

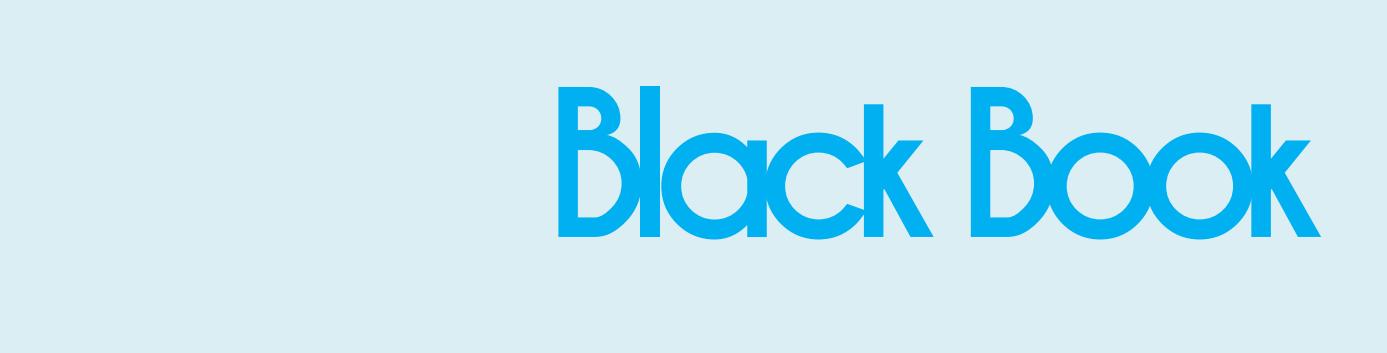
This book contains the details and inner workings of my final year project which consists of an Android OCR app and a dedicated website for the app.

Black Book

T.Y. B.Sc. I.T.
Project Documentation

Meenakshi Madan

TOO LAZY TO TYPE
Android OCR app



Black Book

Acknowledgements

First and foremost I offer my sincerest gratitude to the principal and professors of my college who have supported me throughout my time here, given me valuable knowledge, molded and shaped me into the person I am today.

I'd like to thank our coordinator, Prof. Smruti Nanavaty, for her patience and for sharing her expertise whilst giving me the space to work in my own way.

I cannot thank my family enough for bringing me up the way they did. The source behind my excellence is you.

I'd like to thank Ayush Samantroy for lending me his Android tablet. This has helped me immensely in developing and testing my app.

This list would be incomplete without mentioning all of the developers and education institutes around the world that share their knowledge, work, and wisdom over the Internet.

Preface

It gives me immense pleasure in presenting this book, the Android application, and the website I developed as my final year project.

The idea came to me first while I was pursuing an online course on Machine Learning, but at the time I didn't seem to think much of it. Text recognition seemed too ordinary and I couldn't think of any reason why anyone would want to use it so the idea didn't stick for long and soon I was looking for other ideas. After jumping over several ideas over the course of semester 5, and partially developing and ALMOST finalizing on a word game + Sudoku app, it finally hit me - notice boards! Posters! Street signs! Book names! I noticed how often I'd just click pictures of text instead of writing or typing it down myself. I also noticed several others doing the same. THAT is when I realized how brilliant, but also necessary, making this app was. Immediately I went to test out my theory. Was this even possible? Would Android phones be able to handle the processing required? Would I need to reinvent the wheel or was there a base that I could use? Within a couple of days, I had a prototype ready that answered a few of my questions. I went ahead and submitted a synopsis for it and began working on it.

At the time I barely knew anything about Android devices. Why would I attempt building an Android app then I hear you say? Well, the most obvious reason is that almost everyone I know uses an Android powered device. People love it. And Android users also love downloading third party apps and customizing their device. Chances were that I'd find people willing to look around my app, and possibly some genuine customers.

I quickly began looking into the finer details of developing the application. Once I had the basic setup of OCR, I began thinking about accuracy. Initially, the accuracy was really bad, about 30%. After a lot of research, I discovered what kinds of image processing

were required, and how I could implement them on my app. This process took the longest and was the most frustrating period to go through. It seemed hopeless because I expected 100% accuracy and my app wasn't giving me that. I looked through a few other OCR apps to see what their accuracy was. Some of them gave me hope because if someone else was able to do it, I could figure out a way too.

Another aspect I worked on was training Tesseract, which is the OCR engine I used. Initially, I did this the hard way as the developers mentioned in their documents. This would take up days just to train a handful of fonts. After a while, I discovered an article that used a neat trick to automate some of the steps involved in training the engine. Once I figured out how to use it, training became easy and I was able to train up to 30 fonts, 32 being the maximum number of fonts that the engine can be trained with.

Ultimately, I settled on Imagemagick for the pre-processing part of my app, and then began the phase of trial and error for discovering what filters and what sequence of them would give me a nice clean image. A humongous number of tries and samples later, I found a combination that worked for the app. Things were beginning to fall into place. I was quite happy with the accuracy by this point.

Once I finalized my processing module, I began adding extra features and functionalities to make this a wholesome, harmonious app. I made the graphics myself from scratch using actual 3D models. This took quite long initially, but it made adding changes much easier. Plus, because I had actually hand modeled each of my UI elements, they looked more real and attractive.

I built a dedicated website for this app to form a complete package, authored my black book, and here we are.

In conclusion, I'd like to mention that I enjoyed every moment of coding, even though I wasn't particularly fond of the occasional bug and error message. I enjoyed making the graphics for the application, and I enjoyed writing this book. I hope you enjoy reading the book, using the app, and perhaps for some of you, I hope this helps you in your own projects.

Synopsis

MEENAKSHI MADAN

TY BSc IT
53003100041

December 2, 2012

PROJECT SYNOPSIS FOR AN ANDROID APPLICATION

“TOO LAZY TO TYPE”

(Name will be finalized later)

I. ABSTRACT/MOTIVATION

In today's fast-paced world, people like things in shortcut - quickest, fastest, easiest, least effort path paradigm - and typing out text is no exception. A trend can be observed, majorly in college students, of clicking pictures of whatever they want to look at again. These include pictures of book covers, phone numbers, paragraphs from text books, notice boards, or even a random piece of text they found interesting on a poster in the mall. 6/10 students would prefer clicking pictures instead of typing things out, which is great if you have lots of physical memory on your phone and don't mind searching through thousands of images to find the one picture you're looking for. But what if there was an app on your phone that let you click a picture - just as you normally would - and instead of simply storing the image, the app let you select a region of interest in the image and have it instantly converted to plain text? As a college student who has been strictly following the “click pictures of everything you want to save for later” policy, this app would be immensely helpful for me, in addition to a lot of other people. And that is my motivation for developing this app as my TY project.

II. SCOPE OF SERVICES

To begin with, the app will convert printed text on images to plain text that you can save to a file, copy to clipboard, etc.

The basic outline is as follows:-

- Start the app
- Click on the “Take a photo” button
- This opens up the camera. When satisfied with the positioning and focus/zoom, the user can click the picture or cancel.
- Once the picture is clicked, it is displayed to the user.
- The user can then select a region of interest, say a paragraph on a page in the image, and click OK.
- The app then begins processing the image and outputs a string of text on to the screen.



III. TECHNOLOGIES AND REQUIREMENTS

- The app will run on Android smartphones. The minimum requirement is API 10, but a future update may feature games that require API level 15 or higher.
- Development and testing will be done using an emulator device on a Windows machine, using the Android SDK ADT Bundle and other libraries for efficient and accurate image processing).
- The APK file, i.e. the application itself, will be publicly available for download.
- The Android device will need at least one hardware camera and a camera app (which is usually present by default)

IV. FUTURE IMPROVEMENTS

- Automatically adding tags to images for quicker searches
- Recognizing hand-written text
- Text with messy backgrounds
- Automatically recognizing and saving phone numbers to contacts
- Automatic detection of text regions in scene images
- Improved accuracy and efficiency

V. SUMMARY

This app will serve as a simple but incredibly useful application that uses machine learning and image processing techniques to quickly convert images to text.

Table of Contents

Acknowledgements Preface Synopsis

Engineering Phase

Inception	1
1. The Need.....	2
Problem Definition	2
Existing System	3
Why Android.....	4
2. The Proposal	8
Solution Strategy.....	8
Objectives/Features.....	9
3. Feasibility Study	11
Technical.....	11
Economical.....	12
Legal.....	12
Operational	12
Schedule.....	13
4. Initial Demonstration	14
Elaboration	17
5. WBS and Components	18
Photo OCR Pipeline	18
Components and Modules	20
Test Evaluation Criterion	23
6. Visual Modeling.....	25
Sequence Diagram	25

Use Case Diagram	33
Class Diagram	36
Gantt chart	37

Production Phase

Construction	39
<i>Iterations/Spirals and System Evolution</i>	40
7. Basic Skeleton of the App	41
8. Pre-processing and Training OCR	48
9. Advanced Image Processing and GUI	56
10. Final Touches and Wrap Up.....	62
11. Iteration Summary.....	66
12. The Website	68
13. Graphical User Interface	72
14. APK Structure and Source Code.....	96
Transition.....	114
15. Deployment Pre-requisites and Summary....	115
16. Future Enhancements/Releases	117
User Manual	118
Bibliography	143

Inception

Tasks

- Research and analyse basic idea
- Set goals, objectives
- Synthesize an initial demonstration

Objectives

- Common understanding between co-ordinator and myself on the scope of my project

Deliverables

- Project proposal/synopsis
- Initial demonstration

The Need

Problem Definition

With the drastic increase in the use of digital gadgets such as personal computers, smart phones and tablets, most daily jobs are now performed digitally. This makes everything easier, faster, and more efficient. Given the option, most people today would prefer doing things electronically rather than sitting down with a pen and paper.

In fact, given the option, people would prefer clicking a picture instead of typing things out manually. It is now increasingly common for documents to be scanned so that they can be conveniently viewed and shared via electronic means. However a scan is merely an image capture of the original document, so it cannot be edited or searched through in any way. This results in a decrease in efficiency since people now have to manually correct or search through multiple pages. OCR solves this problem by making the document text searchable.

What such users need, is an application that will run on their smart phones and quickly and accurately get them the text they want so that they can edit it, paste it elsewhere, Google it, or make some other use of it.

Such an application would increase the efficiency and effectiveness of work. The ability to instantly search through content is immensely useful. This is especially useful when the user is on the go and may not have time to manually type out a piece of text. You can now use the copy and paste tools as well, instead of rewriting everything to correct it.

Existing System

The technology that will provide the required features is called Photo OCR. This section describes what OCR is and the current state of OCR technology on mobile phones.

What's text recognition? Part of **Machine Learning**, optical character recognition (OCR) and can be really useful when you've got a **paper document you want in digital, editable form**. You need a scanner to create an image of the document first, but then once you have that image you can run it through an OCR application that will **read each character and attempt to recreate the original document as text**. In most cases it will never be perfect, but with a high-quality scan you can come pretty close. **This is even possible on your phone, which is what I propose to develop.**

Since OCR technology has been more and more widely applied to paper-intensive industry, it is facing more **complex images environment** in the real world. For example: **complicated backgrounds, degraded-images, heavy-noise, paper skew, picture distortion, low-resolution, disturbed by grid & lines, text image consisting of special fonts, symbols, glossary words** and etc. All the factors **affect OCR products' stability in recognition accuracy**.

In recent years, the major OCR technology providers began to develop **dedicated OCR systems**, each for special types of images. They combine various optimization methods related to the special image, such as business rules, standard expression, glossary or dictionary and rich information contained in color images, to improve the recognition accuracy. Such strategy to customize OCR technology is called "Application-Oriented OCR" or "Customized OCR", widely used in the fields of Business-card OCR, Invoice OCR, Screenshot OCR, ID card OCR, Driver-license OCR or Auto plant OCR, and so on. **However, what most people require is general-purpose OCR software. This means that the user is not restricted to a specific type of image, but the application will recognize text from almost any type of image.** This is **what my app gives you.**

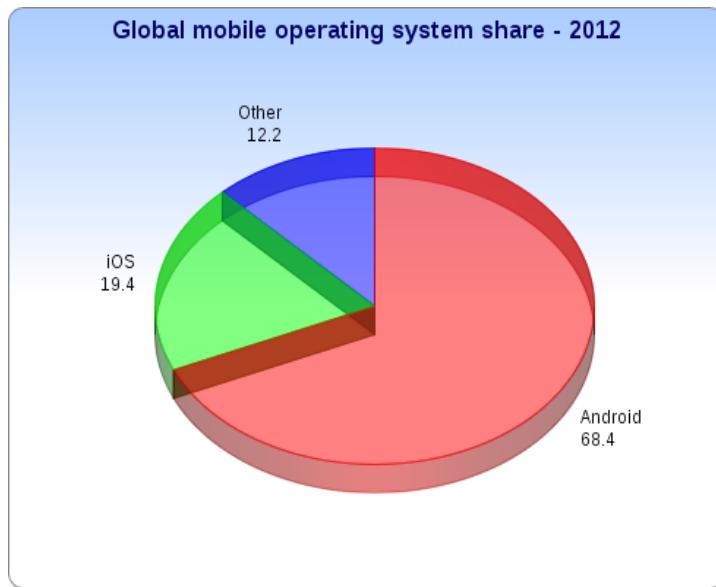
OCR is a delicacy on PCs; the technology itself has advanced leaps and bounds, but the price of this technology is still out of the reach of many. The good, quality OCR programs on Windows cost an arm and a leg while the cheap(er) OCR programs are not very accurate. However, as polarizing as OCR may be on Windows, at least one has a choice of either shelling out lots

of \$\$\$ to get a quality OCR program or not. On Android, unfortunately, there isn't as much of a choice. **OCR apps have not yet penetrated the Android ecosystem as well as one may think. There are only a handful of OCR apps out there, and OCR on Android is non-perfect and is still in its infancy.** One cannot expect one's phone to outdo one's computer in performing OCR. In fact, one can't expect one's phone to do anything more than basic OCR operations. But now, with my app you can.

Why Android

Biggest Addressable Smartphone market

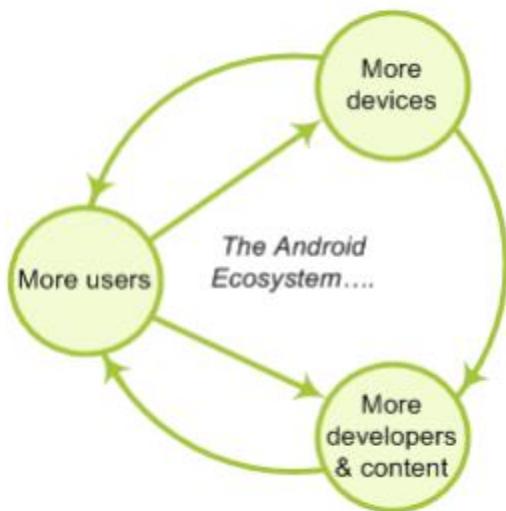
Both the iOS and Android ecosystems are huge and filled with opportunity. However, an increasing amount of research indicates that the Android platform is larger and growing faster than iOS. Most recently Nielson reported that Android has achieved an overall 68.4% market share versus 19.4% for iOS. In addition, the NPD Group released a study stating Android had over 60% of the US activations in Q1 2012 versus 29% for iOS. According to Google, 850,000 Android devices are activated every day.



[Source: <http://venturebeat.com/2013/01/28/android-captured-almost-70-global-smartphone-market-share-in-2012-apple-just-under-20/>]

Open Source

Another great advantage of Android over similar platforms is the fact that Android is open source. Meaning, that no industry player, not even Google, will be able to restrict its development or introduce any changes that would go against your interests. Developers have full access to the phone's functionality, like sending/receiving texts (SMS), using camera or even handling phone calls. And another, probably one of the most important, advantages of being open source is that all the newest revolutionary technologies can be introduced to Android by the various developers who work on it. That's why for Android open source is one of the most important advantages.



[Source: <http://nickallott.com/?p=29#>]

More Innovative

iOS powered devices are released usually once or twice a year, as opposed to dozens of Android devices being released all over the world every year. The result of this is new technologies and features being available to the phone users more often, and therefore future releases of my application with enhanced feature sets will be possible.

One iPhone launch/year may not be enough

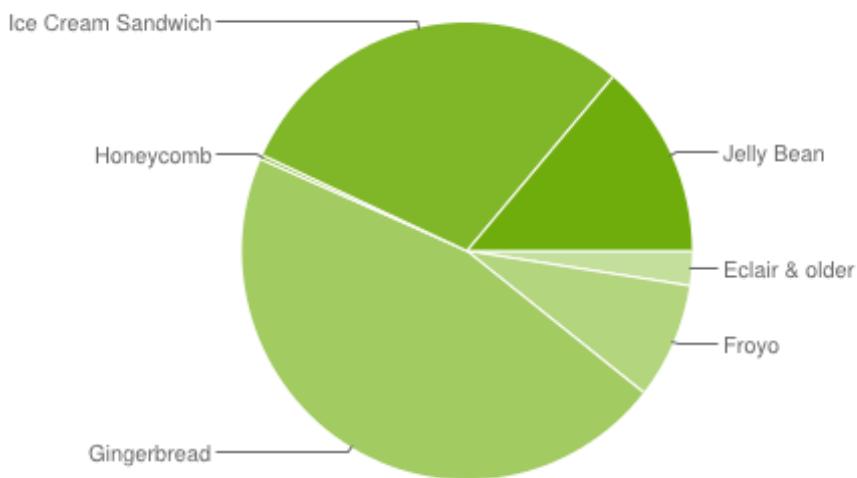
Major Android devices launched in 6 months in the US during 2012

Jan-12	Feb-12	Mar-12	Apr-12	May-12	Jun-12	Jul-12	Aug-12	Sep-12	Oct-12	Nov-12	Dec-12
	Motorola Droid 4		HTC One	HTC EVO 4G, Motorola Droid RAZR MAXX	Samsung Galaxy SIII			iPhone 5	Samsung Galaxy Note 2, HTC One X+, Motorola Droid RAZR HD	Google Nexus 4, LG Optimus G	

Source: Piper Jaffray, OEM press releases

The following pie chart and table is based on the number of Android devices that have accessed Google Play.

Data collected during a 14-day period ending on February 4, 2013



Version	Codename	API	Distribution
1.6	Donut	4	0.2%

<u>2.1</u>	Eclair	7	2.2%
<u>2.2</u>	Froyo	8	8.1%
<u>2.3 -</u> <u>2.3.2</u>	Gingerbread	9	0.2%
<u>2.3.3 -</u> <u>2.3.7</u>		10	45.4%
<u>3.1</u>	Honeycomb	12	0.3%
<u>3.2</u>		13	1.0%
<u>4.0.3 -</u> <u>4.0.4</u>	Ice Cream Sandwich	15	29.0%
<u>4.1</u>	Jelly Bean	16	12.2%
<u>4.2</u>		17	1.4%

[Source: <http://developer.android.com/about/dashboards/index.html>]

The above statistical analysis shows that Gingerbread 2.3.3 is the most common version among Android users. Thus, I decided that API level 10 would be the target API for the application.

The Proposal

Solution Strategy

The solution is to use concepts from Computer Vision (A.I.) to recognize text from images to perform this task. *Optical Character Recognition* is a technology that allows a computer to create text strings out of scanned images or, in our case, an image that is captured with a user's smart phone/tablet.

The basic needs and flow of the application are:

- User starts app
- User clicks button to start OCR
- User holds phone over a piece of text and clicks the picture
- App processes picture
- Text is displayed to the user

The app will turn it into copy-able text you can then paste anywhere in your phone—a document editor, your note-taking app, Gmail, SMS, or anything else you could imagine.

To make the above possible, the following input \Rightarrow process \Rightarrow output layout is required.

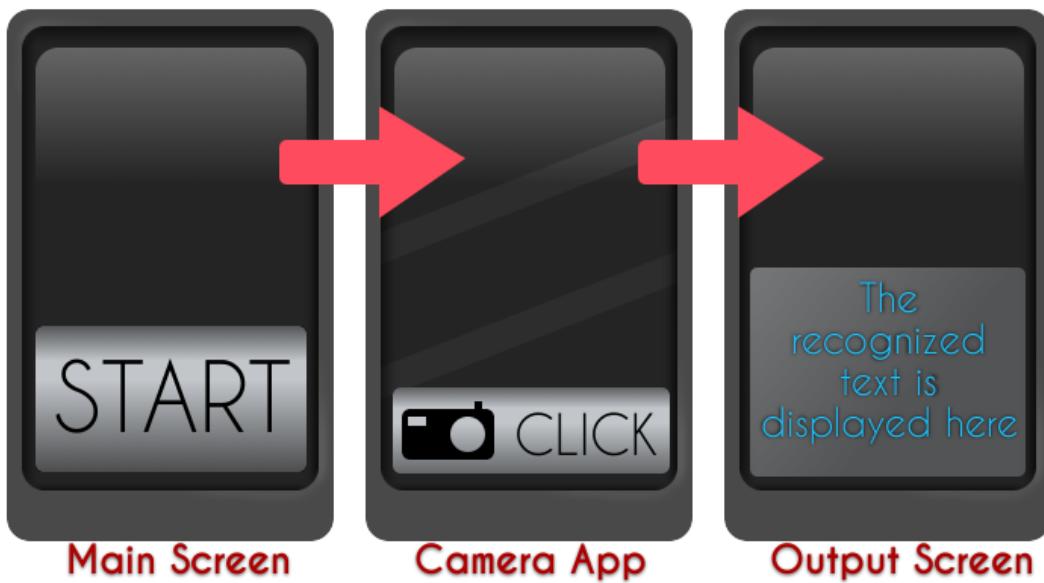
Input \Rightarrow The ability to click a picture from within the app

Process ⇒ The actual text recognition module

Output ⇒ The recognized text

Additionally, some pre-processing might be required to clean up noisy images for better accuracy.

