DEVELOPMENT OF AN AUTOMATED PIN DETECTION APPLICATION USING OPTICAL CHARACTER RECOGNITION

**MBAKWE, CALEB CHUKWUEZUGO**

**CSC/2008/065**

SUPERVISED BY:   
DR. A. I. OLUWARANTI

Table of Contents

[CHAPTER ONE INTRODUCTION 1](#_Toc387614130)

[1.1 Background of study 1](#_Toc387614131)

[1.2 Statement of Problem 2](#_Toc387614132)

[1.3 Aim 2](#_Toc387614133)

[1.4 Specific Objectives 2](#_Toc387614134)

[1.5 Methodology Overview 3](#_Toc387614135)

[1.6 Scope of the Problem 3](#_Toc387614136)

[1.7 Justification 3](#_Toc387614137)

[CHAPTER TWO LITERATURE REVIEW 4](#_Toc387614138)

[2.1 Introduction 4](#_Toc387614139)

[2.2 Background 4](#_Toc387614140)

[2.2.1 Pattern Recognition 4](#_Toc387614141)

[2.2.2 Optical Character Recognition 6](#_Toc387614142)

[2.2.3 Personal Identification Numbers 9](#_Toc387614143)

[2.3 Methods of OCR 9](#_Toc387614144)

[2.3.1 Components of an OCR system 10](#_Toc387614145)

[2.4 Related works 13](#_Toc387614146)

[2.4.1 Prizmo 13](#_Toc387614147)

[2.4.2 Mobile OCR Free 14](#_Toc387614148)

[2.4.3 Scan to Text 14](#_Toc387614149)

[CHAPTER 3 METHODOLOGY 16](#_Toc387614150)

[3.1 Introduction 16](#_Toc387614151)

[3.2 Planning 16](#_Toc387614152)

[3.2.1 The Android Platform 16](#_Toc387614153)

[3.3 Analysis 17](#_Toc387614154)

[3.3.1 System Specification 17](#_Toc387614155)

[3.3.2 Assumptions on which the system is built 17](#_Toc387614156)

[3.3.3 Requirements: 18](#_Toc387614157)

[3.4 System Design 18](#_Toc387614158)

[3.4.1 Interface Design 18](#_Toc387614159)

[3.4.2 Activity Diagram 22](#_Toc387614160)

[3.4.3 Sequence Diagram 22](#_Toc387614161)

[3.5 Evaluation Techniques 25](#_Toc387614162)

[CHAPTER 4 DESIGN AND IMPLEMENTATION 26](#_Toc387614163)

[4.1 Introduction 26](#_Toc387614164)

[4.2 Software and Hardware Implementation 26](#_Toc387614165)

[4.2.1 Application Navigation service 27](#_Toc387614166)

[4.2.2 Camera Service 34](#_Toc387614167)

[4.2.3 Cloud Service 34](#_Toc387614168)

[4.2.4 Text To Speech Service (TTS) 35](#_Toc387614169)

[4.3 Evaluation 35](#_Toc387614170)

[4.3.1 Integration testing 35](#_Toc387614171)

[4.3.2 Alpha testing 35](#_Toc387614172)

[4.3.3 Beta Testing 35](#_Toc387614173)

[CHAPTER 5 CONCLUSION AND RECOMMENDATION 37](#_Toc387614174)

[5.1 Introduction 37](#_Toc387614175)

[5.2 Summary 37](#_Toc387614176)

[5.3 Contribution to Knowledge 37](#_Toc387614177)

[5.4 Challenges 37](#_Toc387614178)

[5.5 Recommendation 38](#_Toc387614179)

[5.6 Conclusion 38](#_Toc387614180)

[REFERENCES 39](#_Toc387614181)

[APPENDIX I 40](#_Toc387614182)

[APPENDIX II 41](#_Toc387614183)

# CHAPTER ONE INTRODUCTION

## Background of study

A common way of passing around confidential information such as Personal Identification Numbers (PINs) in our current age is through the use of cards with protective coverings often referred to as scratch cards. In situations where the PINs are intended for use on a computer device with a camera, it would be time saving time and an added convenience for the user if they could simply take a picture of the already scratched card. This project attempts to design and implement an android application that would achieve this.

A Personal Identification Number (PIN) can be described as a number allocated to an individual and used to validate electronic transactions. It is defined as a multi-digit number that is used by somebody to gain access to an account at an ATM, a computer, or a telephone system. (Microsoft Encarta, 2009). The PIN (often erroneously referred to as PIN number) concept was invented by James Goodfellow. In 2006, he was awarded an Officer of the Order of the British Empire (OBE) in the Queen's Birthday Honours. A PIN in the scope of this project is a secret numeric password shared between a user and a telecommunication network to facilitate the recharge or increment of the balance of a prepaid account. Typically, the user purchases a scratch card which possesses the confidential PIN and uses this PIN on his mobile device to recharge his prepaid account using some network specific code. Upon receiving the PIN, the telecommunication network system looks up the PIN in its database of PINs to ascertain its validity. The transaction is successful only when the number entered matches with a number stored in the system that is correct and has not been used by another user for a similar transaction. Hence, despite the name, a PIN does not personally identify the user.

Optical Character Recognition (OCR) is the identification of printed characters using photoelectric devices and computer software (as separate devices or together in one unit). It is defined as the use of light-sensing methods to identify printed and handwritten material and encode it in machine-readable form for inputting into a computer (Microsoft Encarta, 2009). OCR is the mechanical or electronic conversion of scanned images of handwritten, typewritten or printed text into machine-encoded text. It is widely used as a form of data entry from some sort of original paper data source, whether documents, sales receipts, mail, or any number of printed records. It is a common method of digitizing printed texts so that they can be electronically searched, stored more compactly, displayed on-line, and used in machine processes such as machine translation, text-to-speech and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision. The field of pattern recognition is still very much in its infancy, although in recent years some of the barriers that hampered such automated pattern recognition systems have been lifted through advances in computer hardware providing machines capable of faster and more complex computation (Brimblecombe, 2003).

OCR can be further elucidated as the process of examining printed characters on paper and determining their shapes by detecting patterns of dark and light. Once the scanner or reader has determined the shapes, character recognition methods—pattern matching with stored sets of characters—are used to translate the shapes into computer text. Sometimes OCR is done with special readers, but often it is done using a standard optical scanner and specialized software (Microsoft Encarta, 2009). Early versions of OCR needed to be programmed with images of each character, and worked on one font at a time. "Intelligent" systems with a high degree of recognition accuracy for most fonts are now common. Some systems are capable of reproducing formatted output that closely approximates the original scanned page including images, columns and other non-textual components.

## Statement of Problem

Optical Character Recognition is a greater use of a camera as it extracts text from physical materials but yet, the application areas of this field of computer science are neither pervasive nor ubiquitous. This project taps into the use of OCR to detect PINs in recharge cards using a mobile device with an intention of echoing this PIN to the users and using the detected PINs to carry out various possible functions such as recharging the account, sending to contacts, etc.

## Aim

This project is aimed to design and implement an android based application that uses Optical Character Recognition technology to detect Personal Identification Numbers in recharge cards, giving audio feedback of the PIN to the user and using the detected PIN to carry out other functions.

## Specific Objectives

The Objectives are stated as follows:

1. specify and design a computational model to capture PINs on recharge cards
2. implement a software that captures the PINs based on the model created in (a) and utilises the detected PIN for computations
3. evaluating the system in (b)

## Methodology Overview

1. Extensive review of the theories, process, computation and technologies underlying the Optical Character Recognition of PINs on cards
2. Design of a PIN detection Model using Unified Modelling Language (UML) tools
3. Implement the model designed in (b) using the Java programming language on the Android platform
4. Test the system using alpha and beta testing

## Scope of the Problem

This work is limited to the detection of recharge card PINs which consists of typeset Arabic numerals in English language hence excluding all other special characters and letters.

This is because of the following reasons:

1. OCR is still unstable for detection of handwritten characters
2. PINs do not contain special characters by their definition.
3. Recharge card pins (in Nigerian Network providers at least) only consist of digits.

## Justification

Study has shown that the Android mobile platform is the upcoming giant in the mobile devices industry. It is expected that most people will possess android phones. Software that utilises Optical Character Recognition to fasten the process of obtaining PINs from recharge cards is not available on the android platform as confirmed by the android application hive, Google Play, hence this study.

# CHAPTER TWO LITERATURE REVIEW

## 2.1 Introduction

This chapter consists of the following:

1. A background study of the research work that have been done on the topical concepts involved in this project
2. A review of the concept and methods of Optical Character Recognition
3. A Review of Existing Optical Character Recognition Applications and Projects

## 2.2 Background

The topical concepts in this project are:

* 1. Pattern Recognition (PR)
  2. Optical Character Recognition (OCR)
  3. Personal Identification Numbers (PINs)

### 2.2.1 Pattern Recognition

Character Recognition is an Application area of the wider Computer Science Field known as Pattern Recognition. The word pattern is derived from the same root as the word patron and, in its original use, means something which is set up as a perfect example to be imitated (Adewumi, 2012). There are many definitions for the term pattern recognition. A few are pointed out below:

Pattern Recognition can be defined as the categorization of input data into identifiable classes via the extraction of significant features or attributes of the data from a background of irrelevant detail (Gonzalez and Thomason, 1978). In addition, Pattern Recognition is the science that concerns the description or classification (recognition) of measurements." (Adewumi, 2012)

Pattern recognition algorithms generally aim to provide a reasonable answer for all possible inputs and to do "fuzzy" matching of inputs. This is opposed to pattern matching algorithms, which look for exact matches in the input with pre-existing patterns. A common example of a pattern-matching algorithm is regular expression matching, which looks for patterns of a given sort in textual data and is included in the search capabilities of many text editors and word processors. In contrast to pattern recognition, pattern matching is generally not considered a type of machine learning, although pattern-matching algorithms (especially with fairly general, carefully tailored patterns) can sometimes succeed in providing similar-quality output to the sort provided by pattern-recognition algorithms.

