text:** The image must contain text.
5. **Text-Words:** The text can contain many words.
6. **Words-Characters:** Each word can have any number of English Alphabet's characters.
7. **User-Text:** The user can copy, paste, or select the whole text or just a part of it.

Detailed Design:

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

Class Diagram:

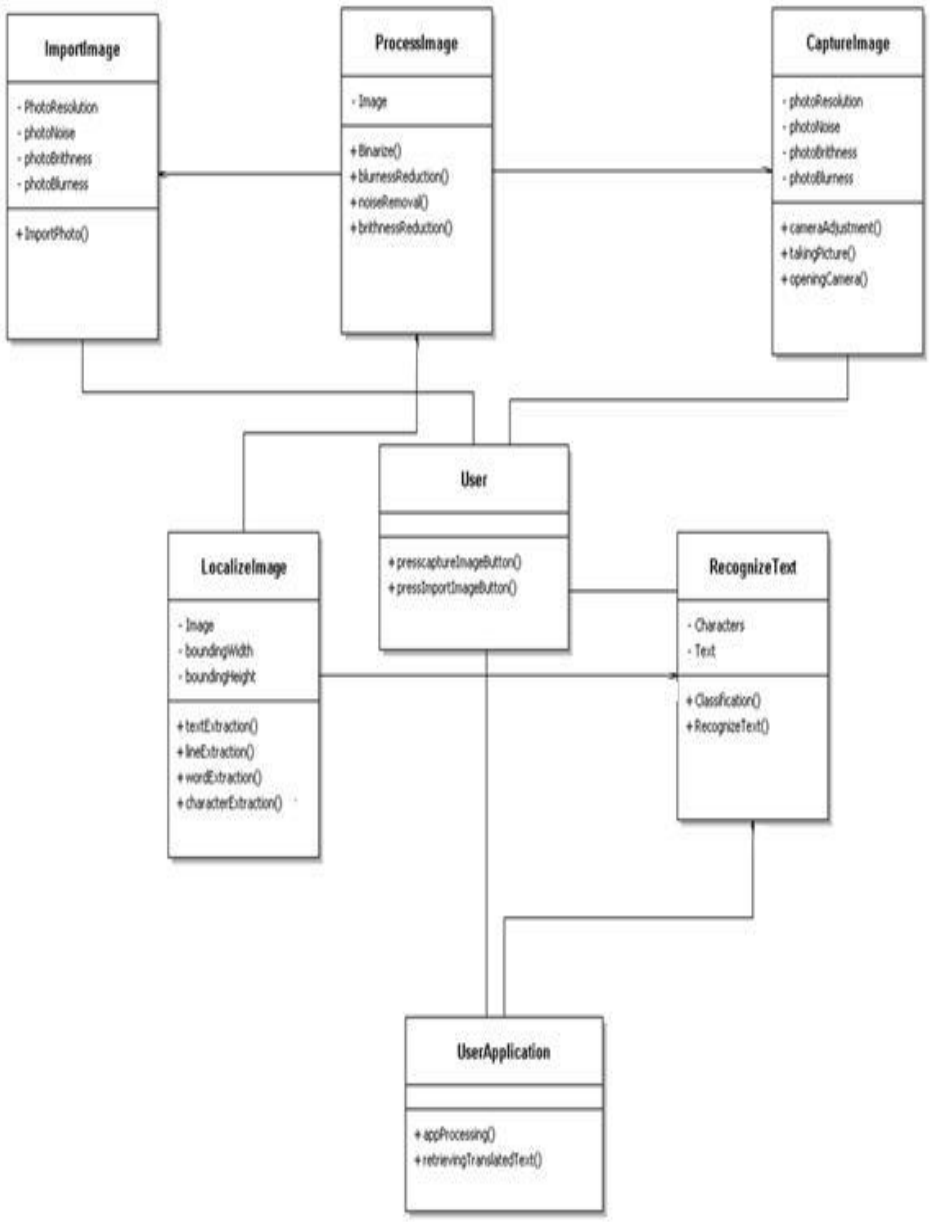

Figure 6: Class Diagram of OCR

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.

Classes 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 of 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 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 method that solely performs the import of the picture from the computer or the phone's directory.

LocalizeImage: This class is responsible for the 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 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 classify 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 is 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.

SOFTWARE COMPONENT AND TECHNOLOGY USED

The software components and technology used in this project are:

Technology enablers:

1. Java as an object oriented programming language.
2. Android SDK(Software development Kit)
3. Android Studio.

Operating Systems:

1. Windows
2. Android OS

Hardware Components:

1. Personal Computers
2. Android based mobile phone with high resolution.

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 fulfils 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.