Minimal user interface requirements are demonstrated as follows:-



Objectives/Requirements/features

- ✓ Quick and easy image-to-text **on the go**
- ✓ **Lightweight** application
- ✓ Supported by highest possible percentage of **Android phones**
- ✓ Reasonably high and consistent **accuracy** of OCR
- ✓ Reasonably less **response time**
- ✓ Support for a reasonable variety of **font types and sizes**
- ✓ Attractive but easy to use **interface**
- ✓ Self-contained application requiring **no internet and data connection charges**

- ✓ Basic **help screen** for detailed usage help
- ✓ Allows users to perform OCR on **new images** (taken using camera) or **images already saved** on phone
- ✓ When taking new images, users can **define a specific rectangular area** to analyse
- ✓ OCR'ed text can be **copied to Android clipboard**, from where users can paste it in whichever app they want (SMS/text, e-mail, document editor, etc.)
- ✓ OCR'ed text can be **saved as a text file**
- ✓ OCR'ed text can be quickly **Googled** (only feature that understandably requires an Internet connection)
- ✓ OCR'ed text can be **shared** with any other app installed on your phone that accepts text like **Facebook, SMS, Bluetooth, Gmail, Twitter, Whatsapp, Dropbox, Google talk**, etc.
- ✓ User can view OCR'ed text **history** (5 most recent)
- ✓ When taking new images, users can on phone's **flash** (if applicable)
- ✓ Recognizes **line-breaks**
- ✓ Doesn't save any information on 3rd party systems so **user's data is completely private**

Feasibility Study

TELOS refers to the five areas of feasibility - Technical, Economic, Legal, Operational, and Scheduling. All of these factors were analysed and studied. A highly brief summary of the report and results are given below.

Technical Feasibility

To develop such an application, the following technical skills and tools are required.

- Artificial Intelligence and Machine Learning (Photo OCR)
- Java Programming Language
- Android Software Development Kit
- Eclipse IDE (optional but recommended)
- Android Emulator and/or Android device

I have worked on softwares that use artificial intelligence and machine learning in Python and Octave before this project. I have also studied the Java Programming Language extensively. Transition is expected to be smooth.

All of the above technical requirements are fulfilled. The project is technically feasible.

Economic Feasibility

Considering that this is an under-grad project, ROI and profit in terms of money is not expected. However, I must and did take into consideration whether developing the product itself was within budget.

The cost of a suitable workstation, libraries, test device (or emulator), and other components was found to be within budget. The project is indeed economically feasible.

Since I am distributing the app free of charge, and no Internet/data charges are mandatory, the product is economically feasible for the end user.

Legal Feasibility

No data is saved by the application without the explicit intention of the user. Therefore, data privacy and other similar legal issues do not apply.

The only condition under which the legality of this application would come into question is if the application was used to extract text from confidential or legally protected documents. The developer, however, has no control of how the user makes use of the application and hence is not responsible for it and is absolved of any liabilities arising therefore. The product is legally feasible.

The product is open source and the code can be used and modified.

Operational Feasibility

The product has high applicability among a variety of users. Being lightweight, accurate, and easy to use, a wide user base is expected.

A study of Photo OCR applications in the Android market was performed. If we were to compare this application with other applications in the field of Photo OCR we would find that most applications are limited to a specific type of

photo, such as barcodes, automobile number plates, business cards, telephone numbers, names, book covers, etc. In contrast, this application is a general purpose OCR with which any type of photo can be used as input.

The study also revealed that while almost all OCR applications in the market needed an internet connection at some point of time to perform even the most basic OCR function, this application is entirely self-contained. No access to internet is required for basic functions.

Another point that came up during the analysis was the accuracy. The accuracy with respect to this application is how correctly the application recognizes the text. Most applications had poor to average accuracy. Only 3 applications were found to have good to excellent accuracy (with a hit or miss chance), but with other limitations. My app achieves 95%+ accuracy with most images.

The operational environment specifications were taken at the lowest level possible (lowest API level of Android that would support all the basic features needed in the application). Simply stated, this means that the application will run on a lot of devices. Choosing a higher API level would have meant a richer, more efficient feature set, but that would reduce the possible user base by 40% to 50%.

The Operating System (Android) itself is available free of charge.

The product is operationally feasible.

Schedule Feasibility

I have worked on similar projects before both, in the field of A.I., as well as Android applications (Android games, to be specific). Taking past experience as well as expert advice into consideration, the effort and schedule break up showed that the project was likely to be delivered on time.

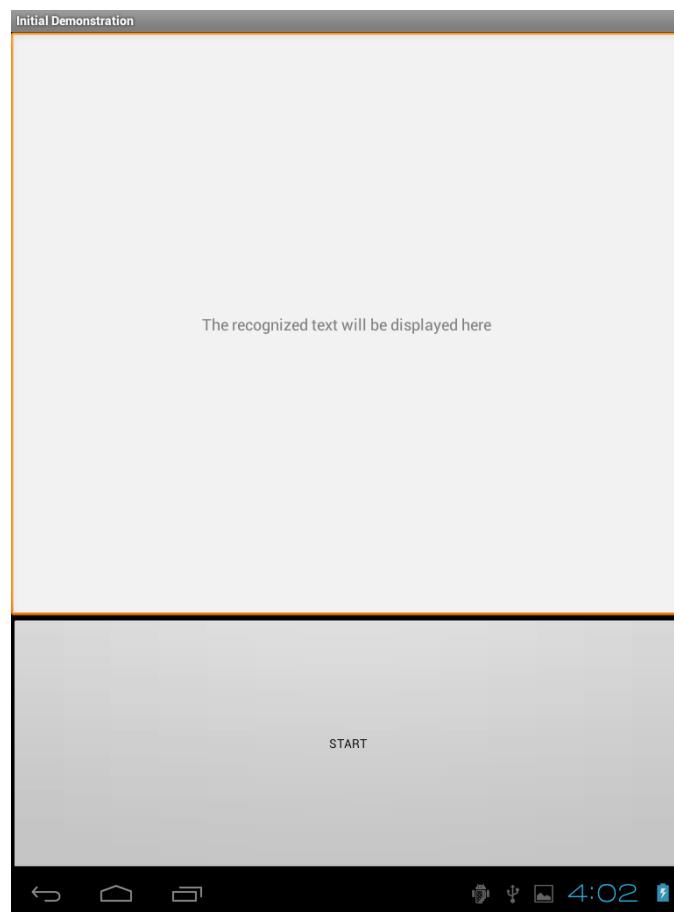
Initial Demonstration

A **throwaway prototype** was developed to demonstrate that the critical use-cases, primary scenarios and requirements were satisfied. This prototype also tested the performance, response time, capacity, accuracy, and range of the application. The main concerns that were addressed are as follows:-

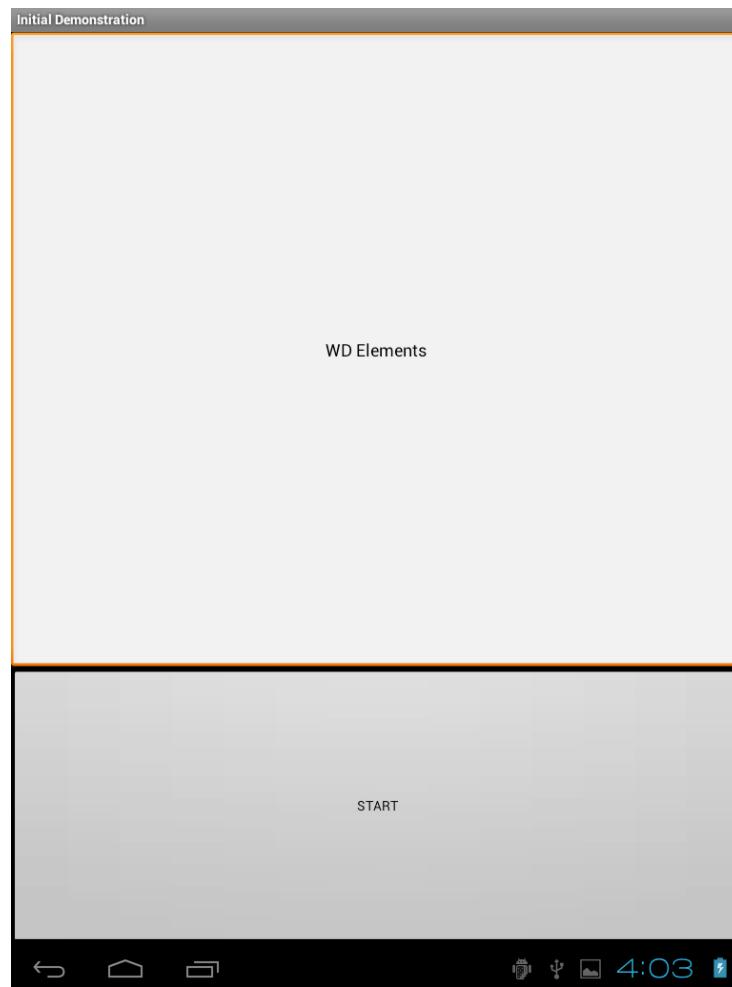
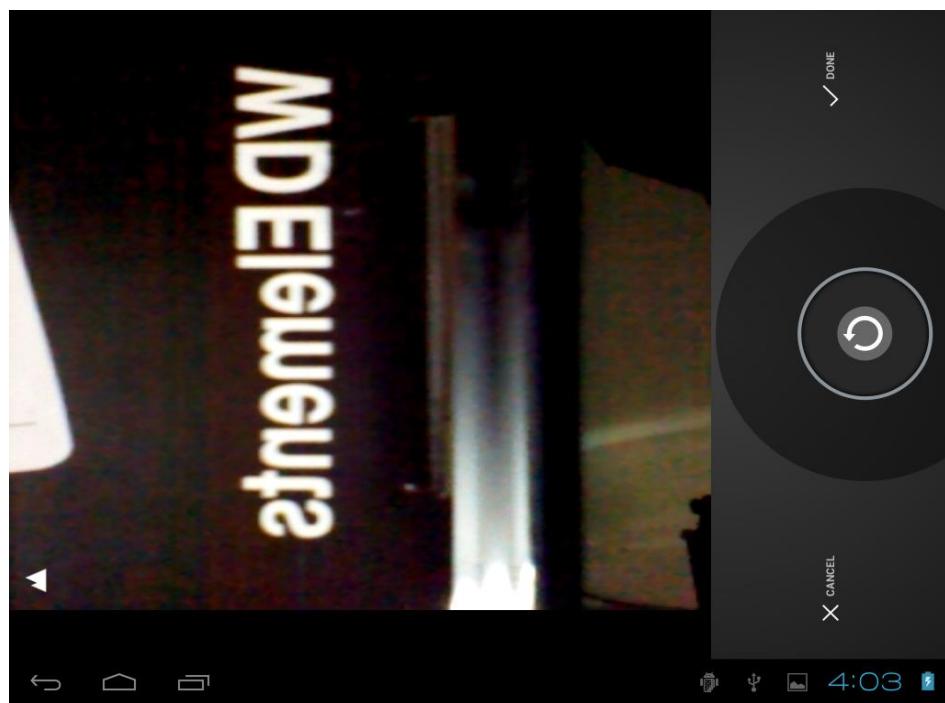
- Does the application start successfully?
- Do all the buttons work?
- Does the application successfully initiate the camera app?
- Does the camera application successfully click and save the picture at the desired location?
- Does the OCR process initialize correctly?
- Does the application display any text?
- How accurate is the text?
- How long did the processing take?
- Does the application work on a variety of pictures?
- Does the application crash at any point?

The results of this throwaway prototype are summarized in the following table. The screenshots of the prototype are displayed next.

<u>Test</u>	<u>Result</u>
UI elements	All function correctly
Camera app initialization	Successful
OCR process initialization	Successful
Text display	Successful
Processing time (in milliseconds)	361
Accuracy	Good
Any crashes to report	No
Overall demo	Successful



TOO LAZY TO TYPE



Elaboration

Tasks

- Thorough research of concepts and technologies
- Based on research, develop system design
- Identify modules, components
- Develop evaluation criteria (exit criteria)

Objectives

- Detailed design based on solid research

Deliverables

- UML Diagrams demonstrating system design

WBS and Components

To formulate a work-breakdown structure and take make/buy decisions of custom, commercial, and open source components, we first need to examine the Photo OCR Pipeline. This will help me decide how to break down my project into a sequence of different components and modules, which is discussed next.

Photo OCR Pipeline

1. Text Detection

First we go through the image and find the regions where there is text in an image. For example, here is a picture taken from my smart phone's camera of a box of Bisleri water bottles. The highlighted area shows some text that a Photo OCR system may find.



2. Character Segmentation

Second, given the rectangle around the text region, we can do character segmentation where we segment the rectangle out into the locations of the individual characters.



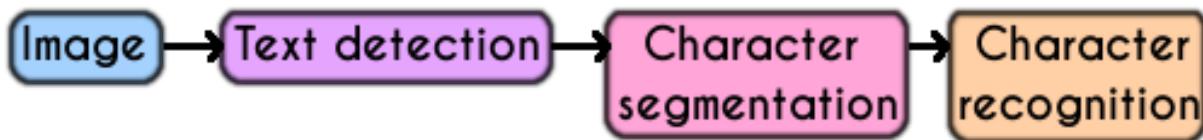
3. Character Classification

And finally, having segmented it out into individual characters, we can then run a classifier which looks at the images of the individual characters and tries to figure out the letters.



And so on, till you find out what all the characters are and what the entire text is.

A system like this is called a **machine learning pipeline**, and in particular, the photo OCR pipeline is depicted as follows.



Components and Modules

As identified in the Inception Phase, the natural break-down of the system is input \Rightarrow process \Rightarrow output.

Input

The input will consist of two levels.

At the first level the user selects whether he wants to select an existing image from the gallery OR click a fresh image using the phone camera.

At the second level, that is once the user has selected the image, user may crop out a part of the image that he wants to perform OCR on.

Process

Unless high customization is required, it is always a good decision to avoid re-inventing the wheel. Further, the time and cost of producing a custom component that implements the photo OCR pipeline is quite high, and given the schedule and budget allotted to my project, it would be wiser to use a

partially implemented component than creating one from scratch. After research and testing of several OCR libraries and implementations of the photo OCR pipeline, Tesseract was found to have the highest accuracy. However, the accuracy was found to be high only when it ran on a completely clean image with a white background and black text. To make this work on an Android device, some amount of pre-processing would be required.

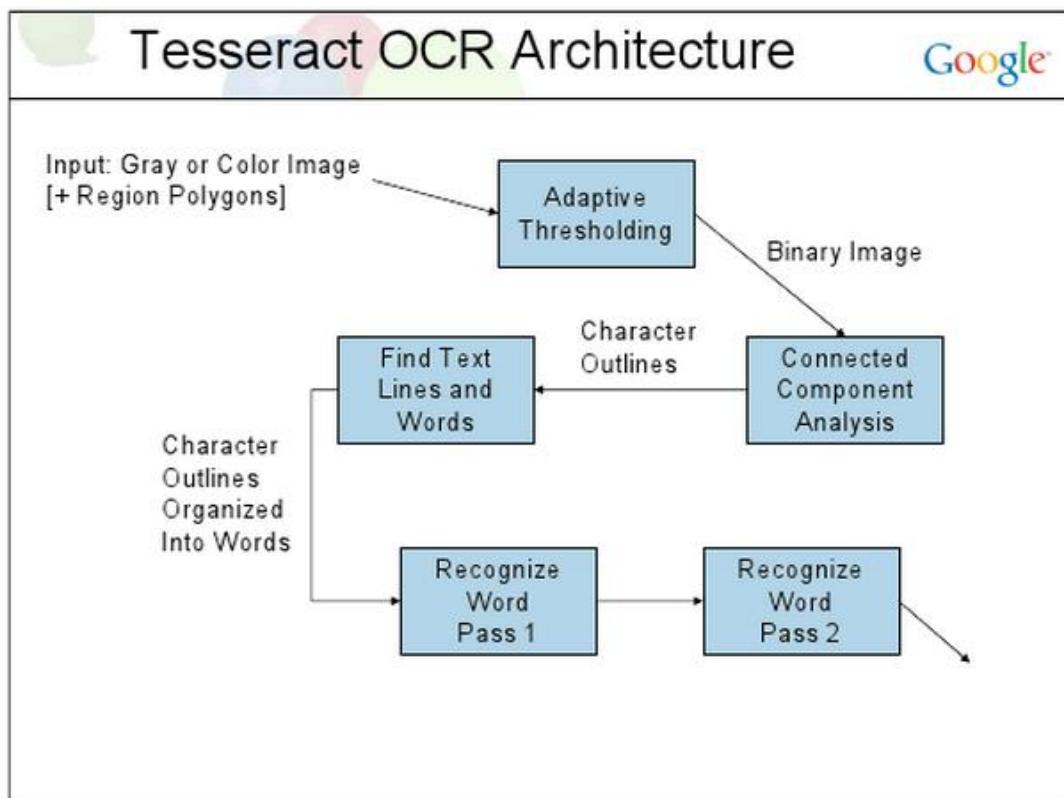
Thus, at least two modules are required for the process component.

1. Pre-processing module

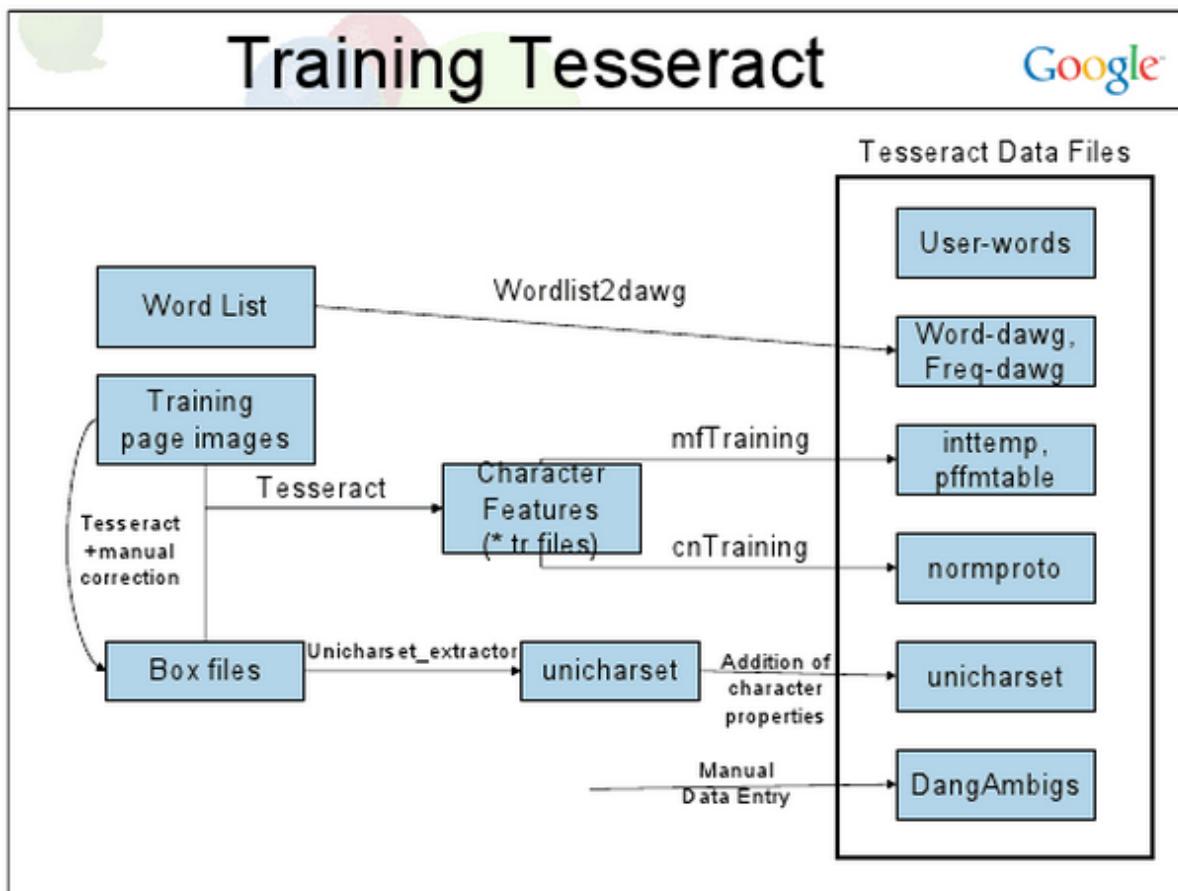
This component will consist of a sequence of image processing algorithms that the input image will go through before being fed to the next module.

2. OCR module

I will use a trainable OCR engine called Tesseract, described below.



The above diagram shows what functionalities the Tesseract engine provides. Described below is the process I will follow to train Tesseract.



[Source for both: <http://tesseract-ocr.googlecode.com/files/TesseractOSCON.pdf>]

Output

This is a simple component that simply displays the text to the user.

Other functionalities

1. Copy the text to the Android clipboard
2. Save the text to a file on the Android phone
3. Open Google search for the text

Test Evaluation Criteria

Once the **smoke test** is done, I then need an evaluation criteria for further tests/assessments, and the parts of the system that I need to work on. Not much can be further improved in terms of performance and accuracy for the input and output components, and so only the “process” component is discussed here.

Ceiling analysis: *What parts of the pipeline to work on to improve overall system performance. (Estimating errors due to each process)*

It is important to have a single real number **evaluation metric** for this learning system. This metric in my project is **character level accuracy**. This metric measures the following: given a test set image, what is the fraction of the characters in the test set image that the application recognizes correctly?

The steps I follow to perform a ceiling analysis on my application is as follows.

1. First I evaluate the accuracy of the overall system. That is, the accuracy of character recognition delivered after the image is processed through each of the modules. I do this by testing the application with a set of test images which I call “test set”.
2. Once that is done, I determine if the accuracy is good enough (80% and above) or if I need to improve some of my modules to produce better accuracy. The 2 modules I check are pre-processing and OCR.
3. I then simulate what happens if I have a pre-processing component that works 100% correctly. That means, I will manually edit the images to produce completely clean black/white images and feed these images into the next stage, which is the Tesseract engine. I then measure the evaluation metric to check what the overall accuracy is.