Pattern recognition is studied in many fields, including psychology, psychiatry, and ethnology, cognitive science, and traffic flow and computer science. Pattern recognition is generally categorized according to the type of learning procedure used to generate the output value. Supervised learning assumes that a set of training data (the training set) has been provided, consisting of a set of instances that have been properly labelled by hand with the correct output. A learning procedure then generates a model that attempts to meet two sometimes conflicting objectives: To perform as well as possible on the training data, and generalize as well as possible to new data. Unsupervised learning, on the other hand, assumes training data that has not been hand-labelled, and attempts to find inherent patterns in the data that can then be used to determine the correct output value for new data instances. A combination of the two that has recently been explored is semi-supervised learning, which uses a combination of labelled and unlabelled data (typically a small set of labelled data combined with a large amount of unlabelled data. In cases of unsupervised learning, there may be no training data at all to speak of; in other words, the data to be labelled is the training data.

#### Components of a Pattern Recognition System

The components of a pattern recognition system are discussed as follows:

1. **Data Acquisition:** One of the most important requirements for designing a successful pattern recognition system is to have adequate and representative training and test datasets. Adequacy ensures that a sufficient amount of data exists. Representative data, on the other hand, ensures that all meaningful variations of field data instances that the system is likely to see are sampled by the training and test data.
2. **Pre-processing:** The goal here is to condition the acquired data such that noise from various sources is removed to the greatest extent that is possible.
3. **Feature Extraction:** The goal of feature extraction is to find a preferably small number of features that are particularly distinguishing or informative for the classification process, and that are invariant to irrelevant transformations of the data.
4. **Feature Selection:** By feature selection, Polikar (2006) refers to selecting a subset of features from a set of features that have already been identified by a preceding feature extraction algorithm. The main question to answer under this setting is then ‘‘which subset of features provide the most discriminatory information?’’

#### Pattern Recognition Technologies

This author gathers that various technologies for pattern recognition exist, and these technologies cover needs for different areas of application as sourced from Line (1993).

The different techniques of automatic pattern recognition are:

* Speech Recognition
* Vision systems e.g. Facial recognition
* QR Codes
* Bar Code
* Optical Mark Reading (Recognition)
* Optical Character Recognition

### 2.2.2 Optical Character Recognition

Optical character recognition belongs to the family of techniques performing automatic pattern recognition. Optical Character Recognition deals with the problem of recognizing optically processed characters. Optical recognition is performed off-line after the writing or printing has been completed, as opposed to on-line recognition where the computer recognizes the characters as they are drawn. Both hand printed and printed characters may be recognized, but the performance is directly dependent upon the quality of the input documents. The more constrained the input is, the better will the performance of the OCR system be. However, when it comes to totally unconstrained handwriting, OCR machines are still a long way from reading as well as humans. However, the computer reads fast and technical advances are continually bringing the technology closer to its ideal.

#### The History of OCR

The first true OCR reading machine was installed at Reader’s Digest in 1954. This equipment was used to convert typewritten sales reports into punched cards for input to the computer. The commercial OCR systems appearing in the period from 1960 to 1965 may be called the first generation of OCR. This generation of OCR machines were mainly characterized by the constrained letter shapes read.

The reading machines of the second generation appeared in the middle of the 1960’s and early 1970’s. These systems were able to recognize regular machine printed characters and also had hand-printed character recognition capabilities. When hand-printed characters were considered, the character set was constrained to numerals and a few letters and symbols. The first and famous system of this kind was the IBM 1287.

In 1966, a thorough study of OCR requirements was completed and an American standard OCR character set was defined; OCR-A. This font was highly stylized and designed to facilitate optical recognition, although still readable to humans. A European font was also designed, OCR-B, which had more natural fonts than the American standard.

For the third generation of OCR systems, appearing in the middle of the 1970’s, the challenge was documents of poor quality and large printed and hand-written character sets.

Table 2.1 shows the timeline that represent the different milestones and chronology of Optical Character Recognition.

#### Applications of OCR

According to Line (1993), OCR application areas can be categorically distinguished under the following:

1. data entry
2. text entry
3. process automation

##### *i.* Data Entry

This area covers technologies for entering large amounts of restricted data. Initially such document reading machines were used for banking applications. The systems are characterized by reading only an extremely limited set of printed characters, usually numerals and a few special symbols. They are designed to read data like account numbers, customers’ identification, article numbers, amounts of money etc. The paper formats are constrained with a limited number of fixed lines to read per document.

##### ii. Text Entry

The second branch of reading machines is that of page readers for text entry, mainly used in office automation. Here the restrictions on paper format and character set are exchanged for

Table 2.1: A short OCR chronology (Line, 1993)

|  |  |
| --- | --- |
| **Year (Range)** | **Comments** |
| 1870 | The very first attempts |
| 1940 | The modern version of OCR |
| 1950 | The first OCR machines appear |
| 1960 – 1965 | First Generation OCR |
| 1965 – 1975 | Second Generation OCR |
| 1975 – 1985 | Third Generation OCR |
| 1968 -> | OCR to the people |

constraints concerning font and printing quality. The reading machines are used to enter large amounts of text, often in a word processing environment.

##### iii. Process Automation

Within this area of application the main concern is not to read what is printed, but rather to control some particular process. This is the application of OCR that this project work intends to adopt that is using the optically recognised PIN as input to actuate other computations such as recharging the user’s phone account, sending the PINs to contacts, etc.

##### *iv.* Other applications

Other areas of applications exist; these are not as rampant and ubiquitous as the previously mentioned areas. A few of these are mentioned below:

* Aid for blind
* Automatic number-plate readers
* Automatic cartography
* Form readers
* Signature verification and identification

### 2.2.3 Personal Identification Numbers

It is extremely important that the PIN be kept private, so that no one other than the account owner can use it. (IBM Corporation, 2010) To make sure that the integrity of this data is protected throughout all transactions, the International Organisation for Standardization (ISO) has technically revised and updated the standard providing requirements for the management and security of PINs (ISO 9564-1). The ISO standard for PIN management helps protect the identification numbers used for cardholder verification against unauthorized disclosure, compromise and misuse everywhere in the world. It thus helps minimize the risk of fraud through electronic funds transfer systems.

## 2.3 Methods of OCR

The main principle in automatic recognition of patterns is first to teach the machine which classes of patterns that may occur and what they look like. In OCR the patterns are letters, numbers and some special symbols like commas, question marks etc., while the different classes correspond to the different characters. The teaching of the machine is performed by showing the machine examples of characters of all the different classes. Based on these examples the machine builds a prototype or a description of each class of characters.

Then, during recognition, the unknown characters are compared to the previously obtained descriptions, and assigned the class that gives the best match.

### 2.3.1 Components of an OCR system

A typical OCR system consists of several components. In Figure 2.1 a common setup is illustrated. The first step in the process is to digitize the analogue document using an optical scanner. When the regions containing text are located, each symbol is extracted through a segmentation process. The extracted symbols may then be pre-processed, eliminating noise, to facilitate the extraction of features in the next step.

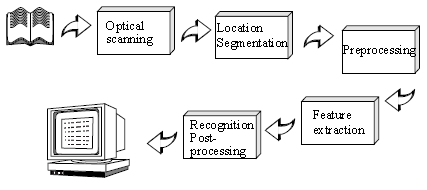
The identity of each symbol is found by comparing the extracted features with descriptions of the symbol classes obtained through a previous learning phase. Finally contextual information is used to reconstruct the words and numbers of the original text.

#### i. Optical Scanning

Through the scanning process a digital image of the original document is captured. In OCR, optical scanners are used, which generally consist of a transport mechanism plus a sensing device that converts light intensity into grey-levels. Printed documents usually consist of black print on a white background. Hence, when performing OCR, it is common practice to convert the multilevel image into a bi-level image of black and white. Often this process, known as thresholding, is performed on the scanner to save memory space and computational effort.

#### ii. Location and segmentation

Segmentation is a process that determines the constituents of an image. It is necessary to locate the regions of the document where data have been printed and distinguish them from figures and graphics. For instance, when performing automatic mail-sorting, the address must be located and separated from other print on the envelope like stamps and company logos, prior to recognition.

  
Figure 2.1: Components of an OCR-system

Applied to text, segmentation is the isolation of characters or words. The majority of optical character recognition algorithms segment the words into isolated characters which are recognized individually. Usually this segmentation is performed by isolating each connected component that is, each connected black area. This technique is easy to implement, but problems occur if characters touch or if characters are fragmented and consist of several parts. The main problems in segmentation may be divided into four groups:

* Extraction of touching and fragmented characters
* Distinguishing noise from text
* Mistaking graphics or geometry for text
* Mistaking text for graphics or geometry

#### iii. Pre-processing

The image resulting from the scanning process may contain a certain amount of noise. Depending on the resolution on the scanner and the success of the applied technique for thresholding, the characters may be smeared or broken. Some of these defects, which may later cause poor recognition rates, can be eliminated by using a pre-processor to smooth the digitized characters.

The smoothing implies both filling and thinning. Filling eliminates small breaks, gaps and holes in the digitized characters, while thinning reduces the width of the line. The most common techniques for smoothing move a window across the binary image of the character, applying certain rules to the contents of the window.

In addition to smoothing, pre-processing usually includes normalization. The normalization is applied to obtain characters of uniform size, slant and rotation. To be able to correct for rotation, the angle of rotation must be found. For rotated pages and lines of text, variants of Hough transform are commonly used for detecting skew. However, to find the rotation angle of a single symbol is not possible until after the symbol has been recognized.

#### iv. Feature extraction

The objective of feature extraction is to capture the essential characteristics of the symbols, and it is generally accepted that this is one of the most difficult problems of pattern recognition. The most straight forward way of describing a character is by the actual raster image. Another approach is to extract certain features that still characterize the symbols, but leaves out the unimportant attributes.

#### v. Post Processing

Post Processing can be categorised into:

1. **Grouping**: The result of plain symbol recognition on a document is a set of individual symbols. However, these symbols in themselves do usually not contain enough information. Instead we would like to associate the individual symbols that belong to the same string with each other, making up words and numbers. The process of performing this association of symbols into strings is commonly referred to as grouping. The grouping of the symbols into strings is based on the symbols’ location in the document. Symbols that are found to be sufficiently close are grouped together.
2. **Error-detection and correction:** Even the best recognition systems will not give 100% correct identification of all characters, but some of these errors may be detected or even corrected by the use of context. There are two main approaches, where the first utilizes the possibility of sequences of characters appearing together. Another approach is the use of dictionaries, which has proven to be the most efficient method for error detection and correction. Given a word, in which an error may be present, the word is looked up in the dictionary. If the word is not in the dictionary, an error has been detected, and may be corrected by changing the word into the most similar word.