IMPLEMENTATION RESULTS

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

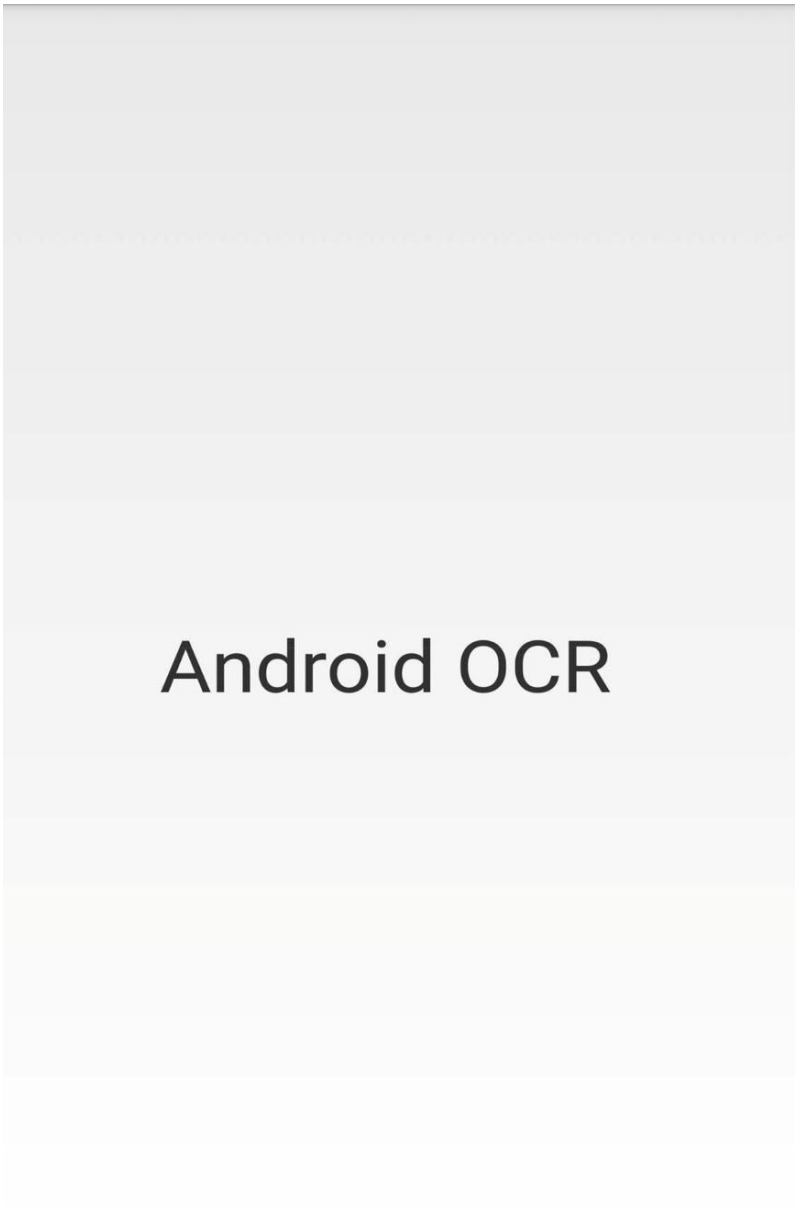

Figure 7

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

Figure 8

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.

Fig 9

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

Fig 10

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.

FUTURE WORK

During this period of the implementation of this capstone project, the Android based devices that I have implemented, I was only concentrated on the text images and documents without skew that contain Latin alphabet due to the time constraint. I could not further extend my work to make my application process text documents in Arabic or Berber as examples.

For the future, I will try to integrate other languages such as Hindi or any language that doesn't contain Latin alphabet. Also, I will concentrate also on the text with skew to ensure that the application will be efficient and employed in further uses.

CONCLUSION

This capstone project is the biggest and most important project that I worked on during our entire academic career. First, it is not only the final step to getting my bachelor degree, but also it is a great self-capacity-check as I stepped away from what I am used to work with to a new development field. During this period, which was less than 5 months, I got to work on a project that helped me improve my capacities in the computer science field especially that it is a different and new platform using Android. It also added a lot to the knowledge that I have accumulated throughout my academic journey at Galgotias University, it widened up my experience in dealing with different platforms, and enhanced my aptitudes to work with similar platforms in real life application which is going to work in our favor in future internships or jobs. Putting the technical part aside, this project had a positive impact on some of my personal traits. It upgraded our punctuality because I had to manage my time to provide the best work I can do in no later time than the deadline by sending our weekly dairies to my supervisor, Mrs. Vinny Sharma. Moreover, having to deal with constant pressure, I learnt to be more efficient and confident with what I am doing, and I managed to prevent stress from getting over me and pushing me to lose control over myself. To conclude, this project was a great experience in my academic career that pushed me on so many levels, technically, professionally, and personally. I am quite satisfied with the outcome that I ended up with, but none of it could have been possible without the help of my supervisor, Mrs. Vinny Sharma, whom I would love to thank extremely.