4. The next step is to simulate 100% accuracy for both components. This would obviously lead to 100% overall system accuracy and hence I will not actually perform it.
5. By subtracting overall System accuracy from pre-processing accuracy, I see the performance gain or increase in accuracy of my application by working on the pre-processing component. This can be achieved by using different filters, image processing techniques, etc.
6. By subtracting pre-processing accuracy from 100, I see the performance gain I would achieve by working on the OCR component (Tesseract engine). This can be achieved by training the Tesseract engine further, using different sets of initialization parameters to the component, using a different OCR library, etc.
7. I will then repeat steps 1-6 over several iterations until I am satisfied with the accuracy of my application.

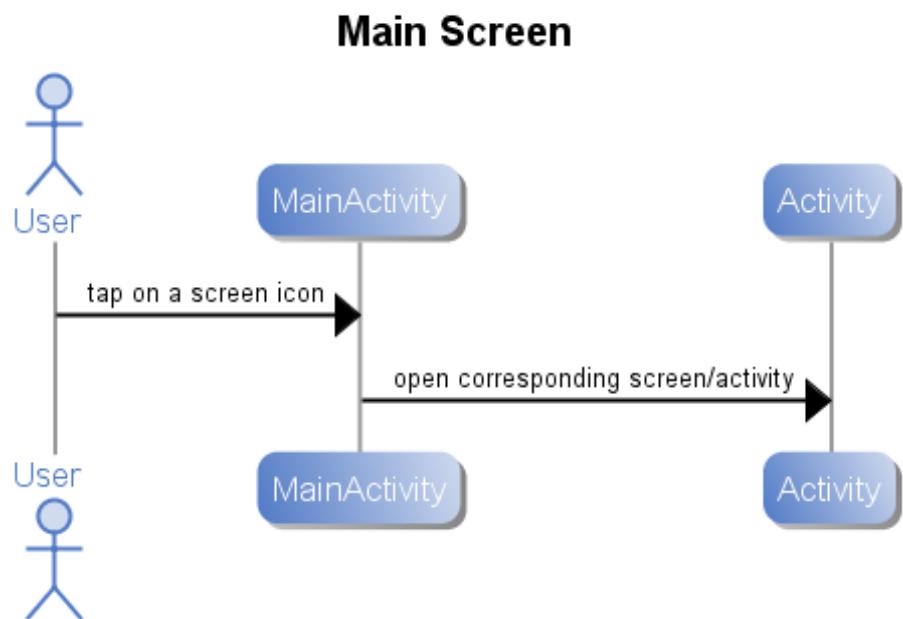
By performing the above sequence of steps, I will be able to determine which component needs to be worked on further to improve my app's performance. This is an efficient and effective way of improving system performance as well as managing schedule, cost and other resources.

Component	Accuracy
Overall System	X%
Pre-processing	Y%
OCR engine	100%

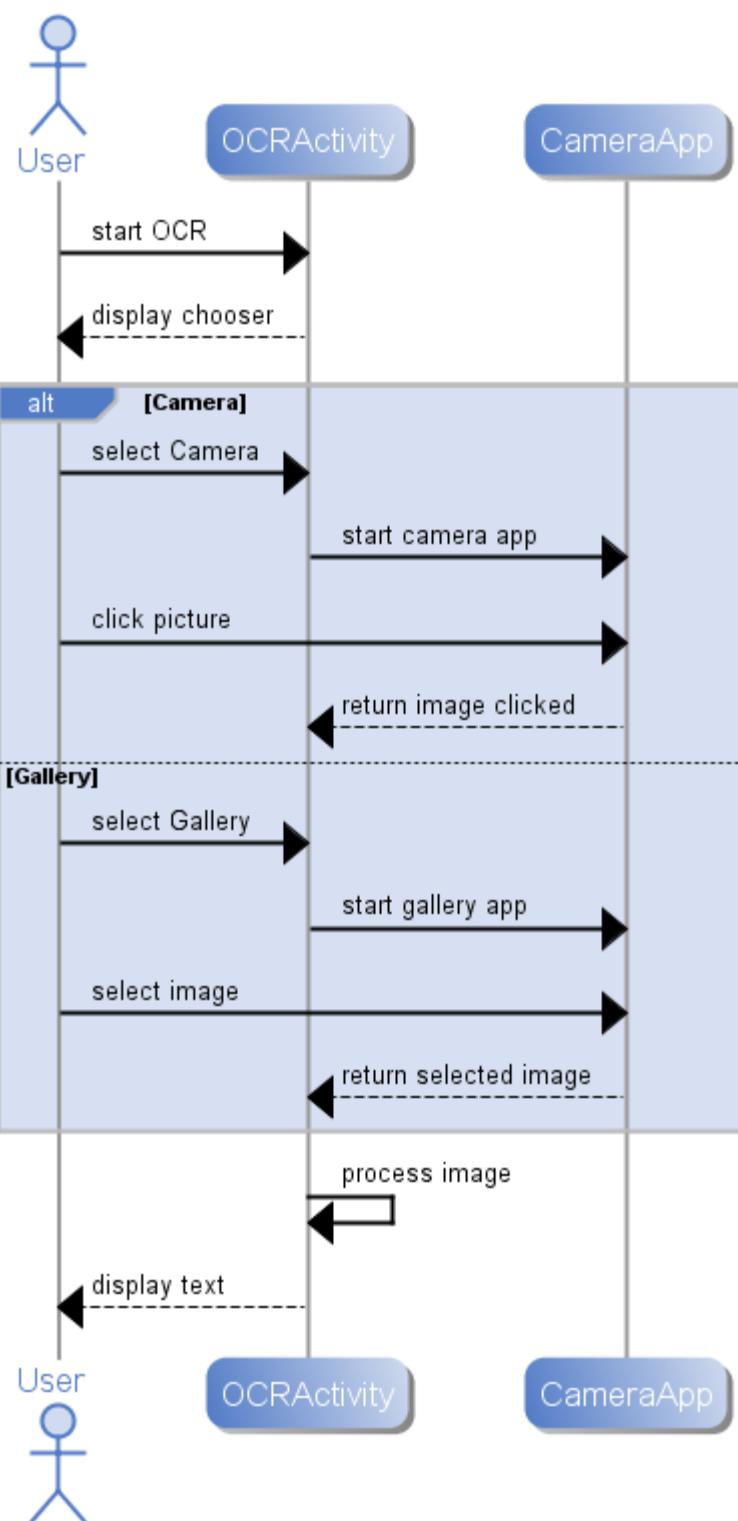
In the production phase section of this book I go into the details of how I implemented this method as well as the results they produced.

Visual Modelling

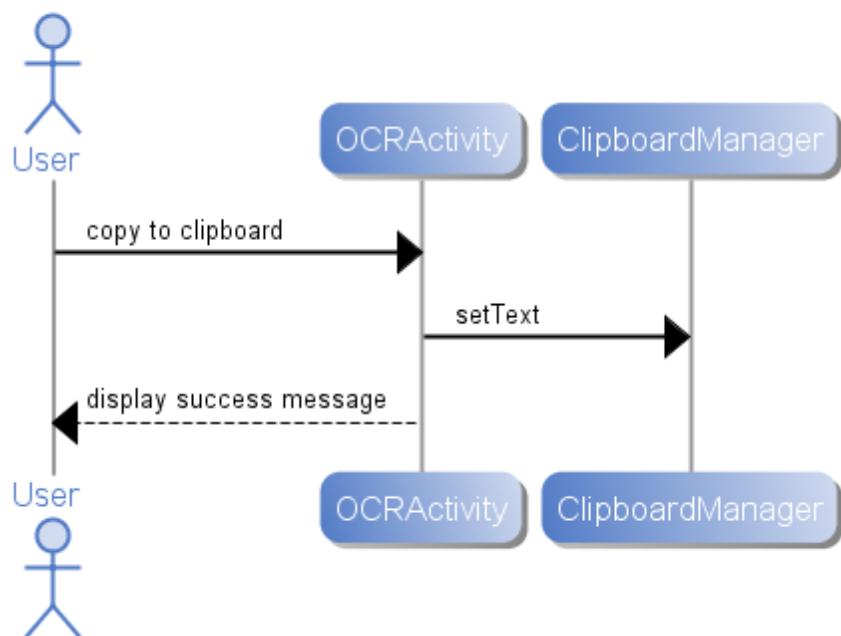
Sequence Diagrams



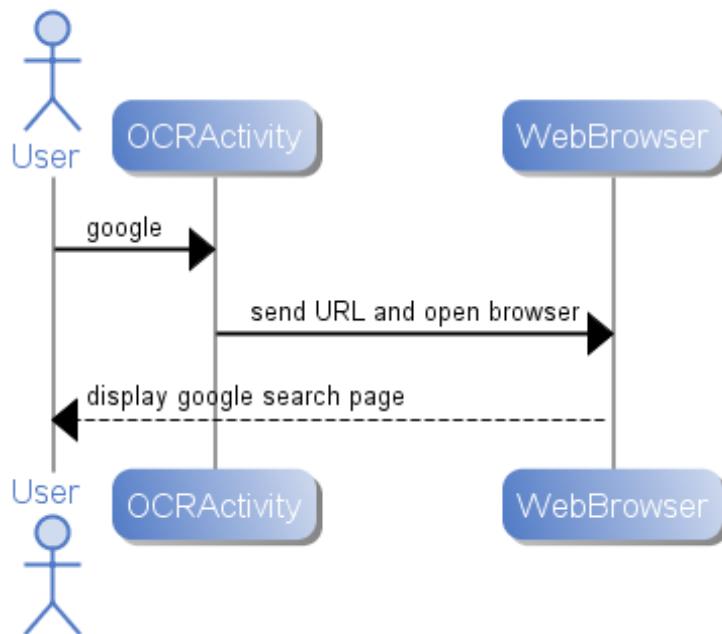
OCR Screen : OCR

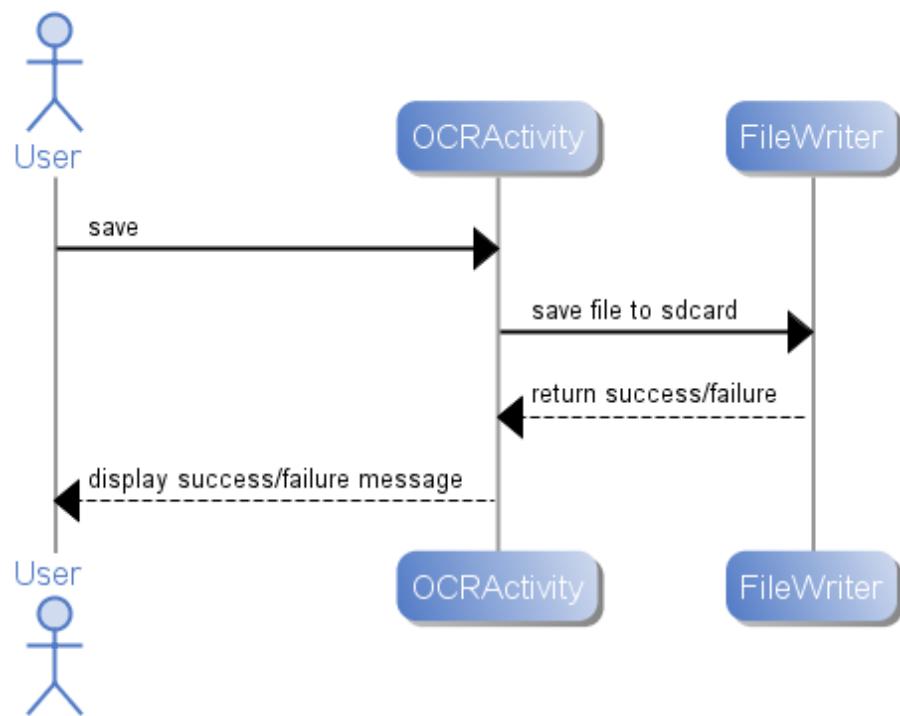
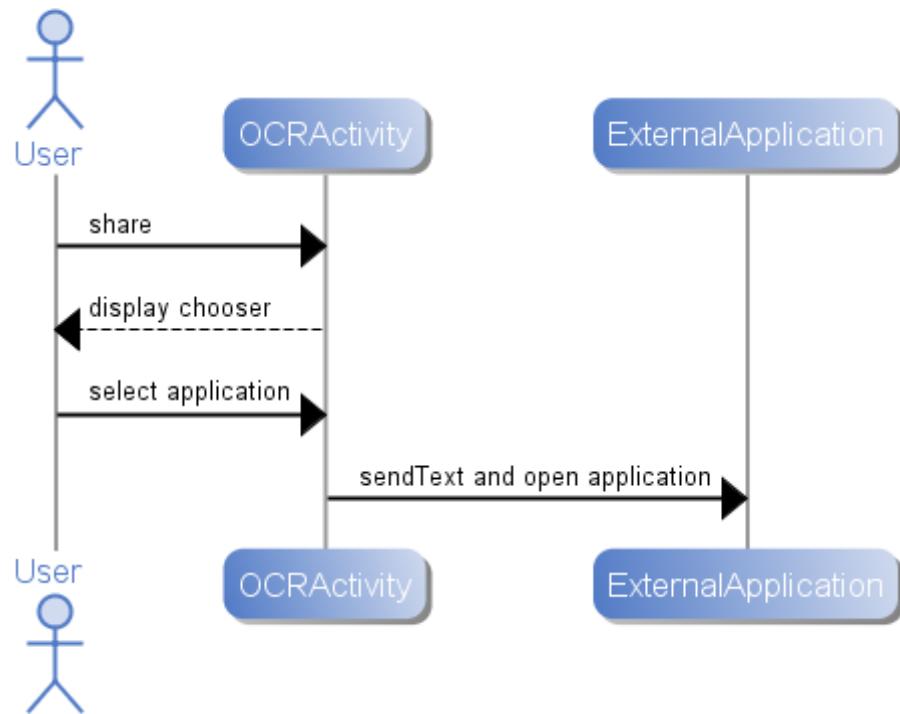


OCR Screen : Copy To Clipboard

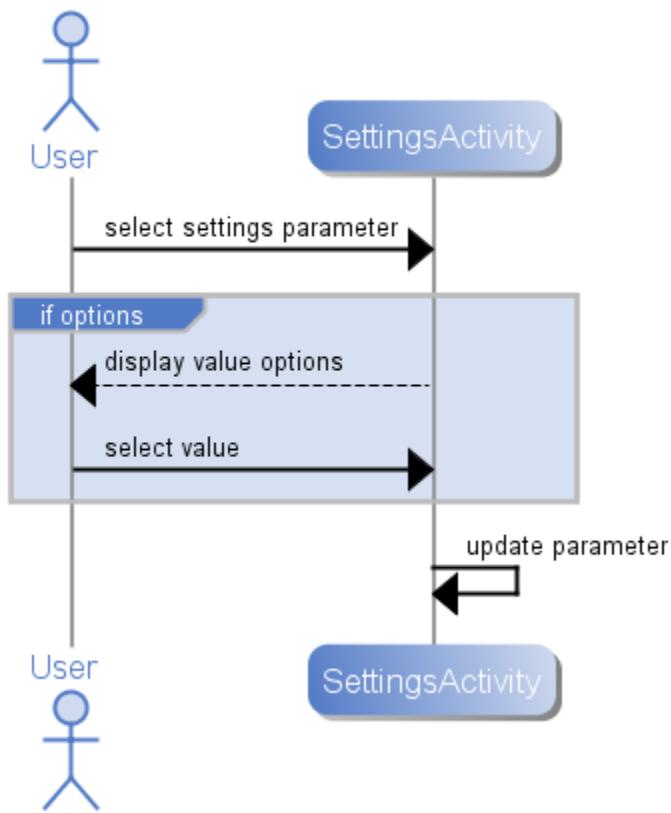


OCR Screen : Google

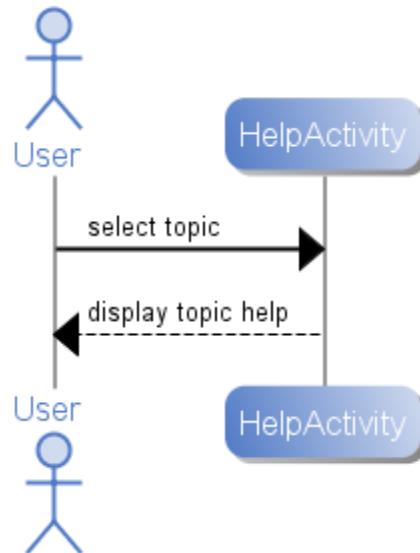


OCR Screen : Save**OCR Screen : Share**

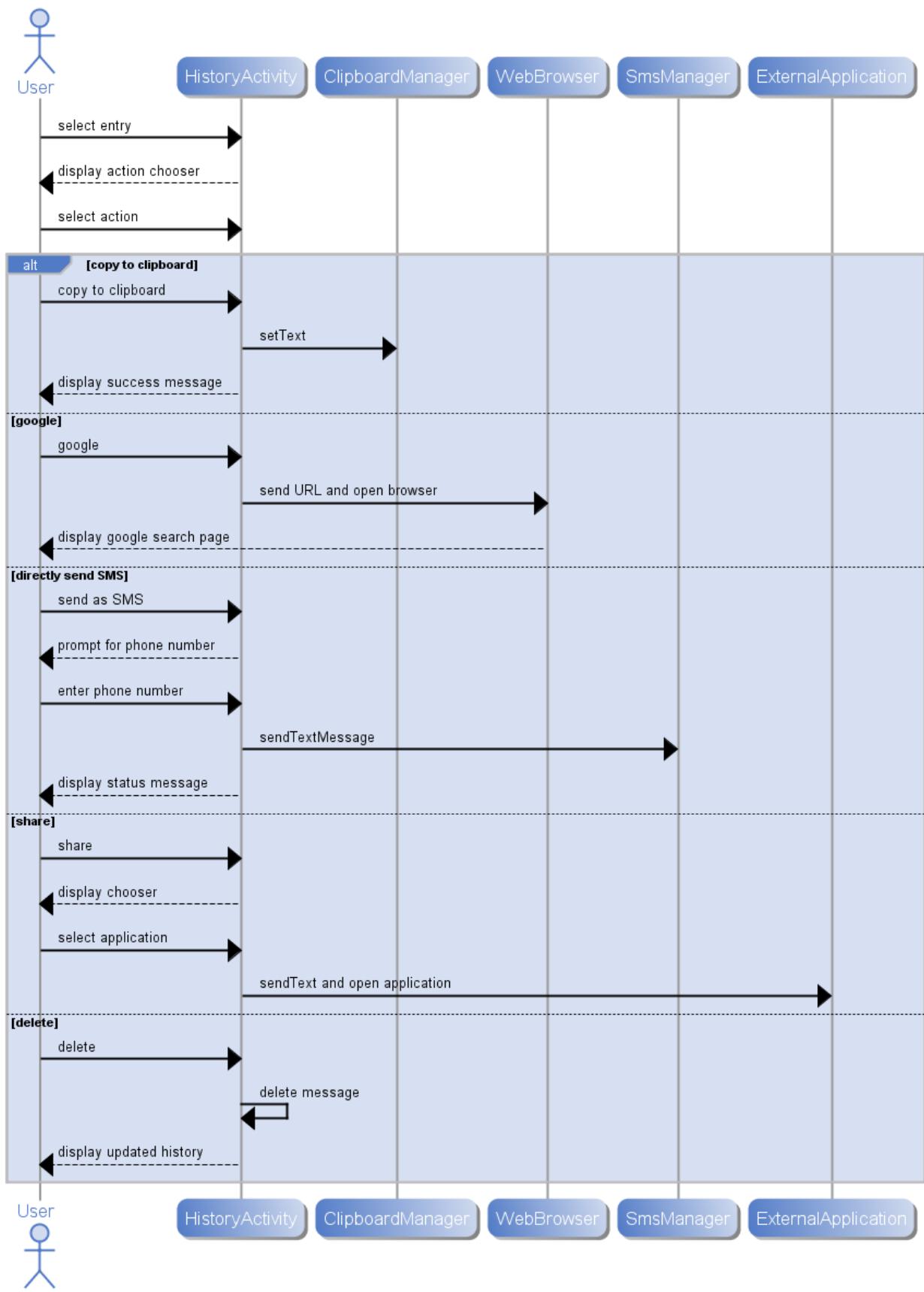
Settings Screen



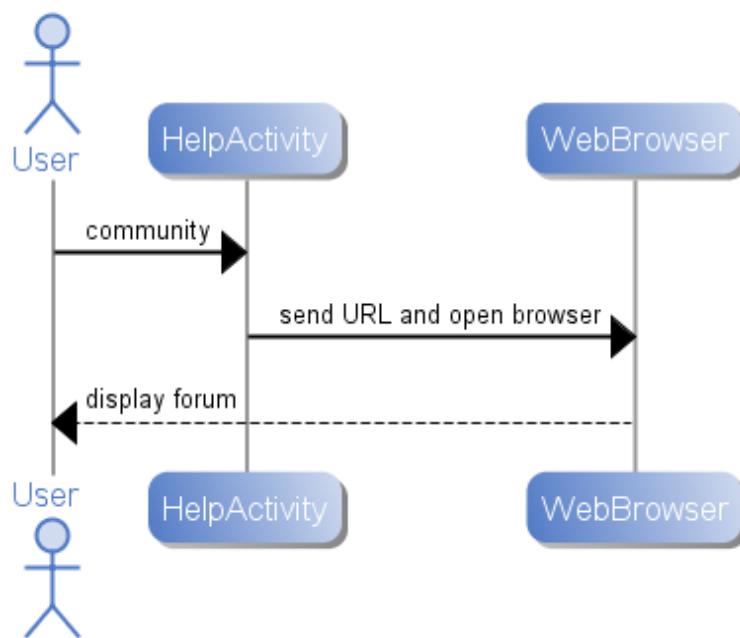
Help Screen - tutorial, buttons, about, tips