## 2.4 Related works

In this section, this writer discusses certain application that have successfully utilised OCR in different ways.

### 2.4.1 Prizmo

Prizmo is a dedicated OCR app that functions on the Apple Macintosh. Its sole purpose is to analyse the text of any scans and convert it into searchable text. It’s not just limited to plain text documents such as receipts and correspondence, Prizmo will even analyse old newspapers and magazines, book covers, pretty much anything with any shape, size and colour of text can be converted by Prizmo.

The Application to be created by the function of this project is to be implemented on the android platform. This has the advantages and disadvantages of mobile computing.

### 2.4.2 Mobile OCR Free

This application aims at turning your smartphone into a document scanner with character recognition (OCR). Mobile OCR will convert your scanned documents from your camera or photo album into a regular text.

This app is simple & reliable. Mobile OCR is compatible with more than 25 languages including : Bulgarian, Catalan, Czech, Chinese (Simplified), Chinese (Traditional), Danish, German, Greek, English, Finnish, French, Hebrew, Hindi, Croatian, Hungarian, Indonesian, Italian, Japanese, Korean, Latvian, Lithuanian, Dutch, Norwegian, Polish, Portuguese, Romanian, Russian, Slovakian, Slovenian, Spanish, Serbian, Swedish, Tagalog, Thai, Turkish, Ukrainian, Vietnamese.

Though the application developed by this project is anchored only by the English language, the application is still superior to many others as it extends the feat of OCR beyond mere process automation.

### 2.4.3 Scan to Text

This is an offline (does not require internet connection to work) Optical Text Recognition (OCR) application which can be used to convert text (either English or Russian ONLY! on a paper to editable digital text on your device. Extracted/converted text can later be edited if required, selected and saved to a clipboard and/or shared via any application installed on the device (e.g. translated with Google translate.)

Characteristics of Scan to Text include:

1. Text recognition is based on "tesseract" OCR.
2. In order to keep app size reasonably small, only English and Russian languages are supported.
3. App contains open source code released under Apache License, Version 2.0: <http://www.apache.org/licenses/LICENSE-2.0.html>

Tesseract is an OCR Engine that was developed at HP Labs between 1985 and 1995. It is now developed and maintained by Google. Tesseract is probably the most accurate open source OCR engine available. Combined with the Leptonica Image Processing Library it can read a wide variety of image formats and convert them to text in over 60 languages. It was one of the top 3 engines in the 1995 UNLV Accuracy test. Between 1995 and 2006 it had little work done on it, but since then it has been improved extensively by Google. It is released under the Apache License 2.0. Tesseract works on Linux, Windows (with VC++ Express or CygWin) and Mac OSX. It can also be compiled for other platforms, including Android and the iPhone.

# CHAPTER 3 METHODOLOGY

## 3.1 Introduction

This chapter gives a description of the proposed system and presents the design tools used to model the proposed system. The model evaluation techniques will also be discussed. This author attempts to follow the standard Systems Development Life Cycle (SDLC). The SDLC has a set of four fundamental phases: planning, analysis, design, and implementation. Each phase is itself composed of a series of steps, which rely upon different techniques that produce different results.

## 3.2 Planning

The planning phase is the fundamental process of understanding why a system should be built and determining how the project will be built. Adequate planning is required in order to successfully design and implement the project. Some fundamental measures have to be put in place in order to build the required knowledge base and technical skills to accomplish the project. These include the collection of data and research into the field of Optical Character Recognition on the Android platform.

### 3.2.1 The Android Platform

Android has similar concepts to desktop applications but packaged differently and structured to make phones more crash-resistant. The building block of the android user interface is the activity. You can think of an activity as being the Android analogue for the window or dialog in a desktop application. While it is possible for activities to not have a user interface, most likely such code will be packaged in the form of content providers or services or hidden from the user as intents.

Content providers provide a level of abstraction for any data stored on the device that is accessible by multiple applications. Intents are system messages, running around the inside of the device, notifying applications of various events, from hardware state changes (e.g., an SD card was inserted), to incoming data (e.g., an email arrived), to application events (e.g., your activity was launched from the device’s main menu). Activities, content providers, and intent receivers are all short-lived and can be shut down at any time. Services, on the other hand, are designed to keep running, if needed, independent of any activity.

It is technically possible to create and attach widgets (Window Gadget) to our activity purely through Java code but the more common approach is to use an Extensible Markup Language (XML)-based layout file. XML is a metalanguage, a form of language or set of terms used for the description or analysis of another language, that allows users to define their own customized markup languages, esp. in order to assist display. In this project, dynamic instantiation of widgets is reserved for more complicated scenarios, where the widgets are not known at compile-time. As the name suggests, an XML-based layout is a specification of widgets’ relationships to each other encoded in XML format. Each XML file contains a tree of elements specifying a layout of widgets and their containers that make up one view hierarchy. The attributes of the XML elements are properties, describing how a widget should look or how a container should behave.

## 3.3 Analysis

The analysis phase answers the questions of who will use the system, what the system will do, and where and when it will be used. During this phase, the concept for the new system is finalised. After requirements gathering, the following specification, assumptions and requirements were derived.

### 3.3.1 System Specification

The System (Application) is expected to receive one of the following as input:

* An already scratched card on which Optical Character Recognition will occur
* Text entered by the user
* Button clicks by the user

The system will execute two important processes:

* Optical Character Recognition of the already scratched card
* Recharging of the phone account using the PIN from the card

The output of the OCR process is the typeset PIN in a text field. The outpu t of the recharge process is an indication of the success or failure of the operation.

### 3.3.2 Assumptions on which the system is built

1. The user must understand English.
2. The card must have been scratched without marring the PIN characters.
3. The phone must possess a camera with an optional flash light

### 3.3.3 Requirements:

#### Operational Requirements:

* The Application will operate on the Android platform.
* A minimum Android version of 2.3.3 is required to run this application
* A camera is required to perform OCR

#### Performance Requirements:

* A response time of at most 10 seconds is expected and predicted for both OCR and phone account recharge processes.

#### Security Requirements:

* No special security requirements are anticipated.

#### Cultural and Political Requirements:

* No cultural and political requirements.

## 3.4 System Design

System designs give a detailed account of the modules which describe the functions, behaviour and structure of the application. It helps in determining the optimum size of program modules, the complexity of data structures and the format of interface definitions. System design is very important in a software development process. The design of this application was done using object–oriented modelling. The user actions, system actions, class diagrams, sequence diagrams are explained in this chapter.

### 3.4.1 Interface Design

The Graphical User Interface enables communication i.e. interaction of the user with the system (application). Android User Interfaces are called layouts and layouts are attached to program classes called activities. This application is expected to have two layouts.

1. The main or landing activity’s layout
2. The Camera activity’s layout.

Figure 3.1 shows a design for the main layout. It consists of four major widgets (Window gadgets): Two buttons, one text area and one choice menu. The first button at the top of the layout will start the phone camera. This causes the camera layout to be invoked. Figure 3.2 shows the camera layout. The launch of the camera can be cancelled by the button on the top right of the layout. The second button can be used to turn the phone’s flash on or off if the phone has a flash. The third button can be used to execute Optical Character Recognition on the part of the camera’s view trapped in the rectangular box in the centre. This returns the viewer to the main layout while the OCR will continue on the background and the result presented to the viewer in the text field for further process execution in this case recharging of his phone balance. To complete this task, the user selects the network of his choice and taps the recharge button which is the last button on the layout.

Camera Button

Output Area

Choose Network

1. MTN
2. GLO
3. AIRTEL

Recharge

Figure 3.1: Main Layout Design

Cancel  
Photo  
Capture

Turn on  
Flash

Take  
 Photo

Figure 3.2: Camera Layout Design

### 3.4.2 Activity Diagram

This user utilises an activity diagram to model the functioning and processes of the system. Figure 3.3 describes the flow of activities that can be accomplished by the user when using the application. On start of the application, the user has two options:

1. The user enters the PIN from the already scratched card directly into the text field or
2. The user opts to use the OCR facility by click of a button.

On selection of the OCR facility, a new user interface appears which gives the user 2 options:

1. The user can choose to cancel the whole operation and go back to the first layout
2. The user can execute the Optical Character Recognition of the preselected segment of the camera view (the card region with the PIN)

On execution of the OCR, the previous interface is restored with the output of the OCR operation in the text field. At this point, the user chooses his network provider and afterwards, he taps the recharge button.

### 3.4.3 Sequence Diagram

This author uses a sequence diagram to model the behaviour and dynamic interaction among objects of the system. The sequence diagram in Figure 3.4 shows a scenario of Optical character Recognition execution and recharging of phone account. It shows the time ordering of the messages being passed in the application. The participants or classes modelled are as follows:

* The user
* The camera
* The application navigator
* The call class.

When the user starts the application, the application navigator comes alive. He then chooses to carry out optical character recognition causing the camera object to be created. After this, he carries out the recharge operation using the detected text and this ends the particular use case.

Use OCR?

Yes

No

Take Picture

Enter Pin manually

Choose Network Provider

Recharge Account

Figure 3.3: Activity Diagram

User

rechage()

performOcr()

onButtonClicked()

application Navigator: MainActivity

call: Call Intent

camera: CameraActivity

onCreate()

Figure 3.4: Sequence Diagram

## 3.5 Evaluation Techniques

In the execution of this project, three evaluation techniques are used.

1. A variety of PINs: Recharge cards of different networks will be used to test the application’s functionality and accuracy in different conditions.
2. Audio feedback of the cards to the user is one of the key functionalities of the application. This serves to allow the user evaluate the card just before utilising it in further computations.
3. Alpha and Beta Testing: This involves the general testing and use of the application both by the developer and those closely involved with the development of the application (Alpha test) and also the use of the application by a few random users just to get their feedback, comments, likes and dislikes of the created application.