APPENDIX

Libraries Used:

```
package com.zdadou.app;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;

import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.Typeface;
import android.text.SpannableString;
import android.text.Spanned;
import android.text.style.StyleSpan;
import android.util.Log;
```

The Words Order:

```
public static int GetWordsIndex(int iPosInString)
{
    int i=0;
    int prev = 0;
    for (; i<m_asWordss.length; i++)
    {
        if (iPosInString >= prev && iPosInString < m_asWordss[i].m_iEnd)
            break;
        prev = m_asWordss[i].m_iEnd;
    }

    if (i == m_asWordss.length) return -1;
    return i;
}
```

Getting the picture:

```
public static int GetWordsIndex(int iPosInString)
{
    int i=0;
    int prev = 0;
    for (; i<m_asWordss.length; i++)
    {
        if (iPosInString >= prev && iPosInString < m_asWordss[i].m_iEnd)
            break;
        prev = m_asWordss[i].m_iEnd;
    }

    if (i == m_asWordss.length) return -1;
    return i;
}
```

The picture's filter (Get Parts):

```
public int bwFilterImage(byte[] bw_image, int width, int height, int bpp)
{


    final int THOLD_WHITE = 0x70;
    final int THOLD_BLACK = 0x30;
    final int BLK_SZ = width/16;

    final int RED_P = 3;
    final int GREEN_P = 6;
    final int BLUE_P = 1;

    if (bpp != 1)
    {
        int local_min = THOLD_WHITE;
        int local_max = THOLD_BLACK;
        int local = 0;

        for (int i=0; i<width * height; i++)
        {


    }


    return 1;

}
```

Division Algorithm:

```
public void SetImgDivisor(int d)
{
    if (d!=2 && d!=4)
        return;
    m_bImgDivisor = (byte)d;
}

public byte GetImgDivisor()
{
    return m_bImgDivisor;
}
```

Word Detection:

```
public final class Words
{
    String m_sBody;
    int m_iConfidence;
    boolean m_bIsValidWords;
    boolean m_bNewLine;
    int m_iEnd;


    public Words()
    {
        m_sBody = "";
        m_iConfidence = 0;
        m_bIsValidWords = false;
        m_bNewLine = false;
        m_iEnd = 0;
    }


    public Words(String body, int conf)
    {
        if (!mConfig.m_bReplaceUnknownChars)
            m_sBody = body;
        else
        {
            m_sBody = body.replaceAll("[\n\r]+", "");
            m_sBody = m_sBody.replaceAll("[^A-Za-z0-9!?.,%$@&*(){}<>:;'\"-]+", "#");
        }
        m_iConfidence = conf;

        if (m_sBody.contains("#"))
            m_bIsValidWords = false;
        else m_bIsValidWords = MyIsValidDictionaryWords(body);

        m_bNewLine = body.contains("\n");
        Log.v(TAG, "Words("+m_sBody+","+conf+")");
    }
}
```

Words Order:

```
public static int GetWordsIndex(int iPosInString)
{
    int i=0;
    int prev = 0;
    for (; i<m_asWordss.length; i++)
    {
        if (iPosInString >= prev && iPosInString < m_asWordss[i].m_iEnd)
            break;
        prev = m_asWordss[i].m_iEnd;
    }

    if (i == m_asWordss.length) return -1;
    return i;
}
```

User Settings:

```
public final static class OCRConfiguration
{
    public int m_iMinOveralConfidence;
    public int m_iMinWordsConfidence;
    public int m_iPSMMode = PM_Automatic;
    public boolean m_bUseBWFilter;
    public boolean m_bShowValidWordssOnly;
    public boolean m_bReplaceUnknownChars;

    private byte m_bImgDivisor = 2;
    private String m_sLanguage = "eng";
    public String m_asLanguages[] = null;


    public OCRConfiguration()
    {
        LoadFabricDefaults();
    }


    public void LoadFabricDefaults()
    {
        m_iMinOveralConfidence = 60;
        m_iMinWordsConfidence = 50;
        m_iPSMMode = OCR.PM_Automatic;
        SetImgDivisor(2);
        m_bUseBWFilter = false;
        m_bShowValidWordssOnly = false;
        m_bReplaceUnknownChars = true;
        m_sLanguage = "eng";
        Log.v(TAG, "LoadFabricDefaults(): " + m_sLanguage);
    }
```

Initialize and destruct OCR:

```
private OCR()
{
    String text = libVer();

    classInitNative();
    initializeNativeDataNative();

    mConfig.m_asLanguages = getLanguagesNative();

    Log.v(TAG, "OCR Initialize() done - libver "+ text + " no-langs-installed=" + mConfig.m_asLanguages.length);
}

public static void Initialize()
{
    if (m_OCRinstance == null)
    {
        m_OCRinstance = new OCR();
    }
}


public static OCR get()
{
    return m_OCRinstance;
}


public void Destructor()
{
    Log.v(TAG, "OCR - finalize");
    if (m_bIsLibraryInitialized)
        ClearOCR();
    m_bIsLibraryInitialized = false;
}
```

REFERENCES

[1] Android Developers. (n.d.). Retrieved November 20, 2015, from http://developer.android.com/

[2] Deitel, P. J. (2012). Android for Programmers: an App-driven approach. Upper Saddle River, NJ : Prentice Hall ; London : Pearson Education, c2012.

[3] Welcome to NewCircle. (n.d.). Retrieved November 20, 2015, from https://thenewcircle.com/

[4] Android 6.0 Marshmallow. (n.d.). Retrieved November 20, 2015, from https://developer.android.com

[5] Download. (n.d.). Retrieved November 20, 2015, from http://developer.android.com/sdk/index.html