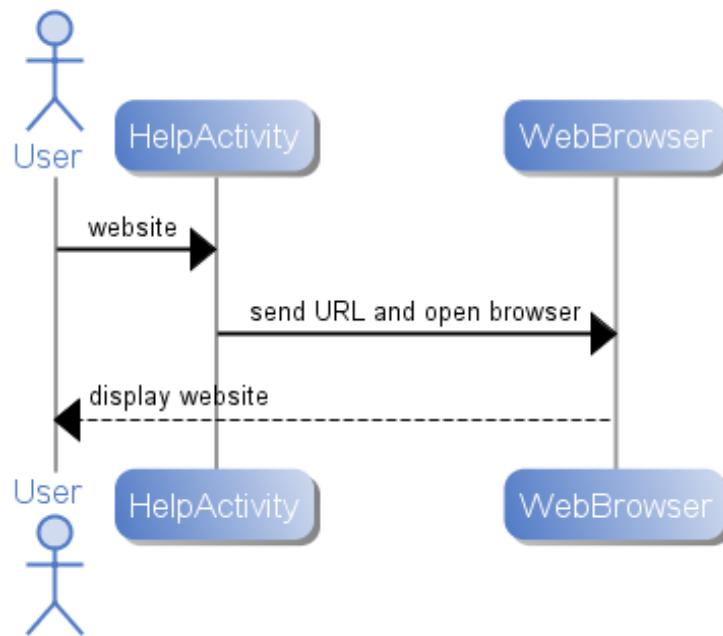
History Screen



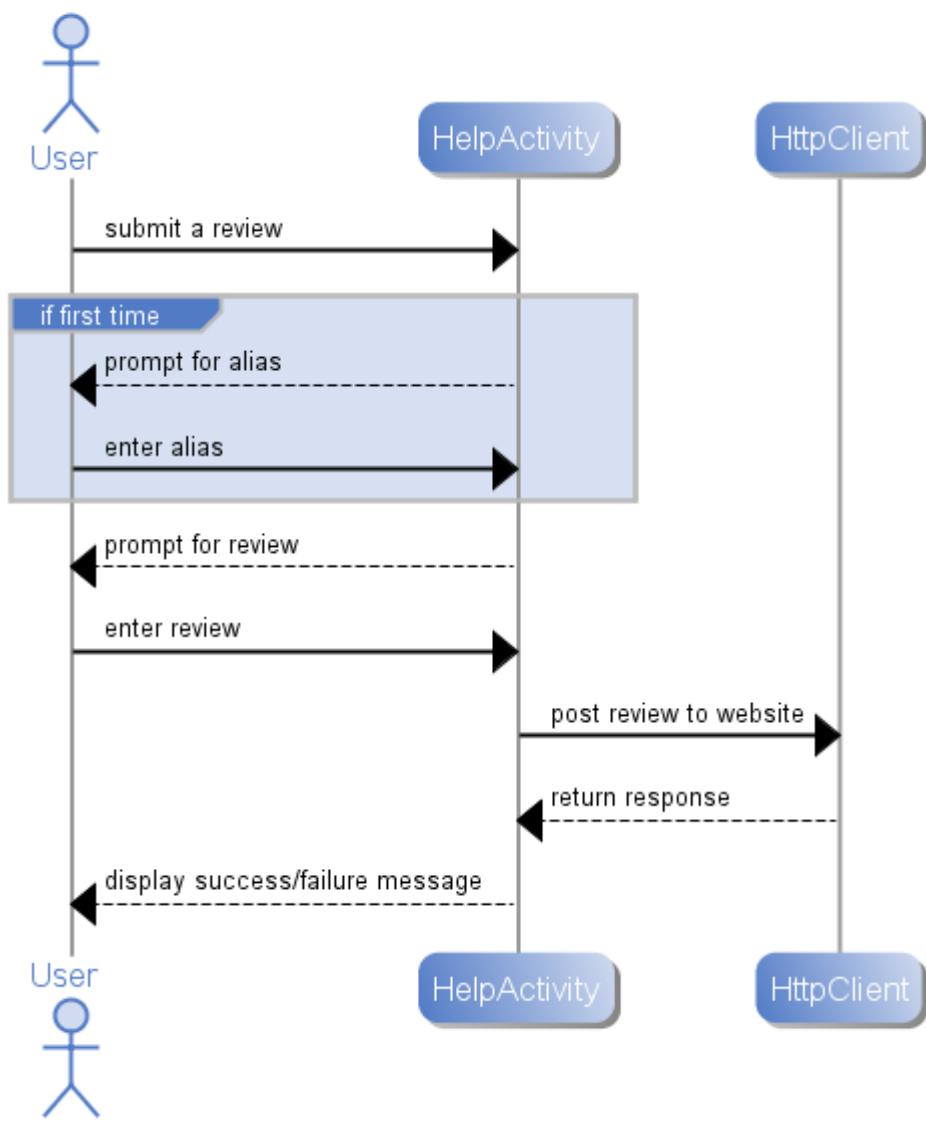
Help Screen - community



Help Screen - website

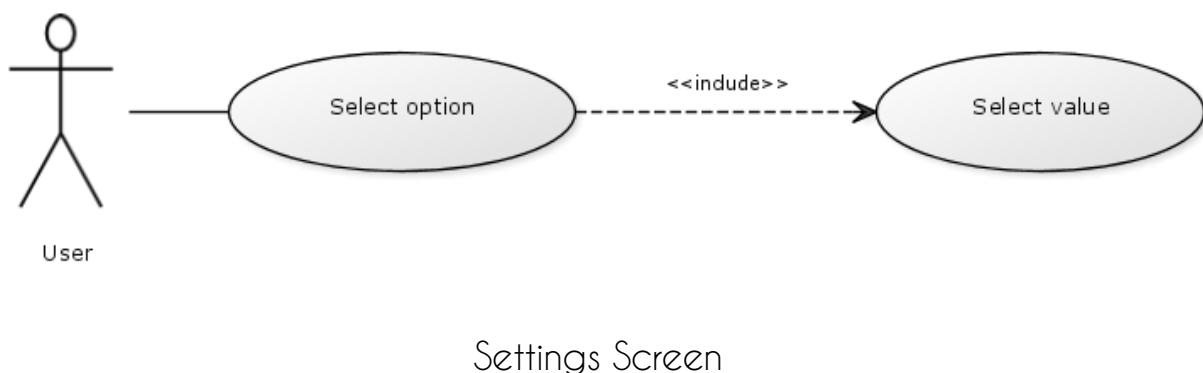
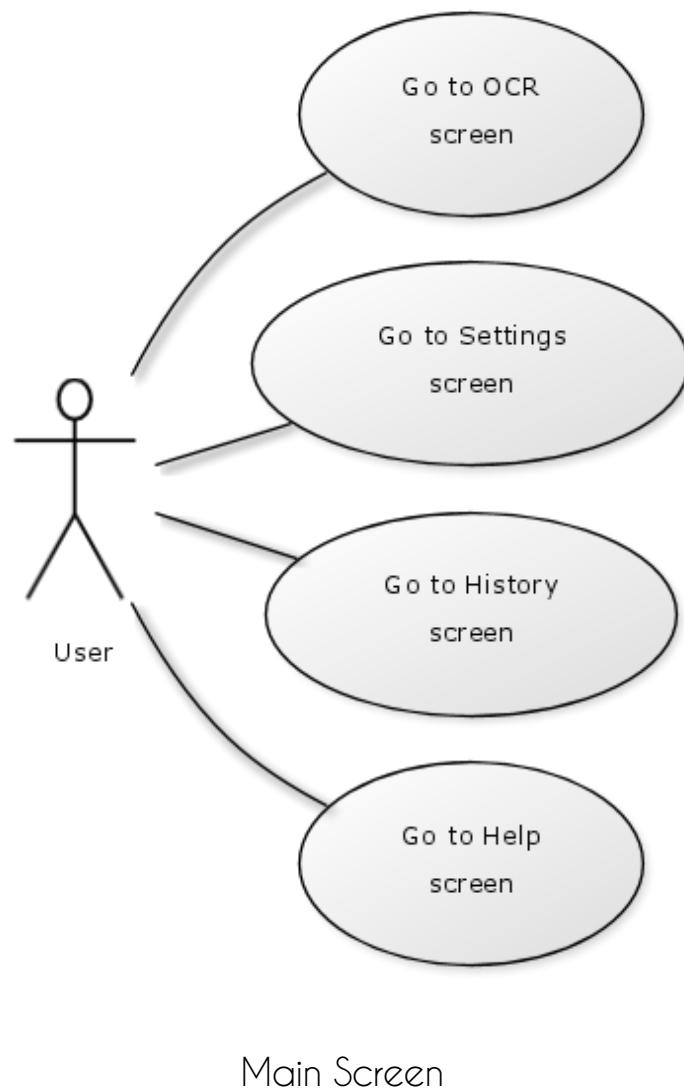


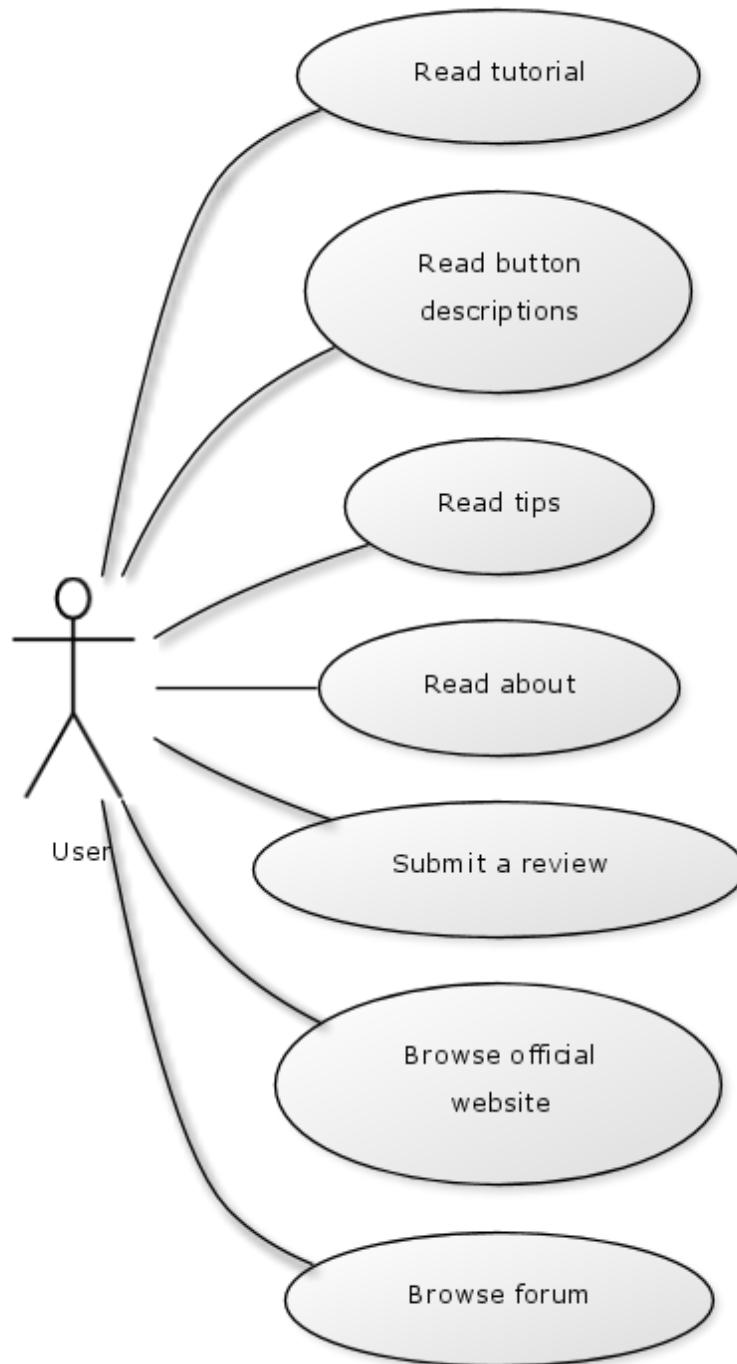
Help Screen



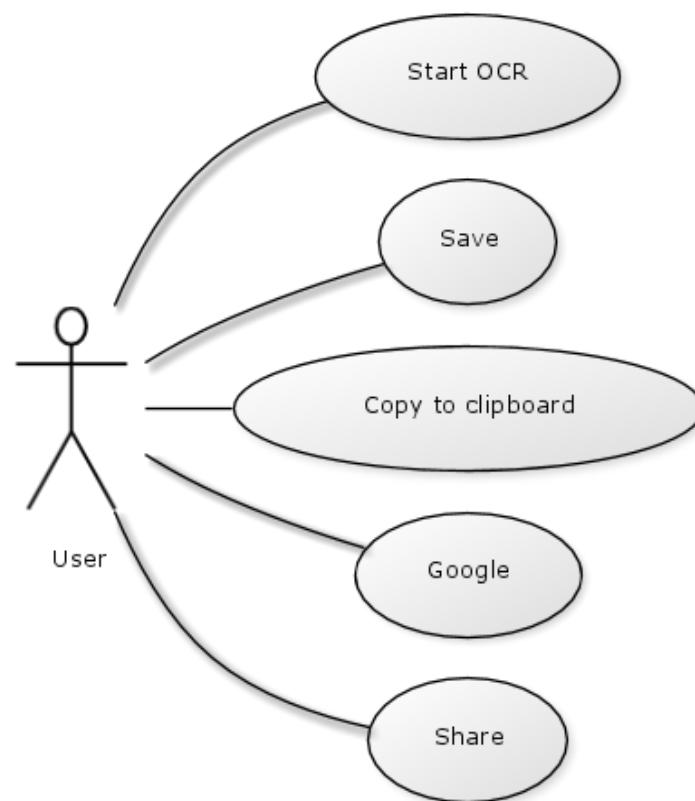
(Submit a review)