# CHAPTER 4 DESIGN AND IMPLEMENTATION

## 4.1 Introduction

This chapter is primarily focused on how the android application was implemented. The implementation of each main component of the system and the integration of these components are discussed. It also highlights the tests carried out on the functionality and performance of the main components of the system and the system as a whole.

The approach utilised by this project in implementing the Systems Development Life Cycle is Rapid Application Development (RAD). There are three RAD categories:

1. Phased development
2. Prototyping
3. Throwaway Prototyping

This author uses a prototyping-based approach where analysis, design and implementation were done concurrently and all three phases performed repeatedly in a cycle until the final system was completed. Using this technique, the basics of analysis and design were performed and work immediately commenced on a prototype which was continuously upgraded till it met the originally desired features as accurately as possible. Figure 4.1 shows the flow of SDLC steps that yield the final application using prototyping.

## 4.2 Software and Hardware Implementation

The Integrated Development Environment (IDE) used in implementing the entire application was the Android Developer Toolkit, a modification of Eclipse. This was used because it is the most stable android development environment available till date and it is supported by Google.

The main components of this application are:

1. Application Navigation Service
2. Camera service
3. Optical Character Recognition service
4. Text To Speech (TTS) service

The application navigation was a purely software implementation while the camera service was a mix of software and hardware. The Optical Character Recognition service is a cloud service and the Text to Speech service was a consumption of the default android TTS service.

### 4.2.1 Application Navigation service

This was an essential component of the system which coordinates the other parts of the system. It also acts as an interface between the user and the other services. It was an Android activity implemented using the Java language applying the object-oriented programming paradigm to make the navigation service robust and allow a high level of code reuse for rapid development. The source code for the application activity is in Appendix I.

Figure 4.2 to 4.5 shows the screen shots of the layout implementation. The designs were followed as strictly as possible as discussed in chapter 3 with the addition of a digit count field which is used to aid the user in knowing the number of digits of the PIN as a quick guide to help know if the OCR operation occurred correctly and the use of an image button instead of a text button as the trigger for camera service. Table 4.1 shows the distribution of Network providers to PIN digit count in Nigeria. It can be seen that this similarity in PIN digit count makes it difficult to automatically make a decision on the network provider of the card. To combat this, a drop down option was provided from which the user chooses the network provider he wishes to load credit on. This has an advantage in that users with phones that support multiple SIM cards can recharge both accounts conveniently and need not bother to memorise the short codes for their respective network providers as the application has already done this pre-processing on the background.

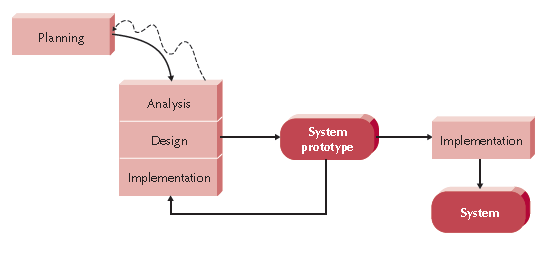


Figure 4.1: Prototyping

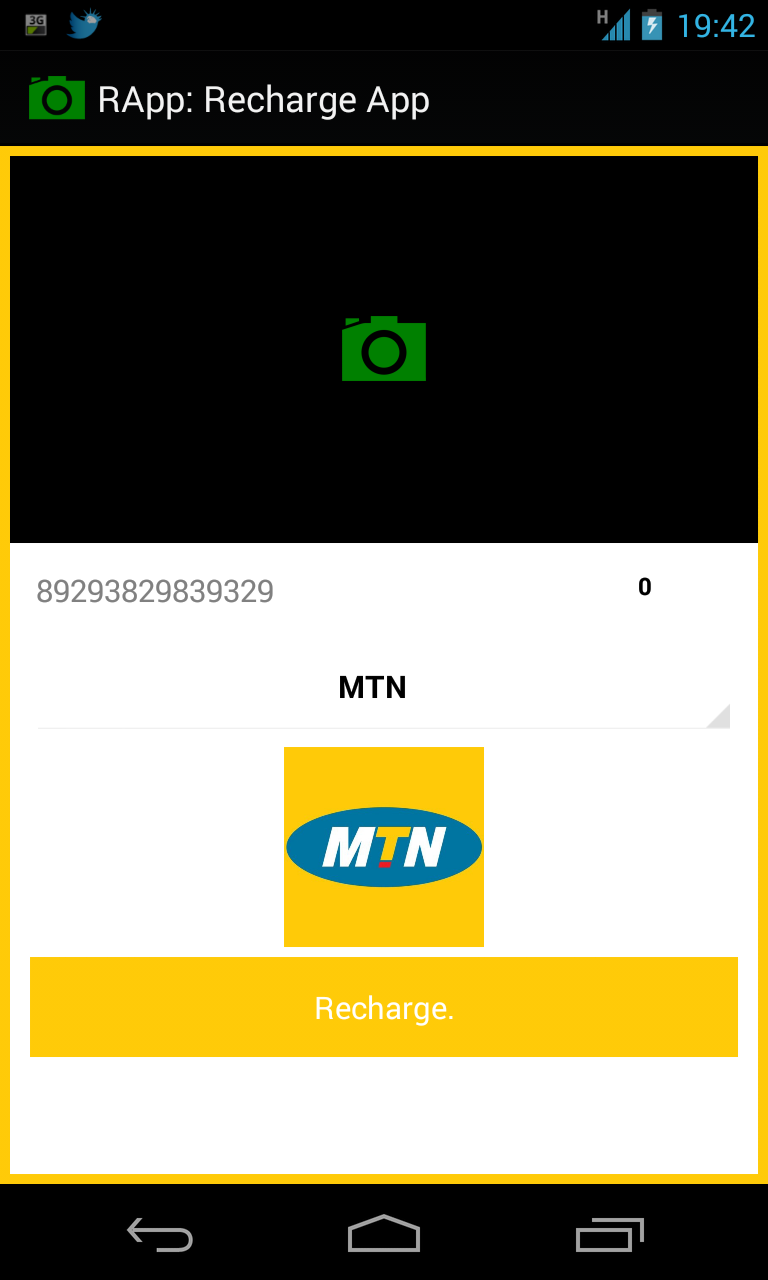


Figure 4.2: Application Navigation (MTN)

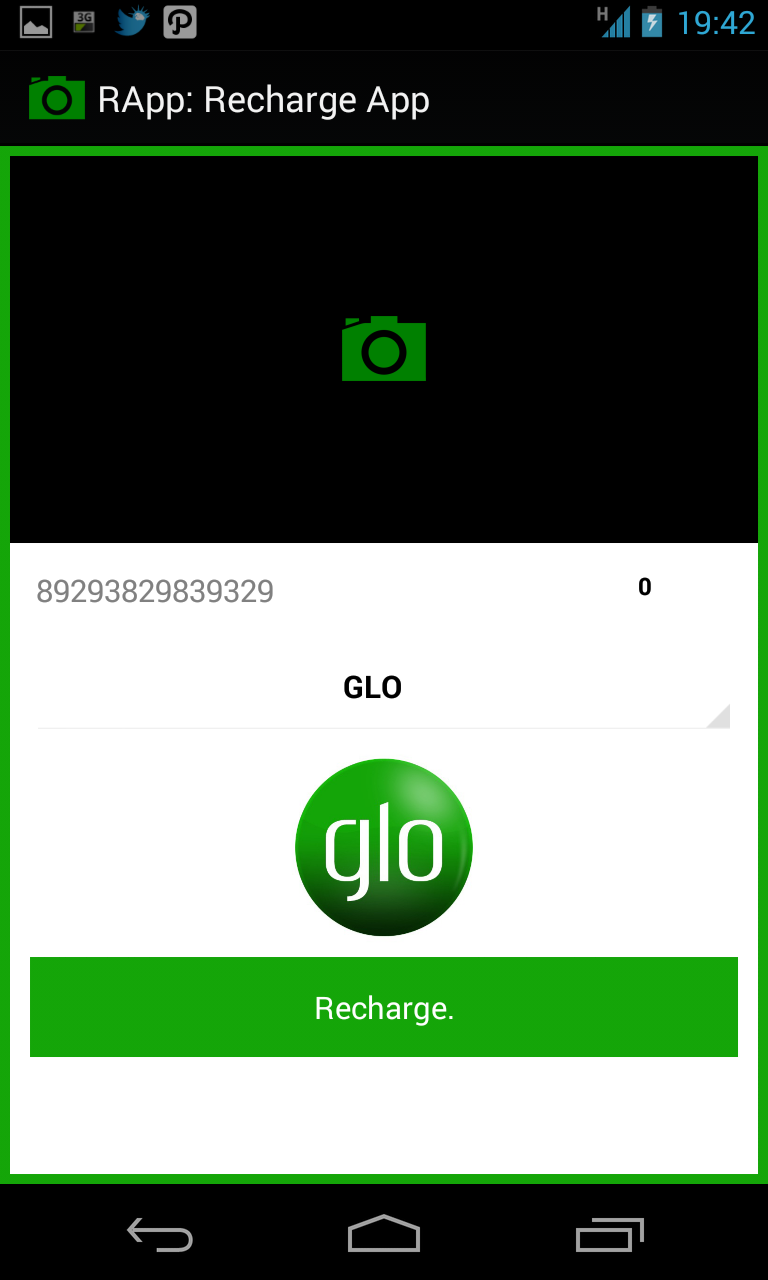


Figure 4.3: Application Navigation (GLO)