Use Case Diagrams

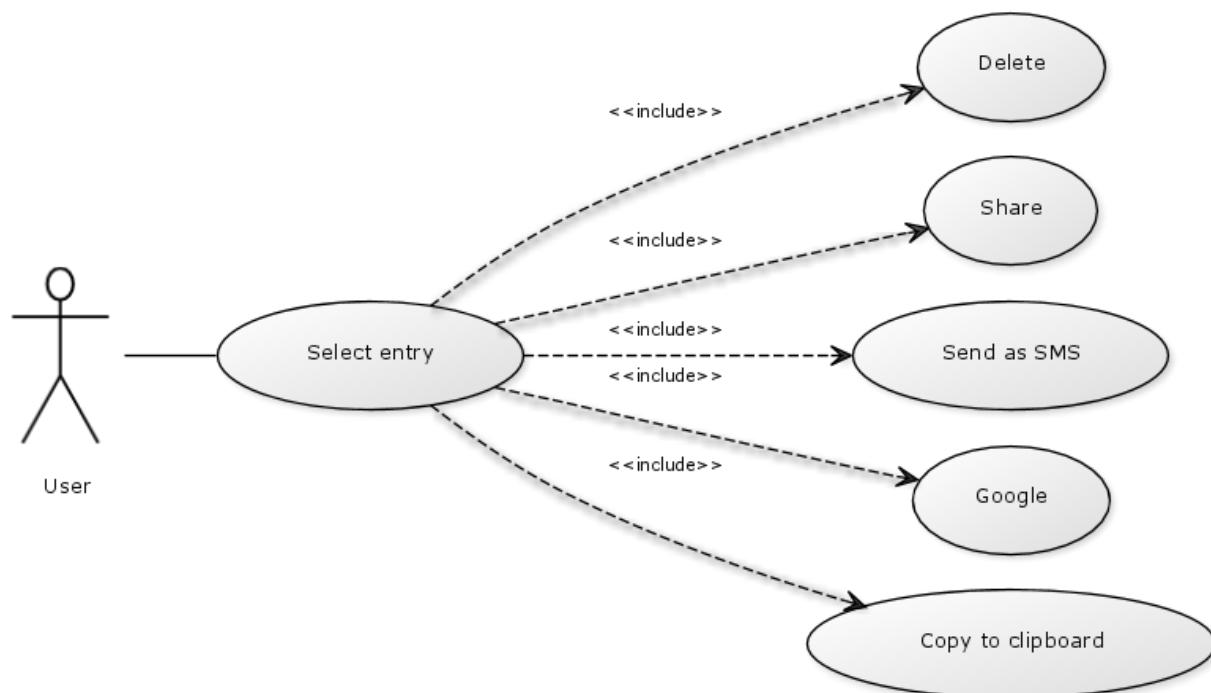




Help Screen

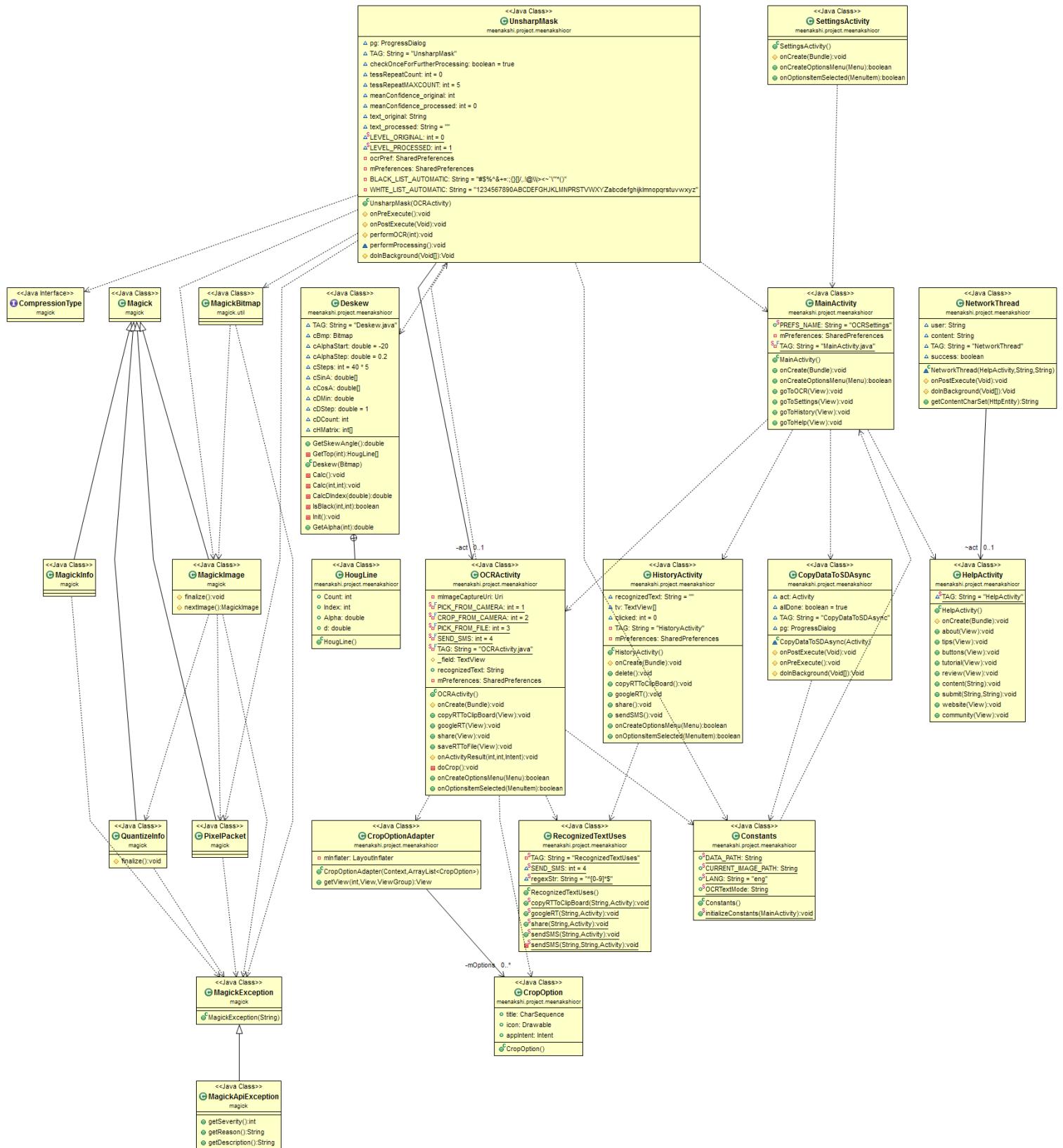


OCR Screen

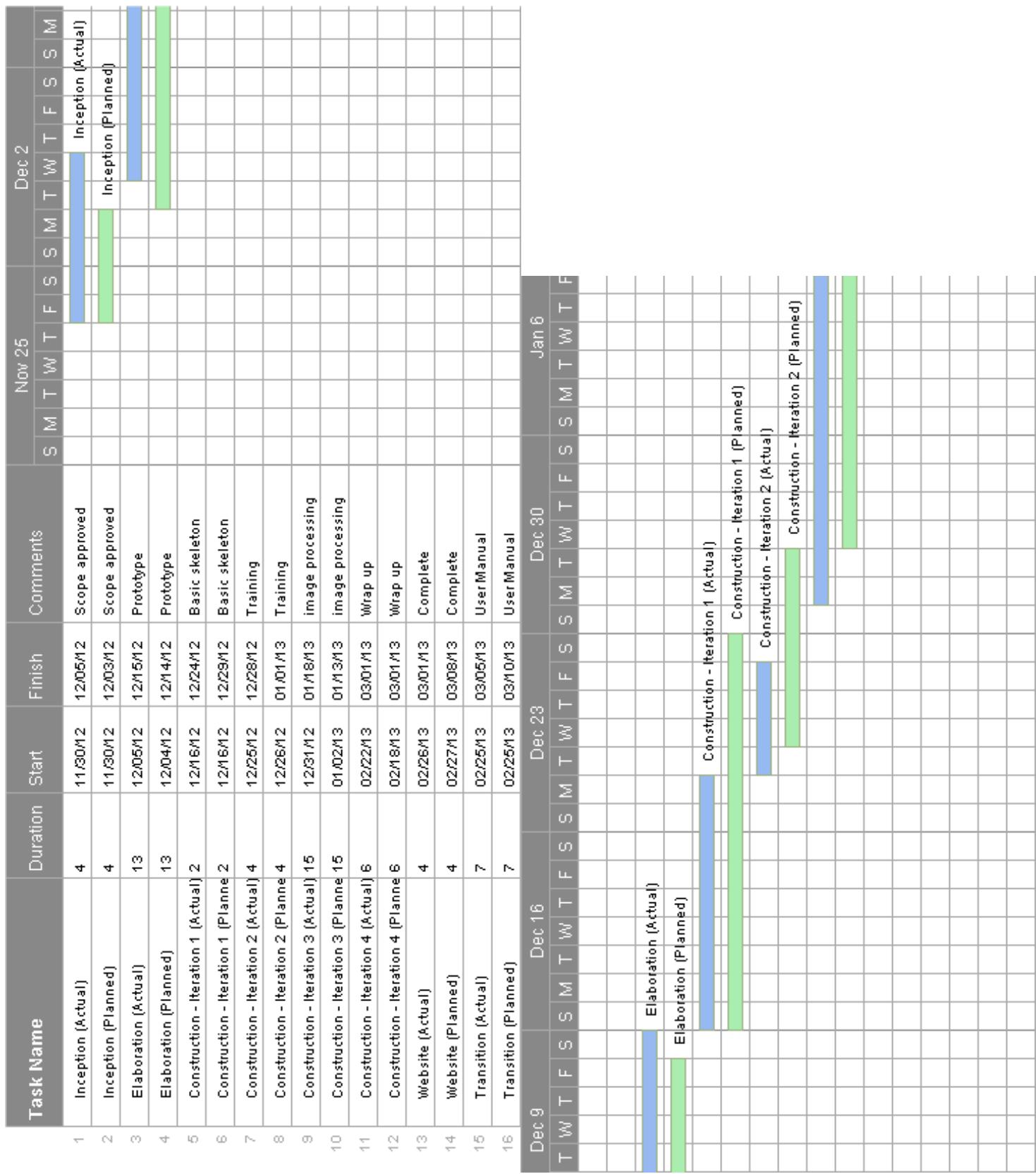


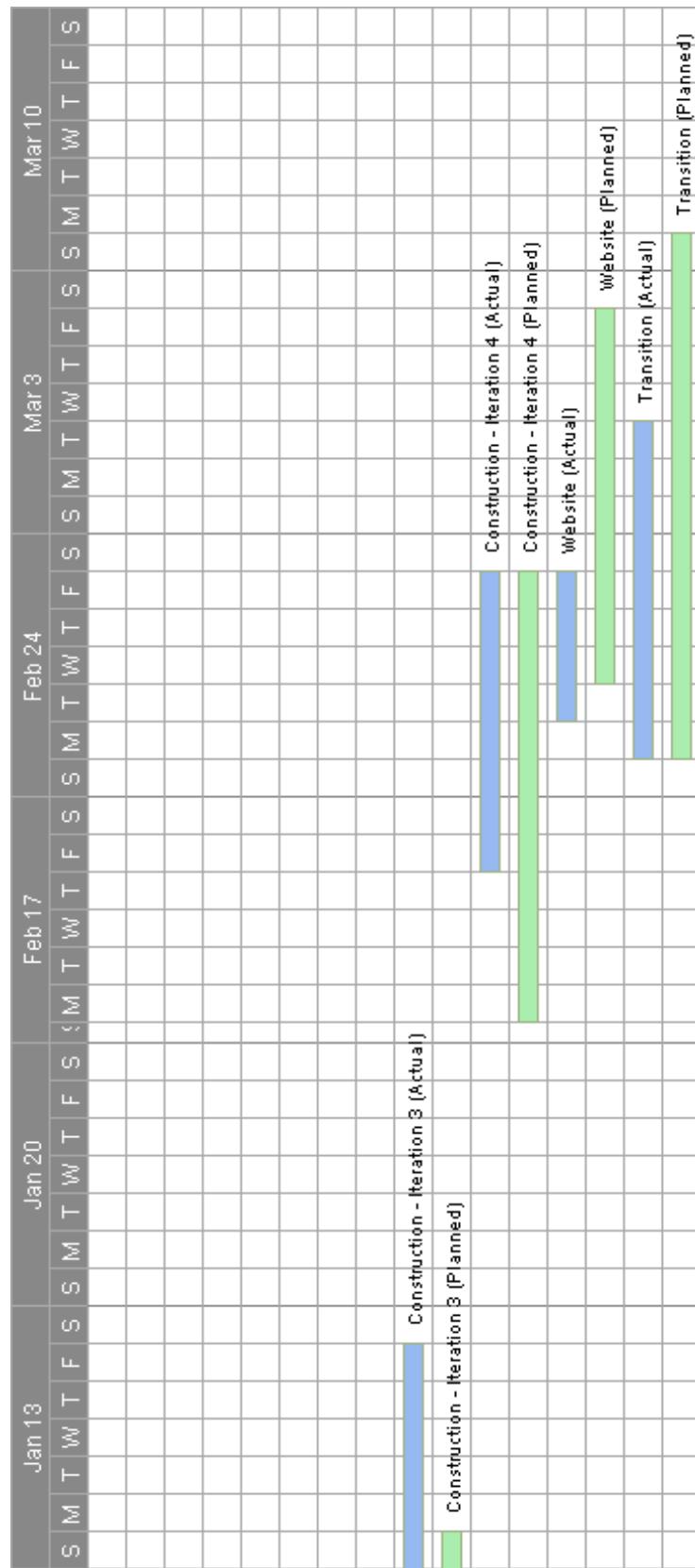
History Screen

Class Diagram



Gantt Chart





Construction

Tasks

- Coding and testing
- Experimentation and analysis

Objectives

- Finish software development

Deliverables

- Source code
- Product demonstration

Iterations/Spirals

This part consists of the increments produced over the 4 iterations to develop the product. 4 iterations were required as **new requirements and changes** were required to be added, and also because the product did not quite pass the **exit criteria** in the first 3 iterations.

In each iteration, a set of **objectives** are stated which define WHAT is to be achieved in the iteration. Following that, I **code and incorporate the changes** into the product. Within each iteration several small **increments/commits** were produced, but I will only show a **code snippet** of the final increment of each iteration in this book. **The code snippet contains disjointed statements separated by a horizontal separator, specifically only those statements that have changed in the Pre-processing and OCR module since the last iteration.**

At each stage, **at least 3 types of tests were performed:**

1. **Unit tests** of the newly incorporated changes to confirm that they work as intended
2. **Regression tests** to make sure that they are compatible with the rest of the system and no unintended side effects are produced
3. **System test against the exit criteria and requirements** defined in the Elaboration Phase to check if the product is ready

Basic skeleton of the app - Iteration 1

Objectives

- Add option to **select image from gallery** instead of accepting only new images from camera
- Add **experimental pre-processing** to see if it improves accuracy
- Design a basic **UI**
- Implement processing on a separate **thread**

Coding

What was coded:-

- ✓ Pre-processed the image before feeding it to Tesseract engine
 - increased/decreased DPI
 - made grey scale
 - applied Gaussian blur
 - applied unsharp mask filter
 - applied contrast

- o corrected orientation (portrait/landscape)
- ✓ Moved all pre-processing to AsyncTask to avoid application crashes
- ✓ Added option to select image from camera OR gallery
- ✓ Added optional crop feature

```

package meenakshi.project.meenakshiocr;

public class UnsharpMask extends AsyncTask<Void, Void, Void> {

    final static int KERNEL_WIDTH = 3;
    final static int KERNEL_HEIGHT = 3;

    int[][] kernel_blur = {
        {1, 1, 1},
        {1, 1, 1},
        {1, 1, 1}
    };

    final static int DIV_BY_9 = 9;

    Bitmap bitmap_Source;
    Bitmap afterProcess;
    String TAG = "UnsharpMask";

    private MainActivity mainActivity;

    public UnsharpMask(MainActivity act, Bitmap bitmap) {
        Log.v(TAG, "Begin constructor");
        mainActivity = act;

        bitmap_Source = bitmap;
    }

    private Bitmap processingBitmap(Bitmap src, int[][] knl){
        Bitmap dest = Bitmap.createBitmap(
            src.getWidth(), src.getHeight(), src.getConfig());

        int bmWidth = src.getWidth();
        int bmHeight = src.getHeight();
        int bmWidth_MINUS_2 = bmWidth - 2;
        int bmHeight_MINUS_2 = bmHeight - 2;
        int bmWidth_OFFSET_1 = 1;
        int bmHeight_OFFSET_1 = 1;

        for(int i = bmWidth_OFFSET_1; i <= bmWidth_MINUS_2; i++){
            for(int j = bmHeight_OFFSET_1; j <= bmHeight_MINUS_2; j++){

                //get the surround 7*7 pixel of current src[i][j] into a matrix subSrc[][]
                int[][] subSrc = new int[KERNEL_WIDTH][KERNEL_HEIGHT];
                for(int k = 0; k < KERNEL_WIDTH; k++){
                    for(int l = 0; l < KERNEL_HEIGHT; l++){
                        subSrc[k][l] = src.getPixel(i-bmWidth_OFFSET_1+k, j-bmHeight_OFFSET_1+l);
                    }
                }
                long subSumA = 0;
                long subSumR = 0;
                long subSumG = 0;
                long subSumB = 0;

                for(int k = 0; k < KERNEL_WIDTH; k++){
                    for(int l = 0; l < KERNEL_HEIGHT; l++){
                        subSumA += (long)(Color.alpha(subSrc[k][l])) * (long)(knl[k][l]);
                        subSumR += (long)(Color.red(subSrc[k][l])) * (long)(knl[k][l]);
                        subSumG += (long)(Color.green(subSrc[k][l])) * (long)(knl[k][l]);
                    }
                }
                dest.setPixel(i, j, Color.argb((int)subSumA/9, (int)subSumR/9, (int)subSumG/9, (int)subSumB/9));
            }
        }
        return dest;
    }
}

```

```

        subSumB += (long)(Color.blue(subSrc[k][l])) * (long)(knl[k][l]);
    }

    subSumA = subSumA/DIV_BY_9;
    subSumR = subSumR/DIV_BY_9;
    subSumG = subSumG/DIV_BY_9;
    subSumB = subSumB/DIV_BY_9;

    int orgColor = src.getPixel(i, j);
    int orgA = Color.alpha(orgColor);
    int orgR = Color.red(orgColor);
    int orgG = Color.green(orgColor);
    int orgB = Color.blue(orgColor);

    subSumA = orgA + (orgA - subSumA);
    subSumR = orgR + (orgR - subSumR);
    subSumG = orgG + (orgG - subSumG);
    subSumB = orgB + (orgB - subSumB);

    if(subSumA <0){
        subSumA = 0;
    }else if(subSumA > 255){
        subSumA = 255;
    }

    if(subSumR <0){
        subSumR = 0;
    }else if(subSumR > 255){
        subSumR = 255;
    }

    if(subSumG <0){
        subSumG = 0;
    }else if(subSumG > 255){
        subSumG = 255;
    }

    if(subSumB <0){
        subSumB = 0;
    }else if(subSumB > 255){
        subSumB = 255;
    }

    dest.setPixel(i, j, Color.argb(
        (int)subSumA,
        (int)subSumR,
        (int)subSumG,
        (int)subSumB));
}

return dest;
}

protected void performOCR()
{
    BitmapFactory.Options options = new BitmapFactory.Options();
    options.inSampleSize = 4;

    Bitmap bitmap = BitmapFactory.decodeFile(mainActivity._path, options);

    // Getting width & height of the given image.
    int w = bitmap.getWidth();
    int h = bitmap.getHeight();

    try {
        ExifInterface exif = new ExifInterface(mainActivity._path);
        int exifOrientation = exif.getAttributeInt(
            ExifInterface.TAG_ORIENTATION,
            ExifInterface.ORIENTATION_NORMAL);
    }
}

```

```

Log.v(TAG, "Orient: " + exifOrientation);

int rotate = 0;

switch (exifOrientation) {
    case ExifInterface.ORIENTATION_ROTATE_90:
        rotate = 90;
        break;
    case ExifInterface.ORIENTATION_ROTATE_180:
        rotate = 180;
        break;
    case ExifInterface.ORIENTATION_ROTATE_270:
        rotate = 270;
        break;
}

Log.v(TAG, "Rotation: " + rotate);

if (rotate != 0) {

    // Setting pre rotate
    Matrix mtx = new Matrix();
    mtx.preRotate(rotate);

    // Rotating Bitmap
    bitmap = Bitmap.createBitmap(bitmap, 0, 0, w, h, mtx, false);
}

// Convert to ARGB_8888, required by tess
bitmap = bitmap.copy(Bitmap.Config.ARGB_8888, true);

} catch (IOException e) {
    Log.e(TAG, "Couldn't correct orientation: " + e.toString());
}

Log.v(TAG, "Before baseApi");

TessBaseAPI baseApi = new TessBaseAPI();
baseApi.setDebug(true);
baseApi.init(mainActivity.DATA_PATH, mainActivity.lang);
baseApi.setImage(bitmap);

mainActivity.recognizedText = baseApi.getUTF8Text();

baseApi.end();

Log.v(TAG, "OCRED TEXT: " + mainActivity.recognizedText);

if (mainActivity.lang.equalsIgnoreCase("eng")) {
    mainActivity.recognizedText = mainActivity.recognizedText.replaceAll("[^a-zA-Z0-9]+", " ");
}

mainActivity.recognizedText = mainActivity.recognizedText.trim();
}

@Override
protected void onPostExecute(Void result) {
    Log.v("AsyncTask Mein", "Entered onPostExecute");

    if (mainActivity.recognizedText.length() != 0) {
        mainActivity._field.setText(mainActivity.recognizedText);
    }

    mainActivity.progressBar.setVisibility(View.GONE);
}

@Override
protected Void doInBackground(Void... params) {

```

```
Log.v(TAG, "In runnable thread, before processing");
int w = bitmap_Source.getWidth(), h = bitmap_Source.getHeight();
if(w<300 && h<300)
    afterProcess = OCRIImageProcessing.increaseDPI(bitmap_Source,w,h);
else if(w>1000 || h>1000)
    afterProcess = OCRIImageProcessing.decreaseDPI(bitmap_Source,w,h);
else
    afterProcess=bitmap_Source;
afterProcess = OCRIImageProcessing.makeGreyScale(afterProcess);

afterProcess = OCRIImageProcessing.applyGaussianBlur(afterProcess);
afterProcess = OCRIImageProcessing.applyGaussianBlur(afterProcess);
afterProcess = processingBitmap(afterProcess, kernel_blur);
afterProcess = processingBitmap(afterProcess, kernel_blur);
afterProcess = processingBitmap(afterProcess, kernel_blur);

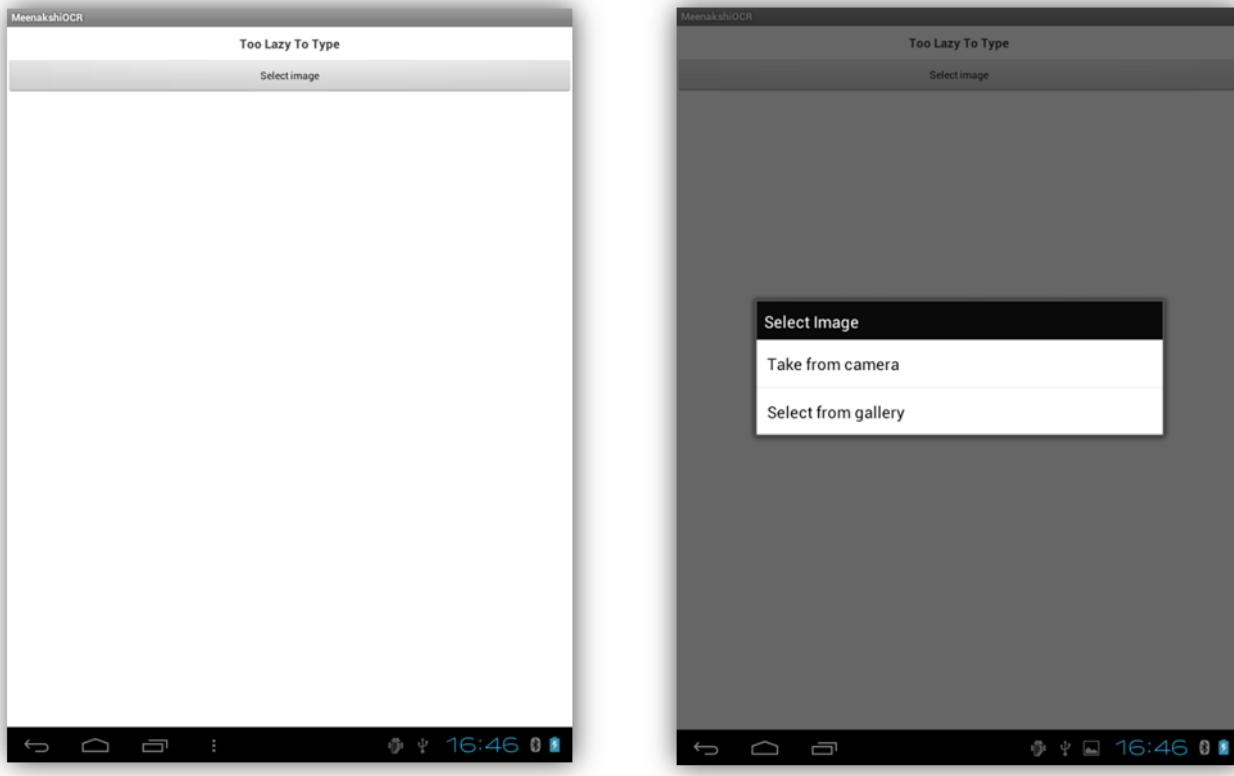
afterProcess = OCRIImageProcessing.applyGaussianBlur(afterProcess);
Log.v(TAG, "In runnable thread, after processing");

try{
    FileOutputStream out = new FileOutputStream(mainActivity._path);
    afterProcess.compress(Bitmap.CompressFormat.JPEG, 100, out);
}catch(Exception e)
{
    Log.v(TAG, e.toString());
}

performOCR();
Log.v("AsyncTask", "End of do In Background");

return null;
}
```

Tests





Results

- Image processing is too slow
- Accuracy is too low
- UI needs improvements
- AsyncTask works beautifully
- All other improvements work correctly

Ceiling Analysis Summary

Component	Accuracy
Overall System	34.615%
Pre-processing	76.47%
OCR engine	100%

Pre-processing and Training OCR for better accuracy – Iteration 2

Objectives

- Use **Leptonica's** built-in **image processing** functions to improve accuracy
- Use image processing techniques such as thresholding, etc.
- **Train the Tesseract engine** to improve accuracy
- Design a better **UI**
- Add functionality to use the text such as **copy to clipboard, save as text file and googling** it.

Coding

Changes since last increment:-

- ✓ Trained Tesseract engine with 30 fonts
- ✓ Added new image processing filters
- ✓ Created a separate OCR screen
- ✓ Customized UI v1

- ✓ Added several new utility classes
- ✓ Incorporated copy, save, and Google text feature

```
/**This class handles the OCR processing and unsharp-masking for processing the image
 * author: Meenakshi Madan
 */

import java.io.File;

import com.googlecode.leptonica.android.AdaptiveMap;
import com.googlecode.leptonica.android.Binarize;
import com.googlecode.leptonica.android.Convert;
import com.googlecode.leptonica.android.Enhance;
import com.googlecode.leptonica.android.Pix;
import com.googlecode.leptonica.android.ReadFile;
import com.googlecode.leptonica.android.Rotate;
import com.googlecode.leptonica.android.Scale;
import com.googlecode.leptonica.android.Skew;
import com.googlecode.leptonica.android.WriteFile;
import com.googlecode.tesseract.android.TessBaseAPI;

public class UnsharpMask extends AsyncTask<Void, Integer, Void> {

    ProgressDialog pg;

    /** Tag for logging purposes */
    String TAG = "UnsharpMask";

    /** To check if OCR needs to be performed again on the same image - if the confidence value is very low */
    boolean checkOnceForFurtherProcessing = true;

    /** Number of times ocr has been performed in this transaction */
    int tessRepeatCount = 0;

    /** Maximum number of times that OCR can be performed on the image in this transaction */
    int tessRepeatMAXCOUNT = 5;

    /** Mean confidence as returned by tesseract on the recognized text */
    int meanConfidence;

    /** Object of OCRActivity, to access variables such as DATA_PATH and view elements */
    private OCRActivity act;

    public UnsharpMask(OCRActivity act, Bitmap bitmap) {
        Log.v(TAG, "Begin constructor");
        this.act = act;

        @Override
        protected void onPreExecute() {
            Log.v("CopData AsyncTask Mein", "Entered onPreExecute");

            pg = new ProgressDialog(act, 0);
            pg.setMessage("Processing...");
            pg.setMax(10);
            pg.setIndeterminate(false);
            pg.show();
        }

        @Override
        protected void onProgressUpdate(Integer... progress) {
            pg.setProgress(progress[0]);
        }
    }
}
```

```

/** Displays text to the user, hides progress bar
 *
 */
@Override
protected void onPostExecute(Void result) {
    Log.v("AsyncTask ", "Entered onPostExecute");

    act.mImageView.setImageBitmap(afterProcess);

    if (act.recognizedText.length() != 0) {
        act._field.setText(act.recognizedText);
        act._field.setVisibility(View.VISIBLE);
        pg.setMessage("Done!");
        pg.dismiss();
        ((Button)act.findViewById(R.id.btn_copyToClipboard)).setVisibility(View.VISIBLE);
        ((Button)act.findViewById(R.id.btn_googleIt)).setVisibility(View.VISIBLE);
        ((Button)act.findViewById(R.id.btn_saveToFile)).setVisibility(View.VISIBLE);
    }
    else
    {
        pg.setMessage("Oops, no text found!");
        pg.setCanceledOnTouchOutside(true);
    }
}

/**
 * Function that performs unsharp masking on the Bitmap object
 * @param src - Bitmap object, the bitmap image to perform the unsharp maskin on
 * @param knl - kernal 2D array
 * @return processed Bitmap
 */

```

```

/** Performs OCR on the bitmap at _path on SDCARD. Also performs any orientation required
 *
 */

```

```

protected void performOCR()
{
    BitmapFactory.Options options = new BitmapFactory.Options();
    options.inSampleSize = 1;

    Bitmap bitmap = BitmapFactory.decodeFile(Constants.CURRENT_IMAGE_PATH, options);

    ExifInterface exif = new ExifInterface(Constants.CURRENT_IMAGE_PATH);
    baseApi.setVariable(TessBaseAPI.VAR_CHAR_BLACKLIST, "#$%^&+=;{}[],!@\\|><~`\"*\\0");
    baseApi.setVariable(TessBaseAPI.VAR_CHAR_WHITELIST,
                        "1234567890ABCDEFGHIJKLMNPRSTVWXYZabedefghijklmnopqrstuvwxyz");

    Log.v(TAG, "After setting variables");
    baseApi.init(Constants.DATA_PATH, Constants.LANG); //, TessBaseAPI.OEM_CUBE_ONLY
    Log.v(TAG, "After init and before setting bitmap");

    Log.v(TAG, "After init and before getUTF8Text");
    act.recognizedText = baseApi.getUTF8Text();
    meanConfidence = baseApi.meanConfidence();
    Log.v(TAG, "OCRED TEXT: " + act.recognizedText);
    Log.v(TAG, "Mean Confidence: " + meanConfidence);
    if (baseApi != null) {
        baseApi.clear();
    }

    Log.v(TAG, "OCRED TEXT: " + act.recognizedText);
    if (Constants.LANG.equalsIgnoreCase("eng")) {

```

```

        act.recognizedText = act.recognizedText.replaceAll("[^a-zA-Z0-9,-;:\"()@$><?!]+", " ");
    }

    act.recognizedText = act.recognizedText.trim();

    afterProcess = bitmap;

}

/***
 * Calls functions to perform required preprocessing and OCR
 */
afterProcess = bitmap_Source;
publishProgress(1);

publishProgress(4);

Log.v(TAG, "After unsharp");

FileOutputStream out = new FileOutputStream(Constants.CURRENT_IMAGE_PATH);

Log.v(TAG, "After saving file to sdcard");

File pic = new File(Constants.CURRENT_IMAGE_PATH);
Pix pix = ReadFile.readFile(pic);
pix = AdaptiveMap.backgroundNormMorph(pic, 16, 3, 200);
pix = Enhance.unsharpMasking(pic, 3, 0.7F);
if(pic.getWidth() < 300 || pix.getHeight() < 300) pix = Scale.scale(pic, 2);
else if(pic.getWidth() > 1200 || pix.getHeight() > 1200) pix = Scale.scale(pic, 1/2);
pix = Convert.convertTo8(pic);

pix = Rotate.rotate(pic, -Skew.findSkew(pic));
pix = Binarize.otsuAdaptiveThreshold(pic);

Log.v(TAG, "After scale and binarize");

//pix = Enhance.unsharpMasking(pic, 3, 0.7F); //gives OutOfMemoryError
WriteFile.writeFileImpliedFormat(pic, pic, 100, true);
afterProcess = WriteFile.writeBitmap(pic);

Log.v(TAG, "In runnable thread, after processing");

publishProgress(8);

publishProgress(10);

Log.v("AsyncTask", "End of do In Background");
pix.recycle();

/** Check if a given String contains any of the characters in the given array
 *
 * @param str source string to check for characters
 * @param searchChars sequence of characters to check for in source string
 * @return boolean value - true if string contains any character, false otherwise
 */
public static boolean containsAny(String str, char[] searchChars) {
    if (str == null || str.length() == 0 || searchChars == null || searchChars.length == 0) {
        return false;
    }
    for (int i = 0; i < str.length(); i++) {
        char ch = str.charAt(i);
        for (int j = 0; j < searchChars.length; j++) {
            if (searchChars[j] == ch) {
                return true;
            }
        }
    }
}

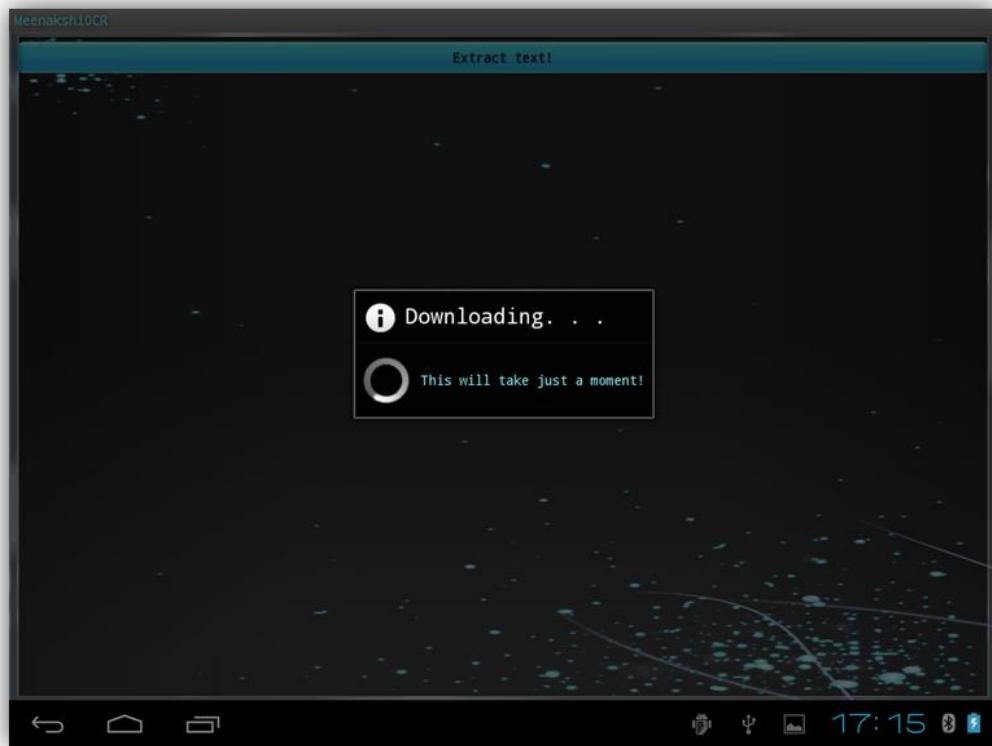
```

```
        }
    }
}
return false;
}

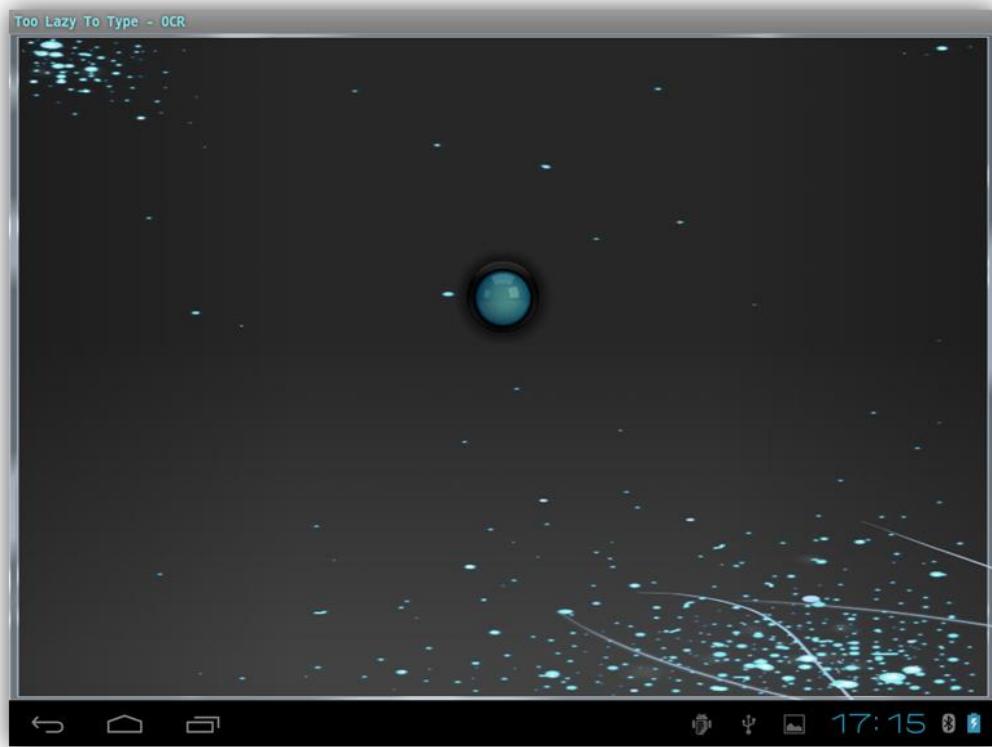
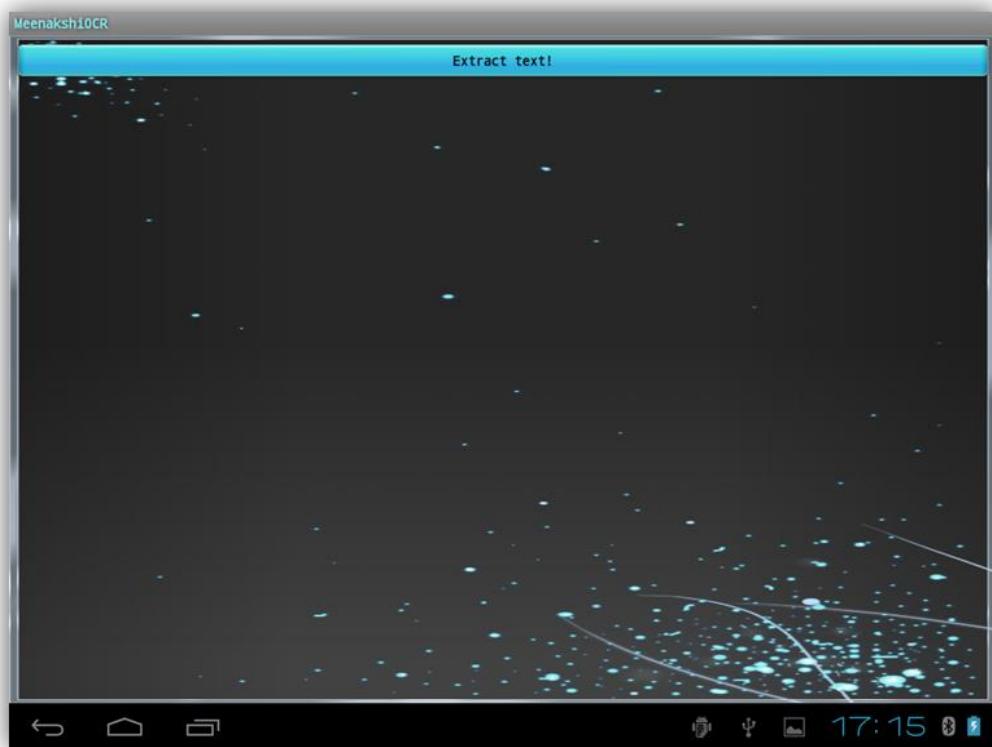
/** Computes whether the string contains any character from the given sequence of characters
 *
 * @param str string to search
 * @param searchChars sequence of characters to look for
 * @return boolean value, true if string contains any characters from sequence, false otherwise
 */

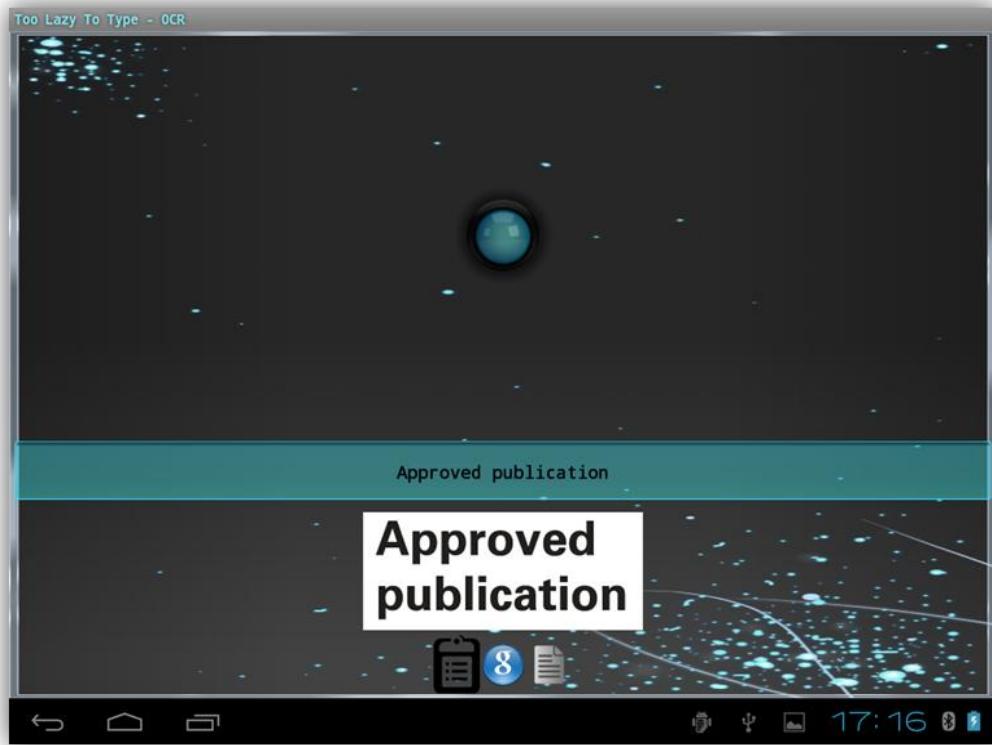
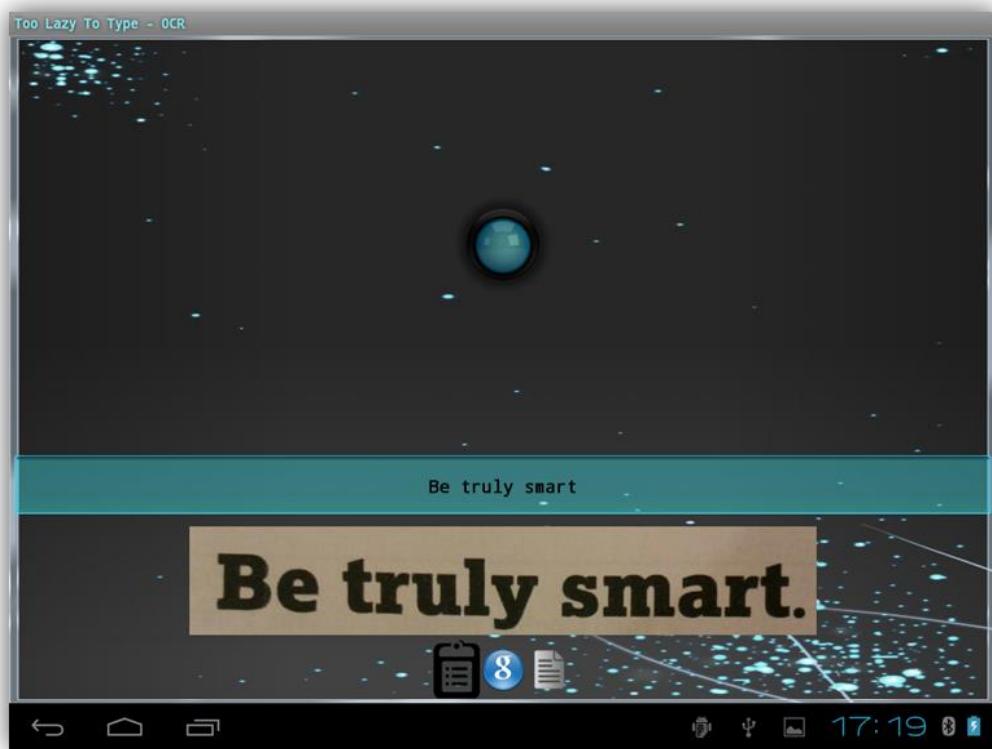
public static boolean containsAny(String str, String searchChars) {
    if (searchChars == null) {
        return false;
    }
    return containsAny(str, searchChars.toCharArray());
}
```

Tests



TOO LAZY TO TYPE





Results

- Processing speed has improved drastically
- Leptonica's built-in functions reduce the image quality drastically and hence cannot be used. Another image library should be used instead.
- Training Tesseract improved the accuracy quite drastically
- UI seems very disconnected
- Accuracy is quite satisfactory
- All other improvements work correctly
- Application crashes with a lot of images

Ceiling Analysis Summary

Component	Accuracy
Overall System	97.0588%
Pre-processing	97.0588%
OCR engine	100%

Advanced image processing and GUI – Iteration 3

Objectives

- Use Image Magick for **pre-processing**
- Use advanced processing techniques such as text dewarp
- Add **Settings and Preferences** activity
- Design an attractive **GUI**