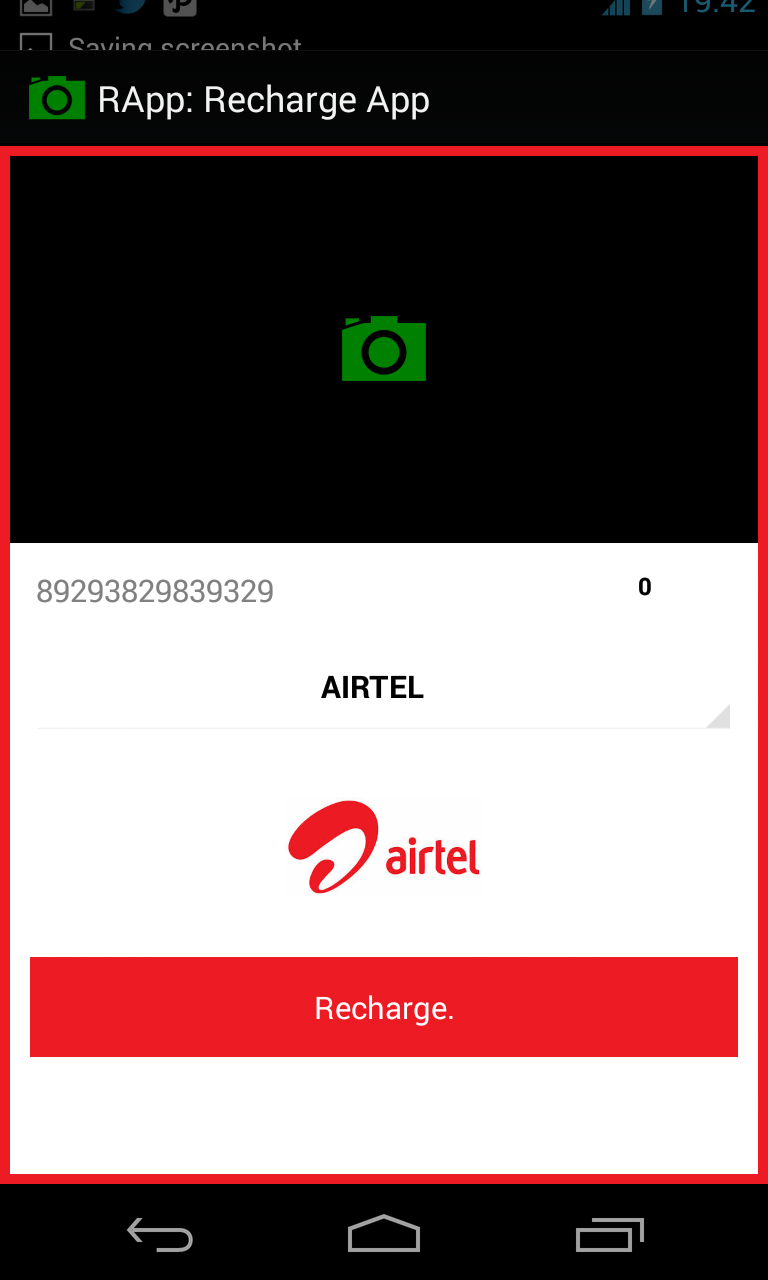


Figure 4.4: Application Navigation (Airtel)

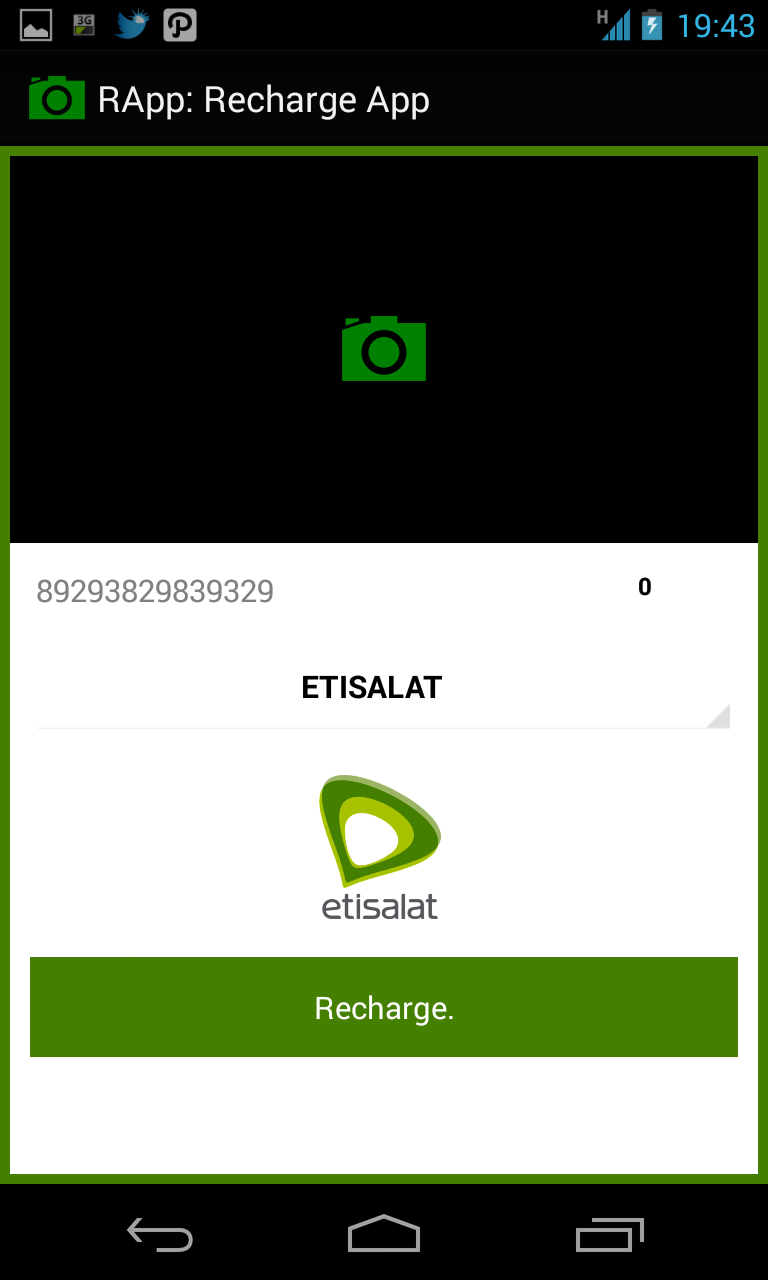


Figure 4.5: Application Navigation (Etisalat)

Table 4.1: PIN digit counts by Network providers in Nigeria

|  |  |
| --- | --- |
| **Network Provider** | **PIN digit count** |
| MTN | 12 |
| GLO | 15 |
| AIRTEL | 16 |
| ETISALAT | 15 |

### 4.2.2 Camera Service

The application utilises the camera of the phone to carry out Optical Character Recognition of the PIN on the recharge card. This service can be described as a collaboration of software and hardware Android commands. The orientation of the camera activity is locked to landscape so as to prevent crashes which occur on change of orientation especially on non-standard Android phones such as Tecno and Infinix phone brands and also because an abstraction of the default camera intent had to be created and implemented so that an adjustable field could be superimposed on it.

An attempt was made to follow the design drafted in Figure 3.2 as closely as possible but instead of normal text buttons, image buttons were used. An adjustable field is implemented to assist the user in taking pictures and also to accommodate for and avoid screen deformation of differently sized devices. This adjustable field serves as a means of pre-processing the image to remove the noise of other characters on the card from it.  
The Code for this activity is displayed in Appendix II

### 4.2.3 Cloud Service

This was an essential component of the system which provides the core functionality of the system, the Optical Character Recognition. A cloud service was chosen because of the limitations to Optical Character Recognition that the Android hardware provided. The results that were gotten with offline Optical Character Recognition returned an average of 50% accuracy on the most advanced devices tested on and the detection of PINs is an operation that requires maximum accuracy. The model of the web service was service oriented because it acts as a utility system providing information and performing operations based on the requests of the other components of the system.

The cloud service communicated with the application by exchanging data in XML format. It was required of the application to parse the XML received and extract the result elements from the XML structure. This was a simpler and more accurate task for an android phone to execute than if the phone was required to execute the Optical Character recognition on its own.

This service was implemented as an asynchronous service on a different thread but within the same activity in the android program. The code for this activity is highlighted in Appendix I.

### 4.2.4 Text To Speech Service (TTS)

Android has the ability to execute TTS actions by creating an instance of the TTS default class. This TTS class comes preinstalled as a service that can be called upon by most android phones. To cater for phones that do not have this feature preinstalled, this project provides for the download of the TTS package by the user from the android application market, Google Playstore.

This service is implemented in the application navigation activity and so the code that implements this service is highlighted in Appendix I also.

## 4.3 Evaluation

This author utilised integration, alpha and beta testing.

### 4.3.1 Integration testing

The integration testing required each component of the system, which was implemented as individual applications firstly, to be tested to ensure that they functioned properly. This was to confirm that the implementation was done satisfactorily and that all components communicated perfectly with each other in the final application. The different components were tested based on their nature and structure.

After integration, communication errors were initially rampant all through the system but this was debugged using Logcat and Android development tools and techniques.

### 4.3.2 Alpha testing

Alpha test refers to the trial of the software produced by this project, carried out by the developer (this author) before the product was made available for beta testing.

The web service was tested by running several case scenarios on them. It was done to uncover any bugs or loop hole in the service and to ascertain the accuracy which turned out to be appropriate for the PIN detection functionality. After more testing of the web service, a technique was devised to fine-tune the web service for PIN detection from words of length less than 15 (the maximum length of the recharge card PIN in Nigeria).

### 4.3.3 Beta Testing

Beta testing refers to the trial of the software produced by this project during the final stages of its development (implementation), carried out by people unconnected with its development. This application was given to friends to test run on their android devices, was hosted on a cloud market but not publicised so that the application could be downloaded by distant people. This beta test brought to light two important flaws. It was discovered that the application will not start up the camera when the battery of the device was below a certain minimum point. It was also discovered that the flash light of certain devices (Tecno) was not manoeuvrable by the application. The later bug was fixed by changing a parameter in the code that controlled the flash light while the former bug was not fixed as it is the default behaviour of the device and no hack was found around it.

# CHAPTER 5 CONCLUSION AND RECOMMENDATION

## 5.1 Introduction

This chapter concludes this project’s write-up. It discusses the project’s summary, contribution to knowledge, the challenges encountered during the course of implementing the project and the recommendations on enhancing the project for further study.

## 5.2 Summary

The background of the project as well as its aims and objectives were discussed in chapter one. Chapter two consist of the literature review which highlighted the necessary theoretically knowledge and related projects which form the foundation of the project. The methodology was discussed in chapter three which explains how the system would be built by identifying key components of the project and their functionalities. It also highlights the requirements to implement the project. The design and implementation of the project was discussed in detail in chapter four.

## 5.3 Contribution to Knowledge

The project developed has introduced another way in which Personal Identification Numbers (PINs) on recharge cards can be loaded to the owners account. This enables people to use recharge cards without typing in text using android keypads which can be very clumsy. The project shows that it is possible to utilise Optical Character Recognition (OCR) to obtain target text that exists among other text, in this case as PIN on cards, using a mobile phone.

## 5.4 Challenges

There were some challenges faced during the implementation the project. The challenges are follows:

1. Accuracy of the OCR engine when deployed locally on the android device was very poor even after continuous training of the engine. This led to the use of a cloud service instead.
2. The camera activity returned a run time error on some devices when the orientation was changed. This was prevented by fixing the orientation of the camera to landscape mode.

## 5.5 Recommendation

This project attempted to get an offline Optical Character Recognition engine but had to settle for an online engine instead. With more research, increment in knowledge and skills and more time, this author believes an accurate offline engine is still feasible and is recommended for use by this application.

## 5.6 Conclusion

In this work, a Personal Identification Number detector using Optical character recognition has been created. We have also shown that irrespective of the size of the average phone, the use of cloud computing can bolster the capacity of a smart phone to execute desktop like functions hence proving that they are “truly smart”.

# REFERENCES

Adewumi, Onaopemipo .O. (2012), Development of a Pattern Recognition System for Faces with Tribal Marks. Ile-Ife: Department of Computer Science and Engineering, Obafemi Awolowo University.

Brimblecombe, Phil (2003). “Face detection using neural network.” Meng Electronic EngineeringSchool of Electronics and Physical Sciences: 615.

Gonzalez, R C. and Thomason, M G (1978). Syntactic Pattern Recognition : An Introduction. Reading, MA: Addison Wesley.

IBM Corporation. z/OS V1R12.0 Cryptographic Services ICSF Application Programmer's Guide. 1 January 2010. http://publib.boulder.ibm.com/infocenter/zos/v1r12/topic/com.ibm.zos.r12.csfb400/csfb4za0268.htm (accessed May 27, 2013).

ISO. ISO standard makes sure that PINs are secure. 3 May 2011. http://www.iso.org/iso/home/news\_index/news\_archive/news.htm?refid=Ref1422 (accessed May 27, 2013).

Line, Eikvil (1993). Optical Character Recognition. Blindern: Norsk Regnesentral.

Polikar, Robi (2006). “Pattern Recognition.” Wiley Encyclopedia of Biomedical Engineering (Wiley Encyclopedia of Biomedical Engineering, Copyright & 2006 John Wiley & Sons, Inc.): 3-8.

# APPENDIX I MAIN ACTIVITY

package com.kaylee.rapp;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.Reader;

import java.util.HashMap;

import java.util.Locale;

import org.xmlpull.v1.XmlPullParser;

import org.xmlpull.v1.XmlPullParserException;

import com.abbyy.ocrsdk.Client;

import com.abbyy.ocrsdk.Task;

import com.abbyy.ocrsdk.TextFieldSettings;

import com.actionbarsherlock.app.ActionBar;

import com.actionbarsherlock.app.SherlockActivity;

import com.bugsense.trace.BugSenseHandler;

import android.app.Activity;

import android.content.ActivityNotFoundException;

import android.content.Intent;

import android.content.SharedPreferences;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.net.Uri;

import android.os.AsyncTask;

import android.os.Bundle;

import android.os.Environment;

import android.speech.tts.TextToSpeech;

import android.text.Editable;

import android.text.TextWatcher;

import android.util.Log;

import android.util.Xml;

//import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemSelectedListener;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.ImageView.ScaleType;

import android.widget.LinearLayout;

import android.widget.Spinner;

import android.widget.TextView;

import android.widget.Toast;

public class MainActivity extends SherlockActivity implements TextToSpeech.OnInitListener {

ActionBar actbar;

EditText number;

Spinner networks;

Button recharge;

ArrayAdapter<String> adapter;

LinearLayout background;

TextView numbcount;

ImageView networklogo;

ImageView pic;

//String outputPath;

File file = new File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY\_PICTURES),"test.jpg");

File ofile = new File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY\_DOWNLOADS),"result.txt");

boolean ocr, ts;

//String imageUrl = "unknown";

private TextToSpeech tts;

private String currentnetwork;

private static final String[] operators = {"MTN", "GLO", "AIRTEL", "ETISALAT"};

private static final HashMap<String, Object> values, networklogos;

static{

values = new HashMap<String, Object>();

values.put("MTN", R.drawable.mtn\_selector);

values.put("GLO", R.drawable.glo\_selector);

values.put("AIRTEL", R.drawable.airtel\_selector);

values.put("ETISALAT", R.drawable.etisalat\_selector);

values.put("MTNCODE", "\*555\*XXX"+Uri.encode("#"));

values.put("GLOCODE", "\*123\*XXX"+Uri.encode("#"));

values.put("AIRTELCODE", "\*126\*XXX"+Uri.encode("#"));

values.put("ETISALATCODE", "\*222\*XXX"+Uri.encode("#"));

networklogos = new HashMap<String, Object>();

networklogos.put("MTN", R.drawable.mtnlogo);

networklogos.put("GLO", R.drawable.glologo);

networklogos.put("AIRTEL", R.drawable.airtellogo);

networklogos.put("ETISALAT", R.drawable.etilogo);

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

//BugSenseHandler.initAndStartSession(MainActivity.this, "bb1beec2");

setContentView(R.layout.activity\_main);

Bundle extras = getIntent().getExtras();

if( extras != null) {

ocr = extras.getBoolean("OCR");

//imageUrl = extras.getString("IMAGE\_PATH" );

//outputPath = extras.getString( "RESULT\_PATH" );

}

if (ocr){

new AsyncProcessTask(this).execute(file, ofile);

tts = new TextToSpeech(this, this);

}

//new AsyncProcessTask().execute(imageUrl);

// Starting recognition process

actbar = getSupportActionBar();

actbar.show();

background = (LinearLayout)this.findViewById(R.id.back);

number = (EditText)this.findViewById(R.id.number);

networks = (Spinner)this.findViewById(R.id.networks);

recharge = (Button)this.findViewById(R.id.recharge);

numbcount = (TextView)this.findViewById(R.id.numcount);

networklogo = (ImageView)this.findViewById(R.id.networklogo);

pic = (ImageView)this.findViewById(R.id.preview);

if(file.exists() && ocr){

Bitmap myBitmap = BitmapFactory.decodeFile(file.getAbsolutePath());

pic.setScaleType(ScaleType.FIT\_XY);

pic.setImageBitmap(myBitmap);

}

adapter = new ArrayAdapter<String>(this, R.layout.operatoritem, operators);

adapter.notifyDataSetChanged();

networks.setAdapter(adapter);

final TextWatcher watch = new TextWatcher(){

@Override

public void afterTextChanged(Editable arg0) {

// TODO Auto-generated method stub

//numbcount.setText(arg0.toString().length());

}

@Override

public void beforeTextChanged(CharSequence arg0, int arg1,

int arg2, int arg3) {

// TODO Auto-generated method stub

//numbcount.setText(arg0.toString().length());

}

@Override

public void onTextChanged(CharSequence arg0, int arg1, int arg2,

int arg3) {

// TODO Auto-generated method stub

//numbcount.setText(arg0.toString().length());

numbcount.setText(String.valueOf(arg0.length()));

try{

if (currentnetwork.equals("MTN") && number.getText().toString().length() == 12){

numbcount.setTextColor(getResources().getColor(R.color.mtnyellow));

}else if (currentnetwork.equals("AIRTEL") && number.getText().toString().length() == 16){

numbcount.setTextColor(getResources().getColor(R.color.airtelred));

}else if (currentnetwork.equals("GLO") && number.getText().toString().length() == 15){

numbcount.setTextColor(getResources().getColor(R.color.glogreen));

}else if (currentnetwork.equals("ETISALAT") && number.getText().toString().length() == 15){

numbcount.setTextColor(getResources().getColor(R.color.etigreen));

}else{

numbcount.setTextColor(getResources().getColor(R.color.black));

}

}

catch (Exception e){

}

}

};

number.addTextChangedListener(watch);

networks.setOnItemSelectedListener(new OnItemSelectedListener(){

@Override

public void onItemSelected(AdapterView<?> arg0, View arg1,

int arg2, long arg3) {

// TODO Auto-generated method stub

currentnetwork = arg0.getItemAtPosition(arg2).toString();

background.setBackgroundResource((Integer) switchBack(currentnetwork));

recharge.setBackgroundResource((Integer)switchBack(currentnetwork));

networklogo.setImageResource(switchLogo(currentnetwork));

//Toast.makeText(getApplicationContext(), currentnetwork, Toast.LENGTH\_SHORT).show();

}

@Override

public void onNothingSelected(AdapterView<?> arg0) {

// TODO Auto-generated method stub

currentnetwork = "MTN";

}

});

pic.setOnClickListener(

new View.OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

Intent camera = new Intent(getApplicationContext(), CameraPreview.class);

startActivity(camera);

}

});

recharge.setOnClickListener(

new OnClickListener(){

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

//Toast.makeText(getApplicationContext(), number(currentnetwork, number.getText().toString()), Toast.LENGTH\_SHORT).show();

if (currentnetwork.equals("MTN") && number.getText().toString().length() == 12){

call(currentnetwork , number.getText().toString());

}else if (currentnetwork.equals("AIRTEL") && number.getText().toString().length() == 16){

call(currentnetwork , number.getText().toString());

}else if (currentnetwork.equals("GLO") && number.getText().toString().length() == 15){

call(currentnetwork , number.getText().toString());

}else if (currentnetwork.equals("ETISALAT") && number.getText().toString().length() == 15){

call(currentnetwork , number.getText().toString());

}else if (number.getText().toString().length() < 16 || number.getText().toString().length() < 15 || number.getText().toString().length() < 12){