Coding

Changes since last iteration:-

- ✓ Trained Tesseract just a bit more
- ✓ Added image-magick image processing library
- ✓ Implemented complex techniques such as text dewarp
- ✓ Created a Settings Activity for user's preferences
- ✓ Designed custom UI v2

```
/** Mean confidence as returned by tesseract on the recognized text */
int meanConfidence_original, meanConfidence_processed=0;
String text_original, text_processed="";

static int LEVEL_ORIGINAL = 0, LEVEL_PROCESSED=1;

private SharedPreferences ocrPref;

private String BLACK_LIST_AUTOMATIC = "#$%^&+=;{}[]/.!@\\|><~`\"*\\";
private String WHITE_LIST_AUTOMATIC = "1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
```

```
if(meanConfidence_original > meanConfidence_processed)
{
    act.recognizedText = text_original;
}
else {
    act.recognizedText = text_processed;
}

((TableRow)act.findViewById(R.id.tableRow3)).setVisibility(View.VISIBLE);
    ((TableRow)act.findViewById(R.id.tableRow4)).setVisibility(View.VISIBLE);
    (ImageView)act.findViewById(R.id.drobotarms).setImageResource(R.drawable.ocrscreen14);
}
else
{
    Toast.makeText(act, "Oops, no text found!", Toast.LENGTH_SHORT).show();
}
```

```
protected void performOCR(int level)

if(ocrPref.getString("whitelist", "None").equals("None"))
{
    baseApi.setVariable(TessBaseAPI.VAR_CHAR_BLACKLIST, BLACK_LIST_AUTOMATIC);
    baseApi.setVariable(TessBaseAPI.VAR_CHAR_WHITELIST, WHITE_LIST_AUTOMATIC);

    Log.v(TAG, "whitelist preferences returned None");
}
else
{
    baseApi.setVariable(TessBaseAPI.VAR_CHAR_WHITELIST, ocrPref.getString("whitelist", "None"));

    Log.v(TAG, "whitelist preferences returned " + ocrPref.getString("whitelist", "None"));
}

if(!ocrPref.getString("psm", "None").equals("None"))
{
    baseApi.setPageSegMode(Integer.parseInt(ocrPref.getString("psm", "None")));
    Log.v(TAG, "PSM preferences returned " + ocrPref.getString("psm", "None"));
}

Log.v(TAG, "PSM preferences returned None");

if(level == UnsharpMask.LEVEL_ORIGINAL)
{
    text_original = baseApi.getUTF8Text();
    meanConfidence_original = baseApi.meanConfidence();

    Log.v(TAG, "OCRED TEXT: " + text_original);
    Log.v(TAG, "Mean Confidence: " + meanConfidence_original);
}
else if(level == UnsharpMask.LEVEL_PROCESSED)
{
    text_processed = baseApi.getUTF8Text();
    meanConfidence_processed = baseApi.meanConfidence();

    Log.v(TAG, "OCRED TEXT: " + text_processed);
    Log.v(TAG, "Mean Confidence: " + meanConfidence_processed);
}
```

```

void performProcessing()
{
    try{
        ImageInfo mi = new ImageInfo(Constants.CURRENT_IMAGE_PATH);
        MagickImage m = new MagickImage(mi);
        if(m.normalizeImage()) Log.v(TAG, "normalize conversion successful");
        else Log.v(TAG, "normalize conversion unsuccessful");

        Deskew d = new Deskew(MagickBitmap.ToBitmap(m));
        double skew = d.GetSkewAngle();
        Log.v(TAG, "After Deskew, skew = " + skew);

        m = m.rotateImage(-skew); // /57.295779513082320876798154814105
        m.setDepth(8);

        m = m.sharpenImage(10, 8);
        if(m.negateImage()) Log.v(TAG, "negate conversion successful");
        else Log.v(TAG, "negate conversion unsuccessful");
        PixelPacket pp = m.getBackgroundColor();
        int bg = pp.getBlue(), thresh;
        Log.v(TAG, "BG color return by getBackgroundColor is: " + bg);
        if (bg<32757) thresh = 60000;
        else thresh = 10000;
        if(m.thresholdImage(32757)) Log.v(TAG, "thresh conversion successful"); // /15000
        else Log.v(TAG, "thresh conversion unsuccessful");
        if(m.negateImage()) Log.v(TAG, "negate conversion successful");
        else Log.v(TAG, "negate conversion unsuccessful");

        m = m.scaleImage(m.getWidth() + 100, m.getHeight() + 100);
        mi.setDensity("300");
        m.setCompression(CompressionType.NoCompression);
        m.setFileName(Constants.CURRENT_IMAGE_PATH); // give new location
        if(m.writeImage(mi)) Log.v(TAG, "Successfully wrote image to path"); // save
        else Log.v(TAG, "Image save unsuccessful");
    }
    catch(Exception e)
    {
        Log.v(TAG, "exception occurred performing magick functions: " + e.toString());
    }
}

/**
 * Calls functions to perform required preprocessing and OCR
 */
@Override
protected Void doInBackground(Void... params) {
    Log.v(TAG, "In runnable thread, before processing");
    performOCR(UnsharpMask.LEVEL_ORIGINAL);
    publishProgress(1);

    if(ocrPref.getBoolean("processimage", true)){
        performProcessing();
        performOCR(UnsharpMask.LEVEL_PROCESSED);

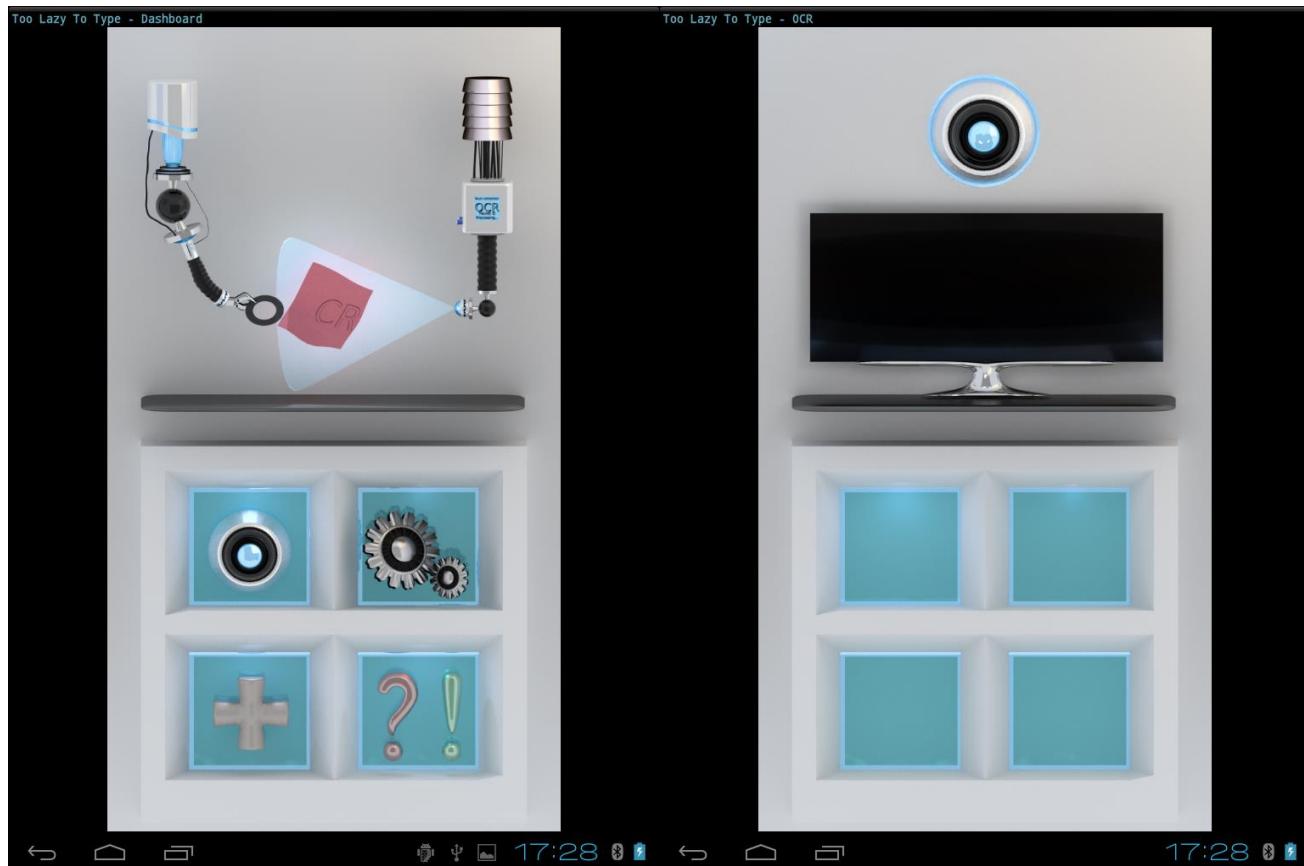
        Log.v(TAG, "Processimage preferences returned true");
    }

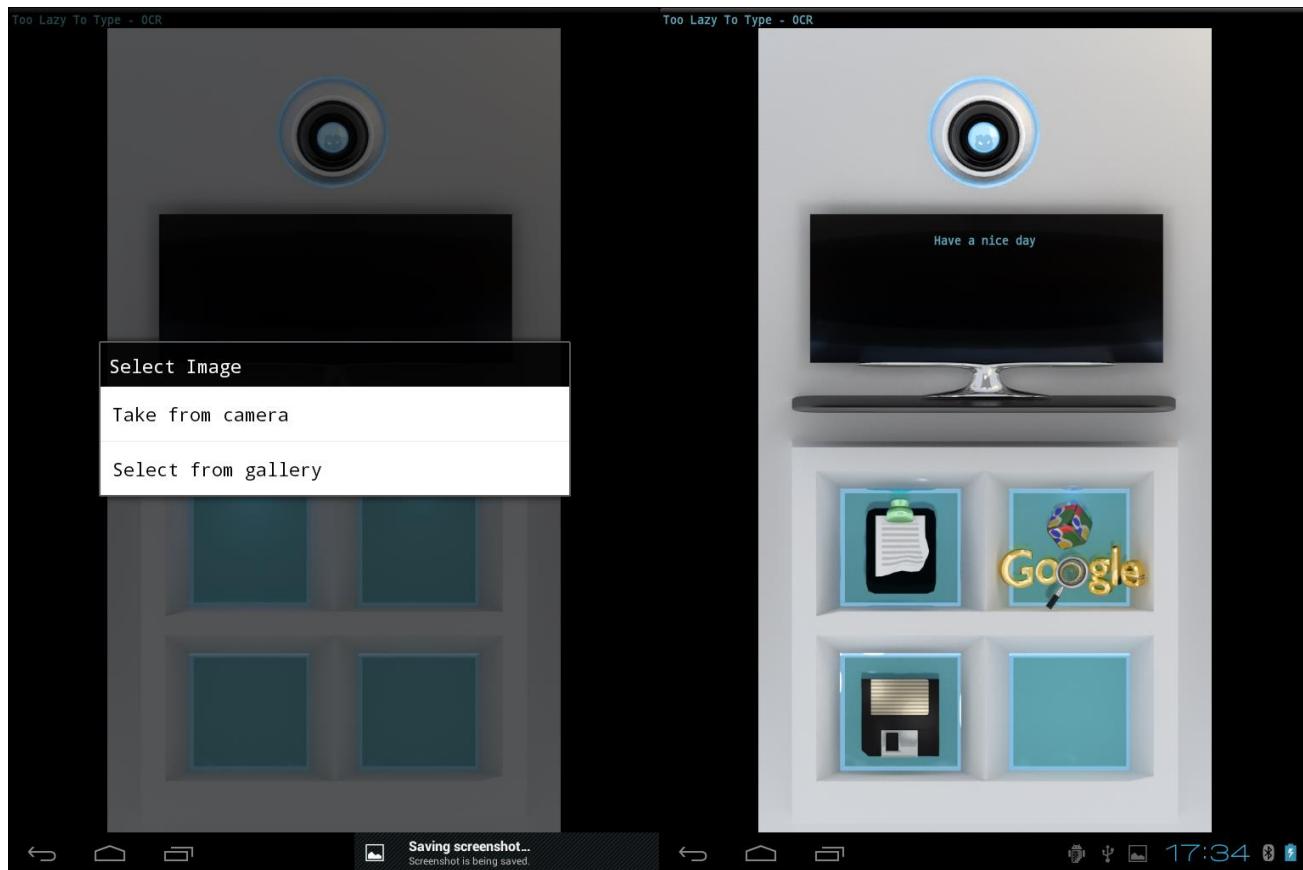
    Log.v(TAG, "Processimage preferences returned false");

    publishProgress(10);
    Log.v("AsyncTask Mein", "End of do In Background");
    return null;
}

```

Tests





Results

- New image processing library works beautifully, although slightly slower
- UI and layouts work well
- Accuracy quite good
- All other improvements work correctly
- Application doesn't crash like it did in the previous increments

Ceiling Analysis Summary

Component	Accuracy
Overall System	96%
Pre-processing	97.0588%
OCR engine	100%

Final touches and wrap up - Iteration 4

Objectives

- Add **Help Screen**
- Add “**recent history**” feature
- Add **SMS and “share”** feature
- **Optimize code**
- Add **welcome splash screen**
- **Optimize UI**
- Add an option for the user to **submit a review** to the website
- **Link the website and forum** from the app
- Final touches and winding up for transition to next phase and release

Coding

Changes since last increment:-

- ✓ Created Help screen and Help UI using WebView and html
- ✓ Designed a custom launcher icon set
- ✓ Designed a complete UI v2 set
- ✓ Created a new History screen for 5 most recent OCR text
- ✓ Optimized code
- ✓ Added a new SMS feature
- ✓ Added a new share feature
- ✓ Added a new welcome splash screen for first time installation

No code snippet is shown for this iteration as this is the final coding iteration and the entire code is displayed in a later section

Tests



Results

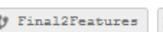
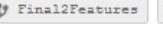
- All improvements work correctly
- Accuracy quite good
- Ready for transition to Deployment Phase

Ceiling Analysis Summary

Component	Accuracy
Overall System	94.736%
Pre-processing	97.0588%
OCR engine	100%

Iteration Summary

Author	Commit	Message	Date
Meenakshi Ma...	fde3be2	add more diagrams and cleanup	6 days ago
Meenakshi Ma...	102489e	add uml diagrams	11 days ago
Meenakshi Ma...	5c39b29	fix links in webview	12 days ago
Meenakshi Ma...	83bfff26	correct help screen typos	12 days ago
Meenakshi Ma...	90f9b96	add link to forum	15 days ago
Meenakshi Ma...	16febab	Add link to website and "Submit Review" feature	15 days ago
Meenakshi Ma...	9bcfe6c	Only show pop-up if entry in "Recent History" screen is not empty,	18 days ago
Meenakshi Ma...	a9113ba	Remove 2 unused files to reduce apk size below the 25MB limit	19 days ago
Meenakshi Ma...	f246bfc	Add help screen, custom launcher icon set, welcome splash screen, an...	19 days ago
Meenakshi Ma...	6ad83ec	Add hint text under icons	19 days ago
Meenakshi Ma...	ab169e1	Add "History" feature	20 days ago
Meenakshi Ma...	65cd287	Add share feature	20 days ago
Meenakshi Ma...	2bca0e1	add sms feature	20 days ago
Meenakshi Ma...	2818ef3	Tried to add a feature to "remove" the text from the image. Didn't work	20 days ago
Meenakshi Ma...	7a68355	Fix android-magick library UnsatisfiedLinkError	2013-01-20
Meenakshi Ma...	bce9e12	Fix progressdialog bug by removing it entirely	2013-01-18
Meenakshi Ma...	0f9c460	Newest UI - Background with transparent buttons	2013-01-17
Meenakshi Ma...	a626361	Brand new UI - Dashboard and OCR Screen completed	2013-01-15
Meenakshi Ma...	91fec07	Add new UI design template	2013-01-13
Meenakshi Ma...	08bf6cf	Add preferences activity	2013-01-10
Meenakshi Ma...	0f9fff4	Pre-process to sharp black/white text. Not perfect acc...	2013-01-03
Meenakshi Ma...	bae3b27	Add diagrams and a few pre-processing changes	2012-12-31
Meenakshi Ma...	b05a232	Add ImageMagick library for better pre-processing	2012-12-31
Meenakshi Ma...	316eb41	Removed all non-leptonica pro	2012-12-30
Meenakshi Ma...	5da7315	Added buttons for googling, saving to clip bo...	2012-12-29
Meenakshi Ma...	fc1f2c4	Make-over, new buttons. UI design v1	2012-12-28
Meenakshi Ma...	b261ece	re-trained 30 fonts to add .;`?@!	2012-12-27
Meenakshi Ma...	75aa033	trained 30 fonts	2012-12-26
Meenakshi Ma...	28a76dc	added training for franklin gothic and gulim ch...	2012-12-26
Meenakshi Ma...	30cf7d4	Added shared preferences to make sure files	2012-12-26

Author	Commit	Message			Date
 Meenakshi M...	e98b677	It's giving a stupid runtime exception, but I'm cr...	  		2012-12-26
 Meenakshi M...	5726a30	All stable, very good accuracy, trained data for...	  		2012-12-25
 Meenakshi M...	32ae4c8	added new training data, still not as accurate.	  		2012-12-25
 Meenakshi M...	58e3857	crashfix v2 - works okay, accu...	   		2012-12-24
 Meenakshi M...	8281892	I dunno what this is but it's making me commit...	  		2012-12-23
 Meenakshi M...	1ca987a	works okay on some images, crashes on some images, crop intent doesn't...			2012-12-23
 Meenakshi M...	0cdd968	Executing image processing and ocr in AsyncTask successfully			2012-12-23
 Meenakshi M...	4a63c65	working fine, without extending thread/runnable			2012-12-23
 Meenakshi M...	a#04e78	preprocessing phase - added gaussian blur and increased dpi			2012-12-23
 Meenakshi M...	b21ea0a	gives very in-accurate ocr but gives some text nonetheless			2012-12-23
 Meenakshi M...	5307196	Created folders on sdcard successfully and final cropped pic is stored			2012-12-22
 Meenakshi M...	2500ef6	Basic selection from gallery/click from camera and crop. Works nicely.			2012-12-22

The Website

Objectives

- Create a dedicated website for the app
- Ensure easy download of the app
- Add brief descriptions and detailed user manuals for the convenience of users
- Add a forum for discussions on the app
- Add a way for users to review the app, and for new users to see those reviews on the website

Coding

What has been coded:-

- ✓ Main page which includes
 - Download link
 - Brief description
 - Slideshow of screenshots
 - News and updates
 - Top 3 reviews

- Quick links such as download, user manual, FAQ, community forum, Facebook page, etc.
- A Facebook plugin so users can “Like” the app and recommend it to their friends
- ✓ Reviews page which shows ALL the user reviews for this app
- ✓ Forum website for interaction among users, discussions about the app, questions and queries, etc.
- ✓ FAQ post to answer the frequently asked questions
- ✓ Facebook page which users can “Like” to keep tabs on what’s going on with the app and updates
- ✓ Developer Tools for fellow developers or students looking to learn

Screenshots

The screenshot shows the homepage of the **Too Lazy To Type** website. At the top, there's a navigation bar with **Home**, **Download**, and **Forum** buttons. Social sharing icons for **Like** and **Send** are also present. The main content area features a large banner with the text "Welcome!" and a robotic arm holding a red card with "CR" on it. To the left, there's a download section for "MeenakshiOCR v1" with a brief description of its features. Below the banner, there are three screenshots of the app interface showing various screens. On the right, there's an "Application Description" section with a bulleted list of features and a text block explaining the app's purpose. Below that, there are links for "Download (.apk)" and "Visit the links below for more information". At the bottom, there are three columns: "Quick Links", "What's New", and "Reviews", each containing a list of links.

Quick Links

- ★ [Download \(.apk\)](#)
- ★ [User Manual \(.pdf\)](#)
- ★ [FAQ](#)
- ★ [Community](#)
- ★ [Reviews](#)
- ★ [Visit us on Facebook](#)
- ★ [Developer Tools](#)

What's New

- ★ New Facebook page is up. You can now officially "like" this app and keep tabs on updates
- ★ Forum is now online! Participate and be a part of the community
- ★ See what other users have to say about this app on Reviews

Reviews

- ★ [Payal](#) : fantabulous app :-*
- ★ [smruti](#): like it
- ★ [Payal](#) : love the ui

facebook  Search for people, places and things 

You are posting, commenting, and liking as Too Lazy To Type — Change to Meenakshi Madan

 Too Lazy To Type Timeline Now 



Too Lazy To Type  Liked 

19 likes · 2 talking about this

App Page
High accuracy Android OCR app with features such as save, copy to clipboard, share, google, sms, recent history, etc.

About 


Write something...

Too Lazy To Type updated their cover photo.  February 28 

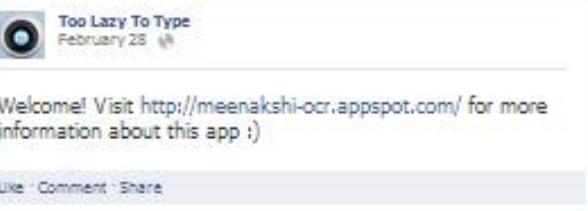
Like · Comment · Share

 Too Lazy To Type February 28 

App Screenshots (23 photos) 

17 Friends Like Too Lazy To Type  +8


Too Lazy To Type joined Facebook. 

 Too Lazy To Type February 28 

Welcome! Visit <http://meenakshi-ocr.appspot.com/> for more information about this app :) 

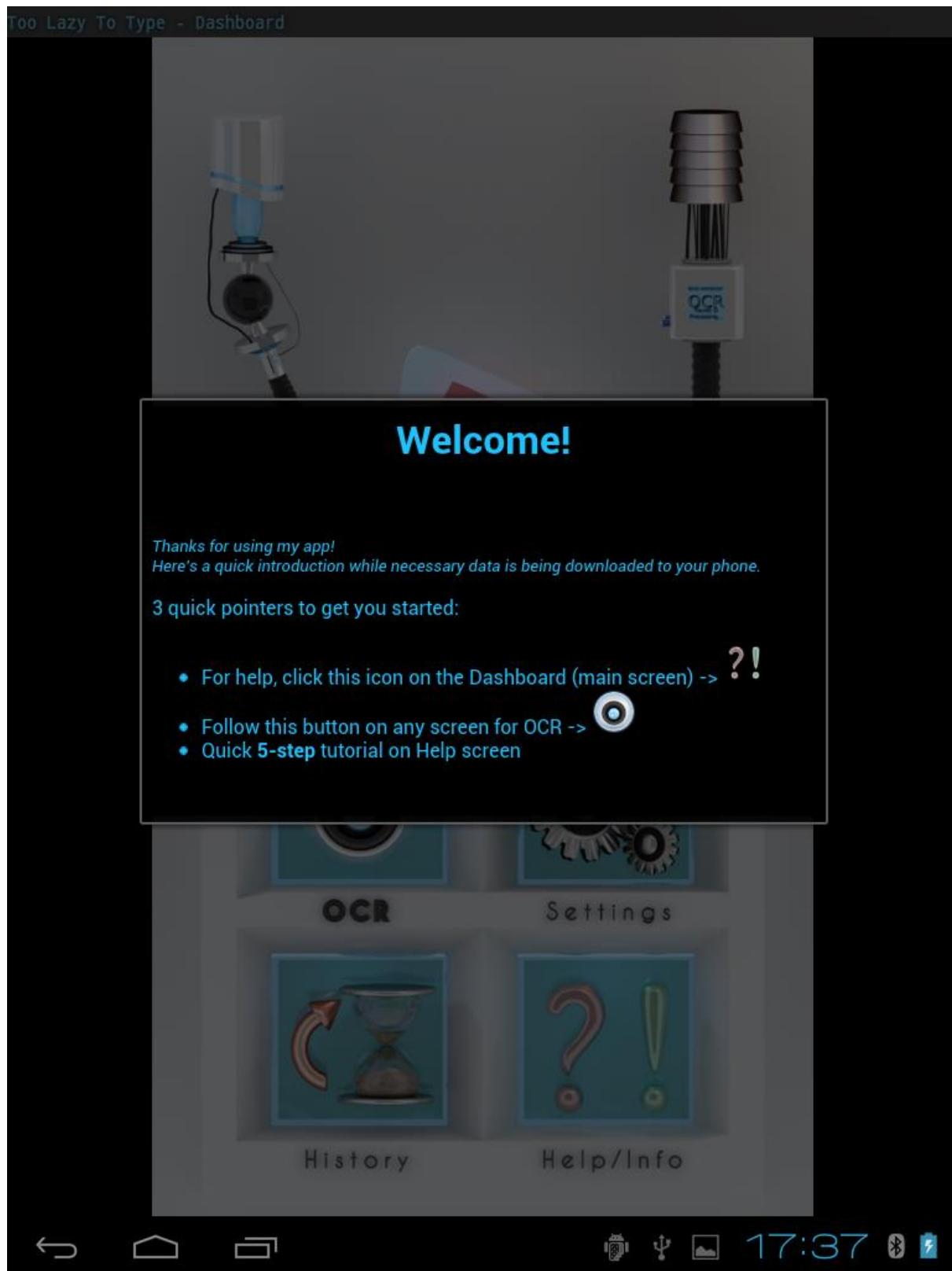
Like · Comment · Share

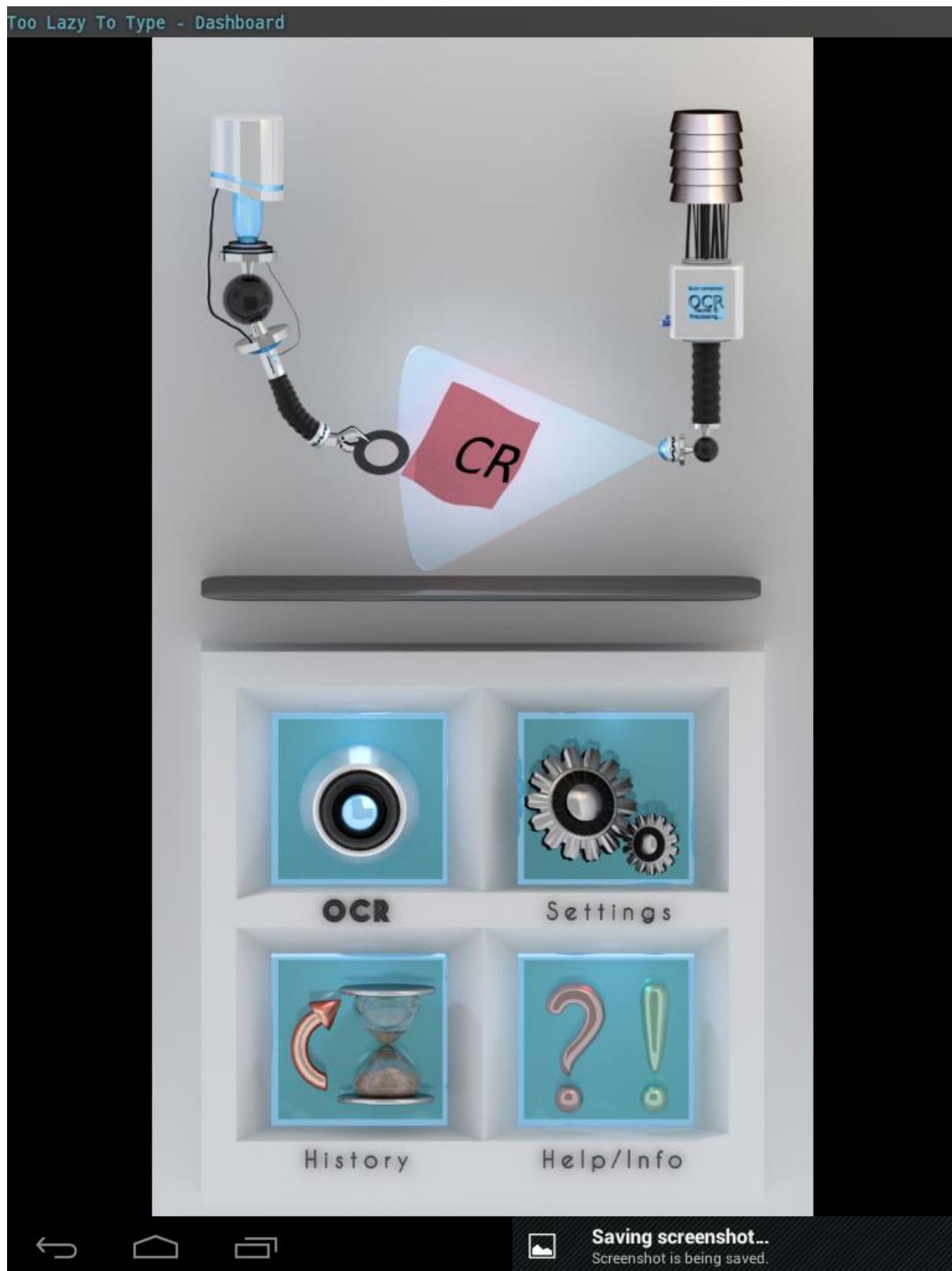
Graphical User Interface

The following are a few screenshots of the application at work.