Toast.makeText(getApplicationContext(), currentnetwork+ " recharge pin digits is incomplete", Toast.LENGTH\_LONG).show();

}else if (number.getText().toString().length() > 16 || number.getText().toString().length() > 15 || number.getText().toString().length() > 12){

Toast.makeText(getApplicationContext(), currentnetwork+ " recharge pin digits is in excess", Toast.LENGTH\_LONG).show();

}

}

}

);

}

private Object switchBack(String network){

return values.get(network);

}

private String number(String key ,String digits){

return values.get(key+"CODE").toString().replace("XXX", digits);

}

private int switchLogo(String network){

return (Integer)networklogos.get(network);

}

private void call(String network, String digits) {

try {

Intent callIntent = new Intent(Intent.ACTION\_CALL);

callIntent.setData(Uri.parse("tel:"+number(network, digits)));

startActivity(callIntent);

} catch (ActivityNotFoundException activityException) {

Log.e("helloandroid dialing example", "Call failed");

}

}

public void updateResults() {

try {

//Toast.makeText(this, "Everything finished", Toast.LENGTH\_SHORT).show();

StringBuffer contents = new StringBuffer();

FileInputStream fis = new FileInputStream(ofile);

Reader reader = new InputStreamReader(fis, "UTF-8");

//BufferedReader bufReader = new BufferedReader(reader);

/\*String text = null;

while ((text = bufReader.readLine()) != null) {

contents.append(text).append(System.getProperty("line.separator"));

}

\*/

//bufReader.close();

XmlPullParser parser = Xml.newPullParser();

parser.setInput(reader);

parser.nextTag();

contents = readFeed(parser, contents);

displayMessage(contents.toString());

if (ts){

speakOut(contents.toString());

}

} catch (Exception e) {

displayMessage("Error: " + e.getMessage());

}

}

private StringBuffer readFeed(XmlPullParser parser, StringBuffer value) throws XmlPullParserException, IOException {

//parser.require(XmlPullParser.START\_TAG, ns, "feed");

while (parser.next() != XmlPullParser.END\_TAG) {

if (parser.getEventType() != XmlPullParser.START\_TAG) {

continue;

}

String name = parser.getName();

// Starts by looking for the entry tag

if (name.equals("value") && parser.next() == XmlPullParser.TEXT) {

value.append(parser.getText()).append(System.getProperty("line.separator"));

}

}

return value;

}

private void displayMessage( String text ){

char[] chars = text.toCharArray();

StringBuffer outputChars = new StringBuffer();

for(char c : chars){

if(Character.isDigit(c))

outputChars.append(c);

else{

outputChars.append("\*");

}

}

Log.i("message", text);

text = outputChars.toString();

number.post( new MessagePoster( text ) );

Log.i("message", text);

}

/\*@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.activity\_results, menu);

return true;

}\*/

class MessagePoster implements Runnable {

public MessagePoster( String message )

{

\_message = message;

}

public void run() {

number.setText( \_message + "\n" );

}

private final String \_message;

}

class AsyncProcessTask extends AsyncTask<File, String, Boolean> {

public AsyncProcessTask(MainActivity activity) {

this.activity = activity;

}

//private ProgressDialog dialog;

/\*\* application context. \*/

private MainActivity activity;

protected void onPreExecute() {

//Toast.makeText(activity, "Processing", Toast.LENGTH\_SHORT).show();

// dialog = new ProgressDialog(activity);

// dialog.setMessage("Processing");

// dialog.setCancelable(false);

// dialog.setCanceledOnTouchOutside(false);

// dialog.show();

}

protected void onPostExecute(Boolean result) {

activity.updateResults();

// if (dialog.isShowing()) {

// dialog.dismiss();

// }

// activity.updateResults();

}

@Override

protected Boolean doInBackground(File... args) {

File inputFile = args[0];

File outputFile = args[1];

try {

Client restClient = new Client();

//!!! Please provide credentials and remove this line. !!!

// Name of application you created

restClient.applicationId = "OLUWAApp";

// You should get e-mail from ABBYY Cloud OCR SDK service with the application password

restClient.password = "JPLU8RmuKcTrrHFeas1AAf5+";

// Obtain installation id when running the application for the first time

SharedPreferences settings = activity.getPreferences(Activity.MODE\_PRIVATE);

String instIdName = "installationId";

if( !settings.contains(instIdName)) {

// Get installation id from server using device id

String deviceId = android.provider.Settings.Secure.getString(activity.getContentResolver(),

android.provider.Settings.Secure.ANDROID\_ID);

// Obtain installation id from server

//publishProgress( "First run of App, initialising things");

String installationId = restClient.activateNewInstallation(deviceId);

//publishProgress( "Done with initialisation");

SharedPreferences.Editor editor = settings.edit();

editor.putString(instIdName, installationId);

editor.commit();

}

String installationId = settings.getString(instIdName, "");

restClient.applicationId += installationId;

//publishProgress( "Uploading image...");

// String language = "English"; // Comma-separated list: Japanese,English or German,French,Spanish etc.

// ProcessingSettings processingSettings = new ProcessingSettings();

// processingSettings.setOutputFormat( ProcessingSettings.OutputFormat.txt );

// processingSettings.setLanguage(language);

TextFieldSettings tfs = new TextFieldSettings();

//publishProgress("Uploading..");

// If you want to process business cards, uncomment this

/\*

BusCardSettings busCardSettings = new BusCardSettings();

busCardSettings.setLanguage(language);

busCardSettings.setOutputFormat(BusCardSettings.OutputFormat.xml);

Task task = restClient.processBusinessCard(filePath, busCardSettings);

\*/

//Task task = restClient.processImage(inputFile.getPath(), processingSettings);

Task task = restClient.processTextField(inputFile.getPath(), tfs);

while( task.isTaskActive() ) {

Thread.sleep(2000);

publishProgress( "Waiting.." );

task = restClient.getTaskStatus(task.Id);

}

if( task.Status == Task.TaskStatus.Completed ) {

//publishProgress( "Downloading.." );

FileOutputStream fos = new FileOutputStream(outputFile);

//FileOutputStream fos = activity.openFileOutput(outputFile,Context.MODE\_PRIVATE);

restClient.downloadResult(task, fos);

//Log.i("kay lee", "It saved");

fos.close();

publishProgress( "Ready" );

} else if( task.Status == Task.TaskStatus.NotEnoughCredits ) {

//publishProgress( "Not enough credits to process task. Add more pages to your application's account." );

} else {

publishProgress( "Task failed" );

}

return true;

} catch (Exception e) {

publishProgress( "Error connecting to the internet");

return false;

}

}

@Override

protected void onProgressUpdate(String... values) {

// TODO Auto-generated method stub

String stage = values[0];

Log.i("toast", stage);

Toast.makeText(activity, stage, Toast.LENGTH\_SHORT).show();

// dialog.setProgress(values[0]);

}

}

public void onInit(int status){

if (status == TextToSpeech.SUCCESS){

int result = tts.setLanguage(Locale.US);

if (result == TextToSpeech.LANG\_MISSING\_DATA || result == TextToSpeech.LANG\_NOT\_SUPPORTED){

Log.e("TTS", "This language is not supported");

Intent installIntent = new Intent();

installIntent.setAction(TextToSpeech.Engine.ACTION\_INSTALL\_TTS\_DATA);

startActivity(installIntent);

}

else{

ts = true;

//btnSpeak.setEnabled(true);

//speakOut();

}

}

else

Log.e("TTS", "Initialization Failed!");

}

private void speakOut(String text){

//String text = number.getText().toString();

tts.speak(text, TextToSpeech.QUEUE\_FLUSH, null);

Log.i("TTS", text);

}

@Override

protected void onDestroy() {

// TODO Auto-generated method stub

if(file.exists() || ofile.exists()){

file.delete();

ofile.delete();

}

if (tts != null){

tts.stop();

tts.shutdown();

}

super.onDestroy();

}

}

# APPENDIX II CAMERA ACTIVITY

package com.kaylee.rapp;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import com.bugsense.trace.BugSenseHandler;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import android.graphics.Bitmap;

import android.graphics.Rect;

import android.graphics.Bitmap.CompressFormat;

import android.graphics.drawable.Drawable;

import android.hardware.Camera;

import android.hardware.Sensor;

import android.hardware.SensorEvent;

import android.hardware.SensorEventListener;

import android.hardware.SensorManager;

import android.hardware.Camera.AutoFocusCallback;

import android.hardware.Camera.Size;

import android.os.Bundle;

import android.os.Environment;

import android.util.DisplayMetrics;

import android.util.Log;

import android.view.KeyEvent;

import android.view.MotionEvent;

import android.view.View;

import android.view.Window;

import android.view.WindowManager;

import android.view.View.OnClickListener;

import android.widget.ImageView;

import android.widget.Toast;

import android.widget.RelativeLayout.LayoutParams;

public class CameraPreview extends Activity implements SensorEventListener {

private Preview mPreview;

private ImageView mTakePicture;

private ImageView mFlash;

private ImageView mSkip;

private TouchView mView;

private boolean mAutoFocus = true;

private boolean mFlashBoolean = false;

private SensorManager mSensorManager;

private Sensor mAccel;

private boolean mInitialized = false;

private float mLastX = 0;

private float mLastY = 0;

private float mLastZ = 0;

private Rect rec = new Rect();

private int mScreenHeight;

private int mScreenWidth;

private boolean mInvalidate = false;

boolean possible = false;

// private File mLocation = new File(Environment.

// getExternalStorageDirectory(),"test.jpg");

private File path = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY\_PICTURES);

private File mLocation = new File(path,"test.jpg");

//private File result = new File(path, "result.txt");

//private String resultUrl = "result.txt";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

//Log.i(TAG, "onCreate()");

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN,

WindowManager.LayoutParams.FLAG\_FULLSCREEN);

// display our (only) XML layout - Views already ordered

//BugSenseHandler.initAndStartSession(CameraPreview.this, "bb1beec2");

setContentView(R.layout.main);

// the accelerometer is used for autofocus

mSensorManager = (SensorManager) getSystemService(Context.

SENSOR\_SERVICE);

mAccel = mSensorManager.getDefaultSensor(Sensor.

TYPE\_ACCELEROMETER);

// get the window width and height to display buttons

// according to device screen size

DisplayMetrics displaymetrics = new DisplayMetrics();

getWindowManager().getDefaultDisplay().getMetrics(displaymetrics);

mScreenHeight = displaymetrics.heightPixels;

mScreenWidth = displaymetrics.widthPixels;

// I need to get the dimensions of this drawable to set margins

// for the ImageView that is used to take pictures

Drawable mButtonDrawable = this.getResources().

getDrawable(R.drawable.camera);

mTakePicture = (ImageView) findViewById(R.id.startcamerapreview);

mSkip = (ImageView) findViewById(R.id.nocamerapreview);

mFlash = (ImageView) findViewById(R.id.flash);

// setting where I will draw the ImageView for taking pictures

LayoutParams ln = new LayoutParams(mTakePicture.getLayoutParams());

ln.setMargins((int)((double)mScreenWidth\*.85),

(int)((double)mScreenHeight\*.30) ,

(int)((double)mScreenWidth\*.85)+mButtonDrawable.

getMinimumWidth(),

(int)((double)mScreenHeight\*.30)+mButtonDrawable.

getMinimumHeight());

mSkip.setLayoutParams(ln);

LayoutParams lf = new LayoutParams(mTakePicture.getLayoutParams());

lf.setMargins((int)((double)mScreenWidth\*.85),

(int)((double)mScreenHeight\*.50) ,

(int)((double)mScreenWidth\*.85)+mButtonDrawable.

getMinimumWidth(),

(int)((double)mScreenHeight\*.50)+mButtonDrawable.

getMinimumHeight());

mFlash.setLayoutParams(lf);

LayoutParams lp = new LayoutParams(mTakePicture.getLayoutParams());

lp.setMargins((int)((double)mScreenWidth\*.85),

(int)((double)mScreenHeight\*.70) ,

(int)((double)mScreenWidth\*.85)+mButtonDrawable.

getMinimumWidth(),

(int)((double)mScreenHeight\*.70)+mButtonDrawable.

getMinimumHeight());

mTakePicture.setLayoutParams(lp);

// rec is used for onInterceptTouchEvent. I pass this from the

// highest to lowest layer so that when this area of the screen

// is pressed, it ignores the TouchView events and passes it to

// this activity so that the button can be pressed.

rec.set((int)((double)mScreenWidth\*.85),

(int)((double)mScreenHeight\*.10) ,

(int)((double)mScreenWidth\*.85)+mButtonDrawable.getMinimumWidth(),

(int)((double)mScreenHeight\*.70)+mButtonDrawable.getMinimumHeight());

mButtonDrawable = null;

mTakePicture.setOnClickListener(previewListener);

mSkip.setOnClickListener(skipListener);

mFlash.setOnClickListener(flashListener);

// get our Views from the XML layout

mPreview = (Preview) findViewById(R.id.preview);

mView = (TouchView) findViewById(R.id.left\_top\_view);

mView.setRec(rec);

}

// this is the autofocus call back

private AutoFocusCallback myAutoFocusCallback = new AutoFocusCallback(){

public void onAutoFocus(boolean autoFocusSuccess, Camera arg1) {

//Wait.oneSec();

mAutoFocus = true;

}};

// with this I get the ratio between screen size and pixels

// of the image so I can capture only the rectangular area of the

// image and save it.

public Double[] getRatio(){

Size s = mPreview.getCameraParameters().getPreviewSize();

double heightRatio = (double)s.height/(double)mScreenHeight;

double widthRatio = (double)s.width/(double)mScreenWidth;

Double[] ratio = {heightRatio,widthRatio};

return ratio;

}

// I am not using this in this example, but its there if you want

// to turn on and off the flash.

private OnClickListener flashListener = new OnClickListener(){

@Override

public void onClick(View v) {

if (mFlashBoolean){

mPreview.setFlash(false);

Toast.makeText(getApplicationContext(), "Flash off", Toast.LENGTH\_SHORT).show();

}

else{

mPreview.setFlash(true);

Toast.makeText(getApplicationContext(), "Flash on", Toast.LENGTH\_SHORT).show();

}

mFlashBoolean = !mFlashBoolean;

}

};

// This method takes the preview image, grabs the rectangular

// part of the image selected by the bounding box and saves it.

// A thread is needed to save the picture so not to hold the UI thread.

private OnClickListener previewListener = new OnClickListener() {

@Override

public void onClick(View v) {

showToast();

if (mAutoFocus){

mAutoFocus = false;

path.mkdirs();

mPreview.setCameraFocus(myAutoFocusCallback);

//Wait.oneSec();

Thread tGetPic = new Thread( new Runnable() {

public void run() {

Double[] ratio = getRatio();

int left = (int) (ratio[1]\*(double)mView.getmLeftTopPosX());

// 0 is height

int top = (int) (ratio[0]\*(double)mView.getmLeftTopPosY());

int right = (int)(ratio[1]\*(double)mView.getmRightBottomPosX());

int bottom = (int)(ratio[0]\*(double)mView.getmRightBottomPosY());

savePhoto(mPreview.getPic(left,top,right,bottom));

mAutoFocus = true;

}

});

tGetPic.start();

startResult();

}

boolean pressed = false;

if (!mTakePicture.isPressed()){

pressed = true;

}

}

};

// just to close the app and release resources.

@Override

public boolean onKeyDown(int keyCode, KeyEvent event) {

if (keyCode == KeyEvent.KEYCODE\_BACK){

finish();

}

return super.onKeyDown(keyCode, event);

}

private boolean savePhoto(Bitmap bm) {

FileOutputStream image = null;

try {

checkExternalMedia();

image = new FileOutputStream(mLocation);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

bm.compress(CompressFormat.JPEG, 100, image);

if (bm != null) {

int h = bm.getHeight();

int w = bm.getWidth();

//Log.i(TAG, "savePhoto(): Bitmap WxH is " + w + "x" + h);

} else {

//Log.i(TAG, "savePhoto(): Bitmap is null..");

return false;

}

return true;

}

public boolean onInterceptTouchEvent(MotionEvent ev) {

final int action = ev.getAction();

boolean intercept = false;

switch (action) {

case MotionEvent.ACTION\_UP:

break;

case MotionEvent.ACTION\_DOWN:

float x = ev.getX();

float y = ev.getY();

// here we intercept the button press and give it to this

// activity so the button press can happen and we can take

// a picture.

if ((x >= rec.left) && (x <= rec.right) && (y>=rec.top) && (y<=rec.bottom)){

intercept = true;

}

break;

}

return intercept;

}

// mainly used for autofocus to happen when the user takes a picture

// I also use it to redraw the canvas using the invalidate() method

// when I need to redraw things.

public void onSensorChanged(SensorEvent event) {

if (mInvalidate == true){

mView.invalidate();

mInvalidate = false;

}

float x = event.values[0];

float y = event.values[1];

float z = event.values[2];

if (!mInitialized){

mLastX = x;

mLastY = y;

mLastZ = z;

mInitialized = true;

}

float deltaX = Math.abs(mLastX - x);

float deltaY = Math.abs(mLastY - y);

float deltaZ = Math.abs(mLastZ - z);

if (deltaX > .5 && mAutoFocus){ //AUTOFOCUS (while it is not autofocusing)

mAutoFocus = false;

mPreview.setCameraFocus(myAutoFocusCallback);

}

if (deltaY > .5 && mAutoFocus){ //AUTOFOCUS (while it is not autofocusing)

mAutoFocus = false;

mPreview.setCameraFocus(myAutoFocusCallback);

}

if (deltaZ > .5 && mAutoFocus){ //AUTOFOCUS (while it is not autofocusing) \*/

mAutoFocus = false;

mPreview.setCameraFocus(myAutoFocusCallback);

}

mLastX = x;

mLastY = y;

mLastZ = z;

}

// extra overrides to better understand app lifecycle and assist debugging

@Override

protected void onDestroy() {

super.onDestroy();

//Log.i(TAG, "onDestroy()");

}

@Override

protected void onPause() {

super.onPause();

//Log.i(TAG, "onPause()");

mSensorManager.unregisterListener(this);

}

@Override

protected void onResume() {

super.onResume();

mSensorManager.registerListener(this, mAccel, SensorManager.SENSOR\_DELAY\_UI);

//Log.i(TAG, "onResume()");

}

@Override

protected void onRestart() {

super.onRestart();

//Log.i(TAG, "onRestart()");

}

@Override

protected void onStop() {

super.onStop();

//Log.i(TAG, "onStop()");

}

@Override

protected void onStart() {

super.onStart();

//Log.i(TAG, "onStart()");

}

public void onAccuracyChanged(Sensor sensor, int accuracy) {

// TODO Auto-generated method stub

}

private boolean checkExternalMedia(){

boolean mExternalStorageAvailable = false;

boolean mExternalStorageWriteable = false;

String state = Environment.getExternalStorageState();

if (Environment.MEDIA\_MOUNTED.equals(state)) {

// Can read and write the media

mExternalStorageAvailable = mExternalStorageWriteable = true;

} else if (Environment.MEDIA\_MOUNTED\_READ\_ONLY.equals(state)) {

// Can only read the media

mExternalStorageAvailable = true;

mExternalStorageWriteable = false;

} else {

// Can't read or write

mExternalStorageAvailable = mExternalStorageWriteable = false;

}

//Log.i("kay lee", Boolean.toString(mExternalStorageWriteable));

return (mExternalStorageAvailable && mExternalStorageWriteable);

}

private void startResult(){

Intent results = new Intent(this, MainActivity.class);

results.putExtra("OCR", true);

//results.putExtra("IMAGE\_PATH", mLocation.getPath());

//results.putExtra("RESULT\_PATH", result.getPath());

startActivity(results);

}

private void startResultSkip(){

Intent results = new Intent(this, MainActivity.class);

results.putExtra("OCR", false);

startActivity(results);

}

private void showToast(){

Toast.makeText(this, "Processing Image, Please be Patient", Toast.LENGTH\_LONG).show();

startResult();

}

// This method skips the preview

private OnClickListener skipListener = new OnClickListener() {

@Override

public void onClick(View v) {

startResultSkip();

}

};

}