(in order of appearance)

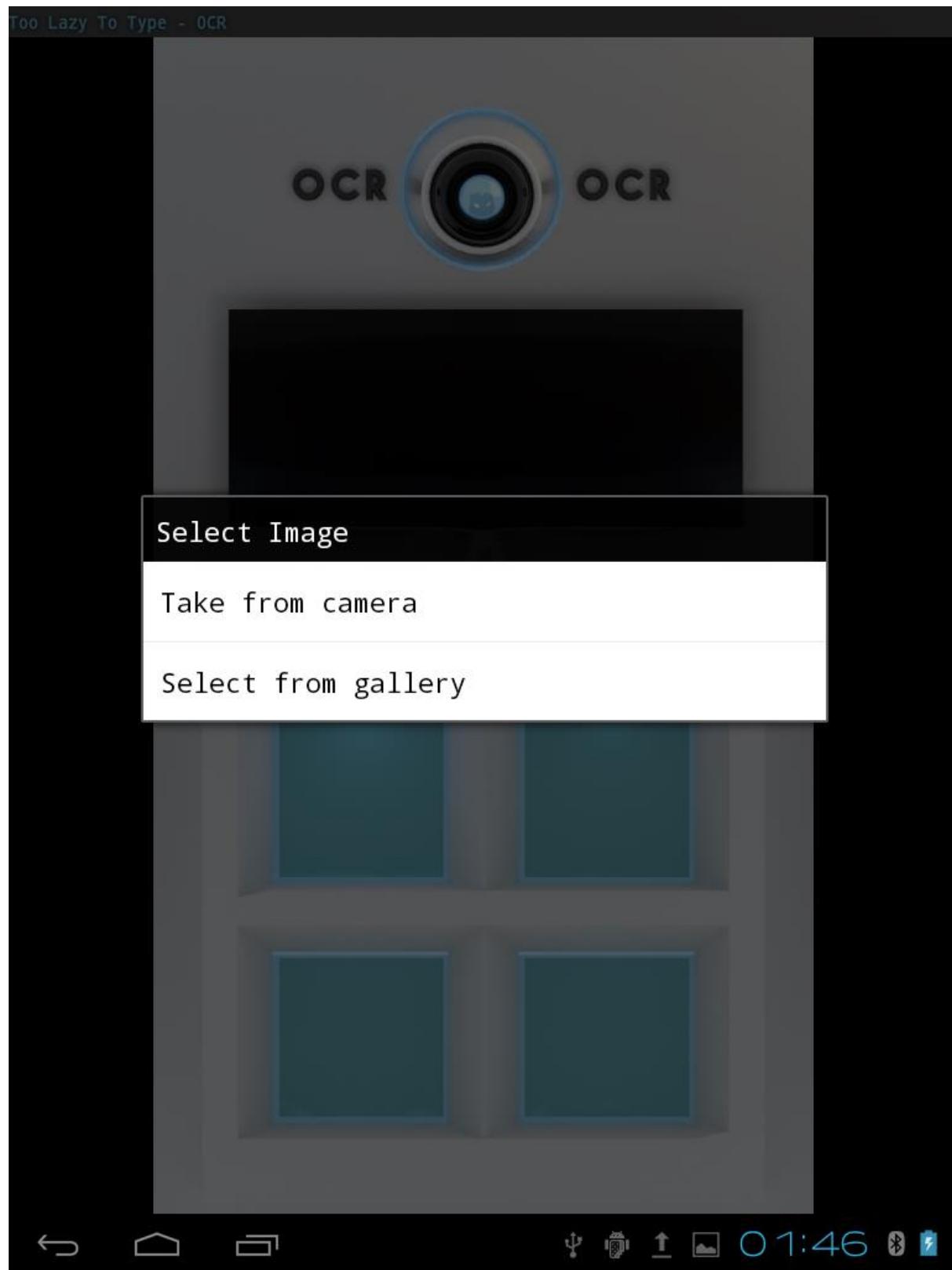
- ✓ Welcome screen (only comes up on first use)
- ✓ Main Screen / Dashboard
- ✓ OCR Screen
- ✓ Chooser that pops-up after pressing the OCR button
- ✓ Crop Screen
- ✓ Progress dialog while the image is being processed
- ✓ The OCR'ed text is displayed
- ✓ Share feature in action (3 screenshots)
- ✓ Recent History Screen – the latest text is reflected immediately
- ✓ Chooser that pops-up upon clicking an entry on the History Screen
- ✓ Direct SMS feature (Messenger app doesn't open, the message goes directly and notifies the user of the state)
- ✓ Settings/Preference Screen
- ✓ Help/Info Screen
- ✓ About section on Help Screen
- ✓ Tips section on Help Screen
- ✓ Buttons section on Help screen (3 screenshots)
- ✓ 5-step tutorial on Help Screen (4 screenshots)
- ✓ Submit a review dialog

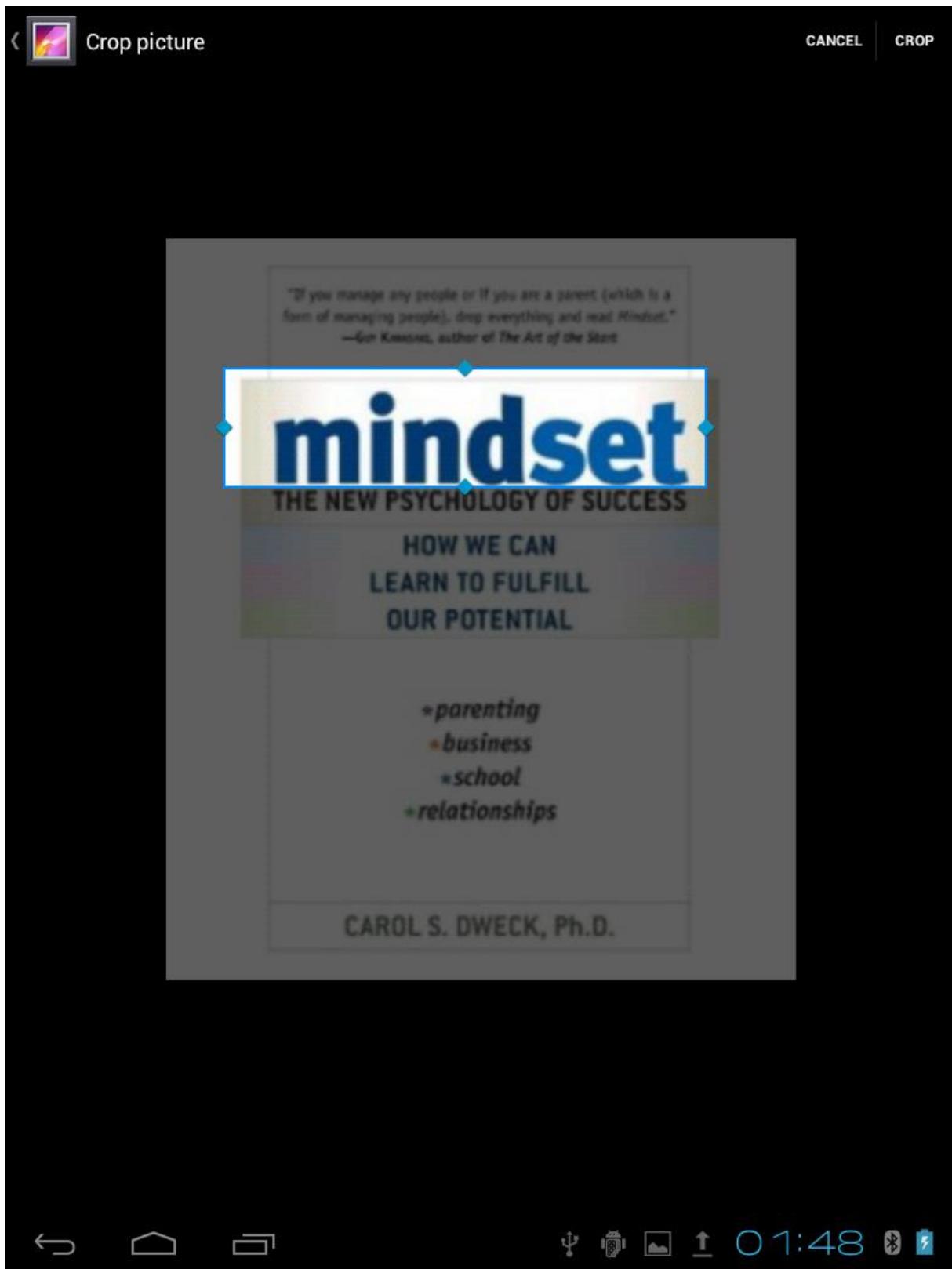




TOO LAZY TO TYPE



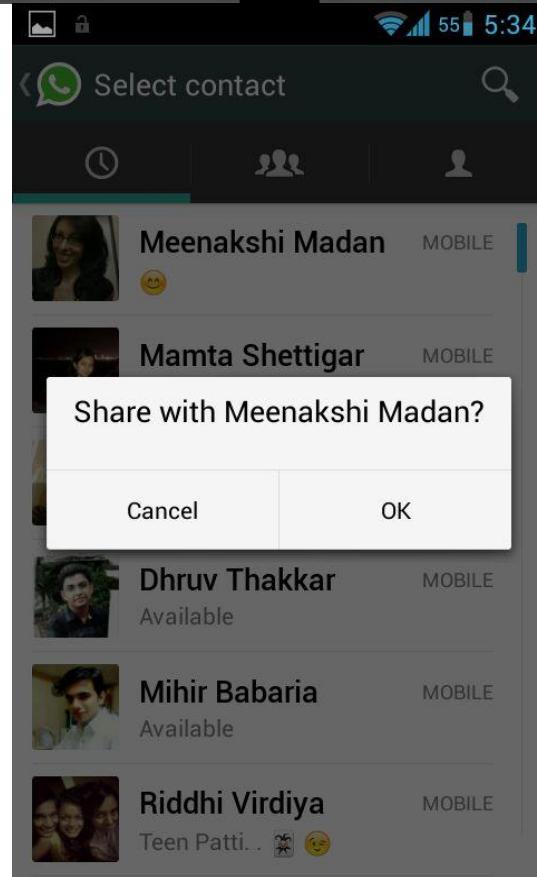
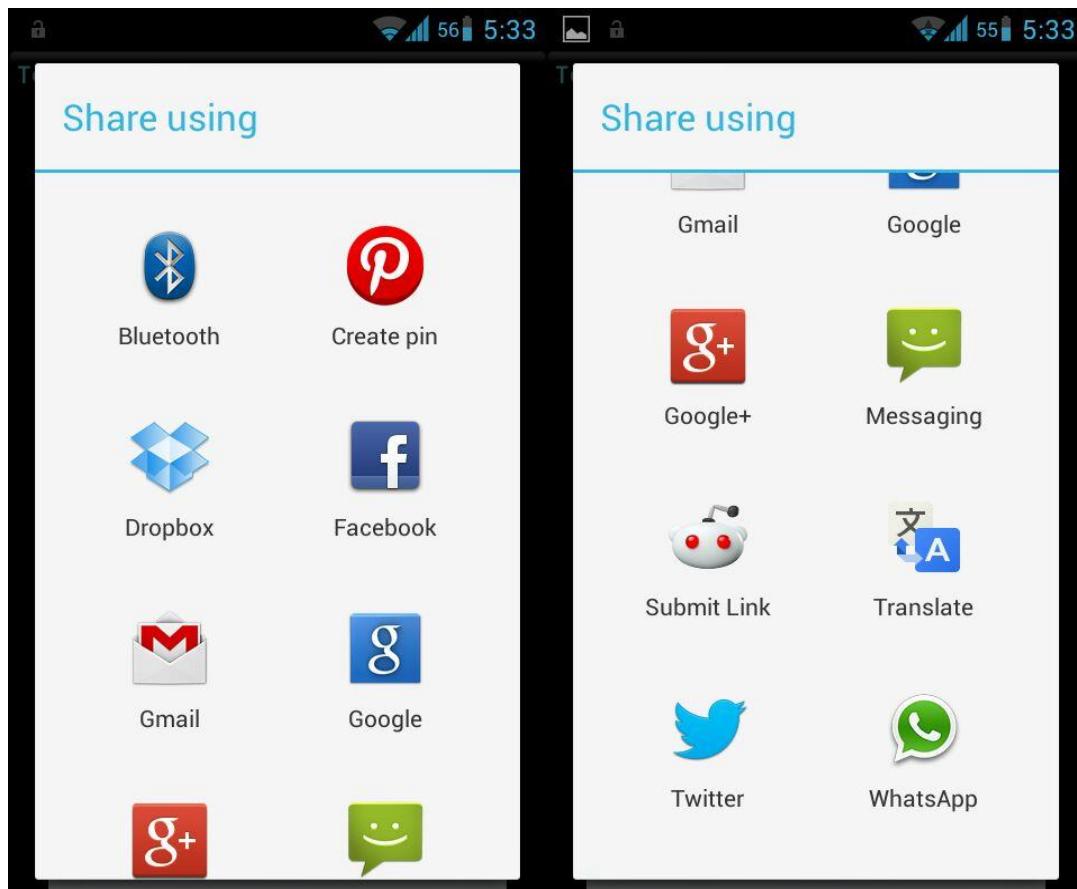




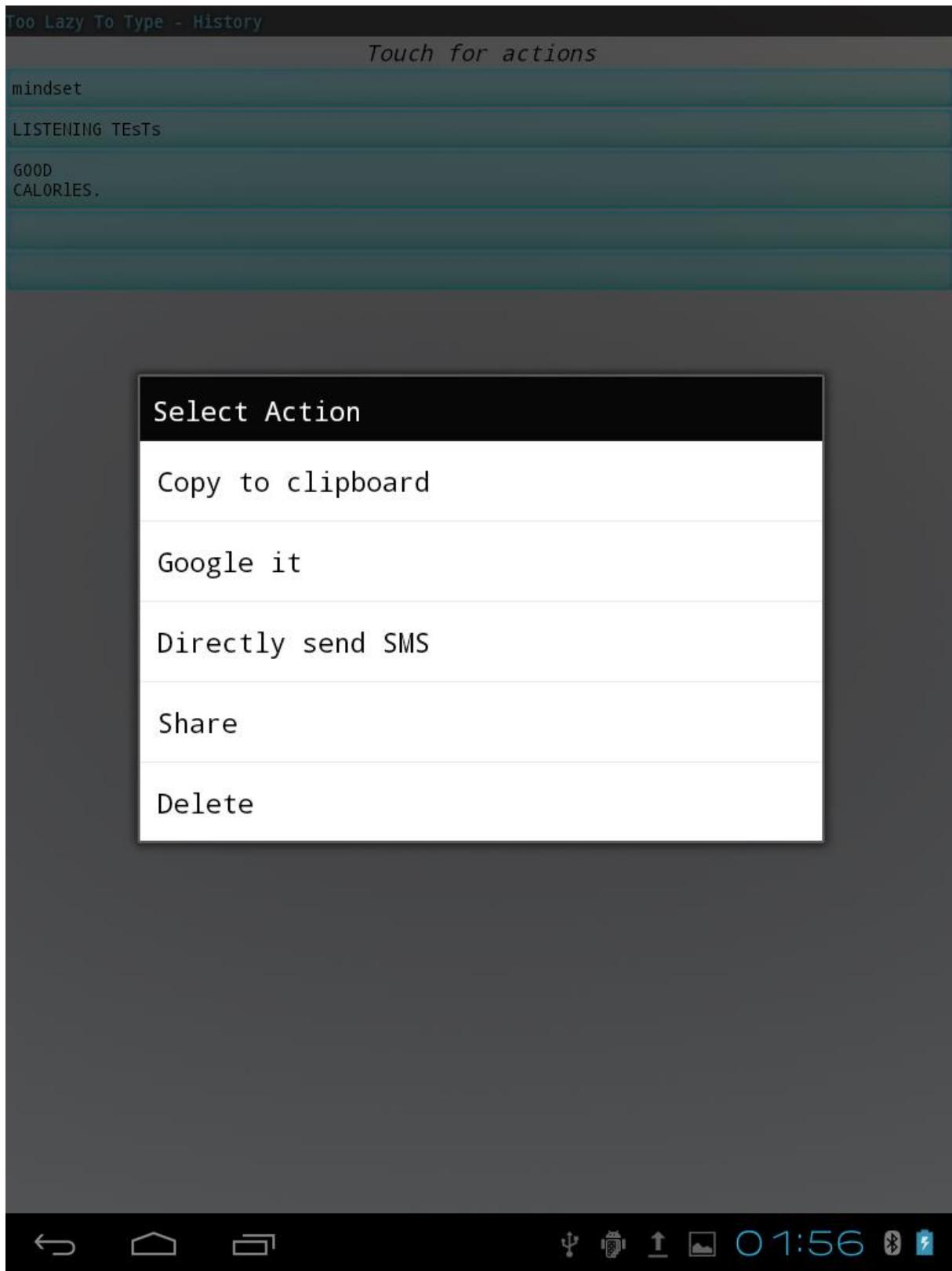
TOO LAZY TO TYPE

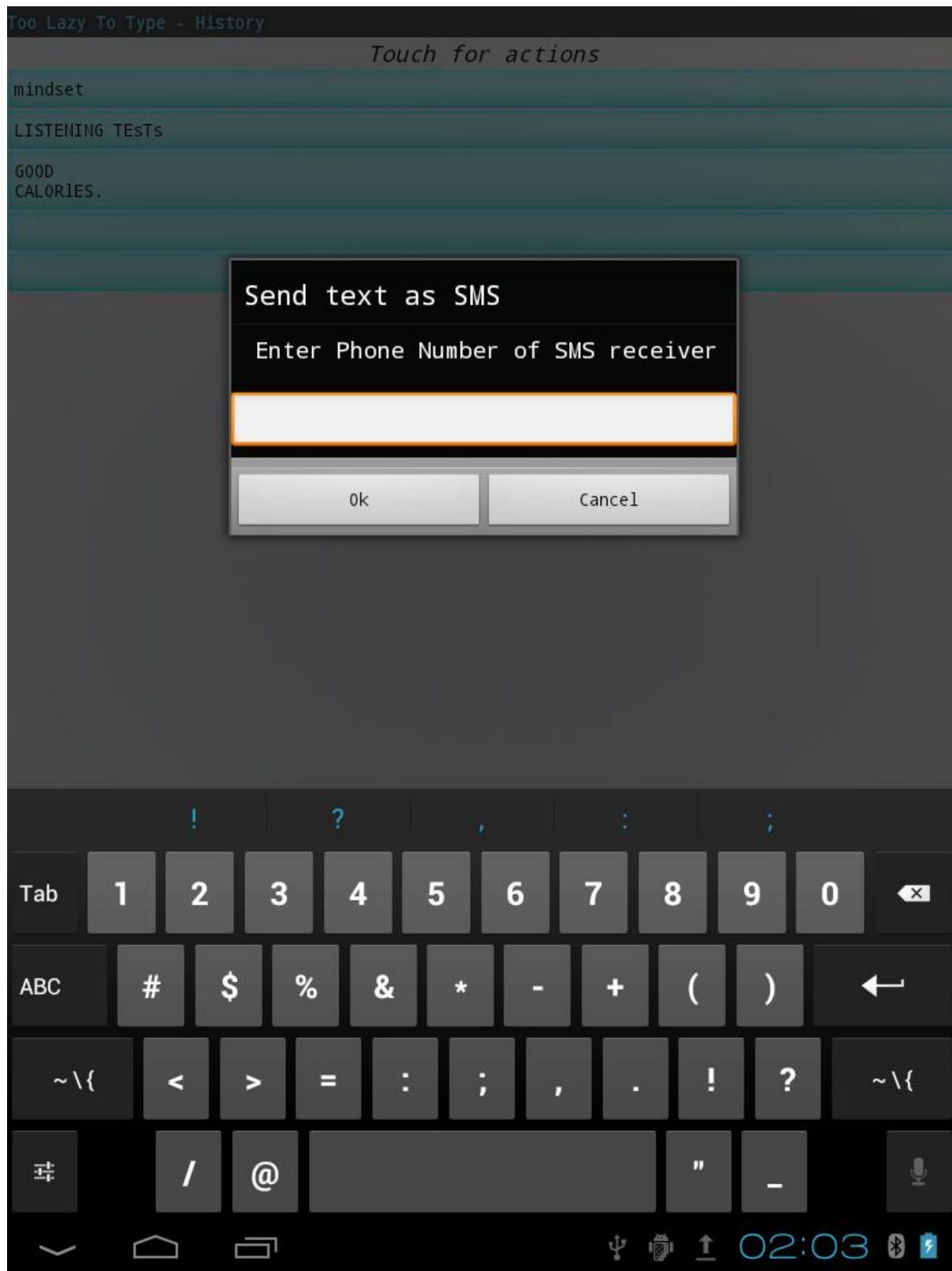


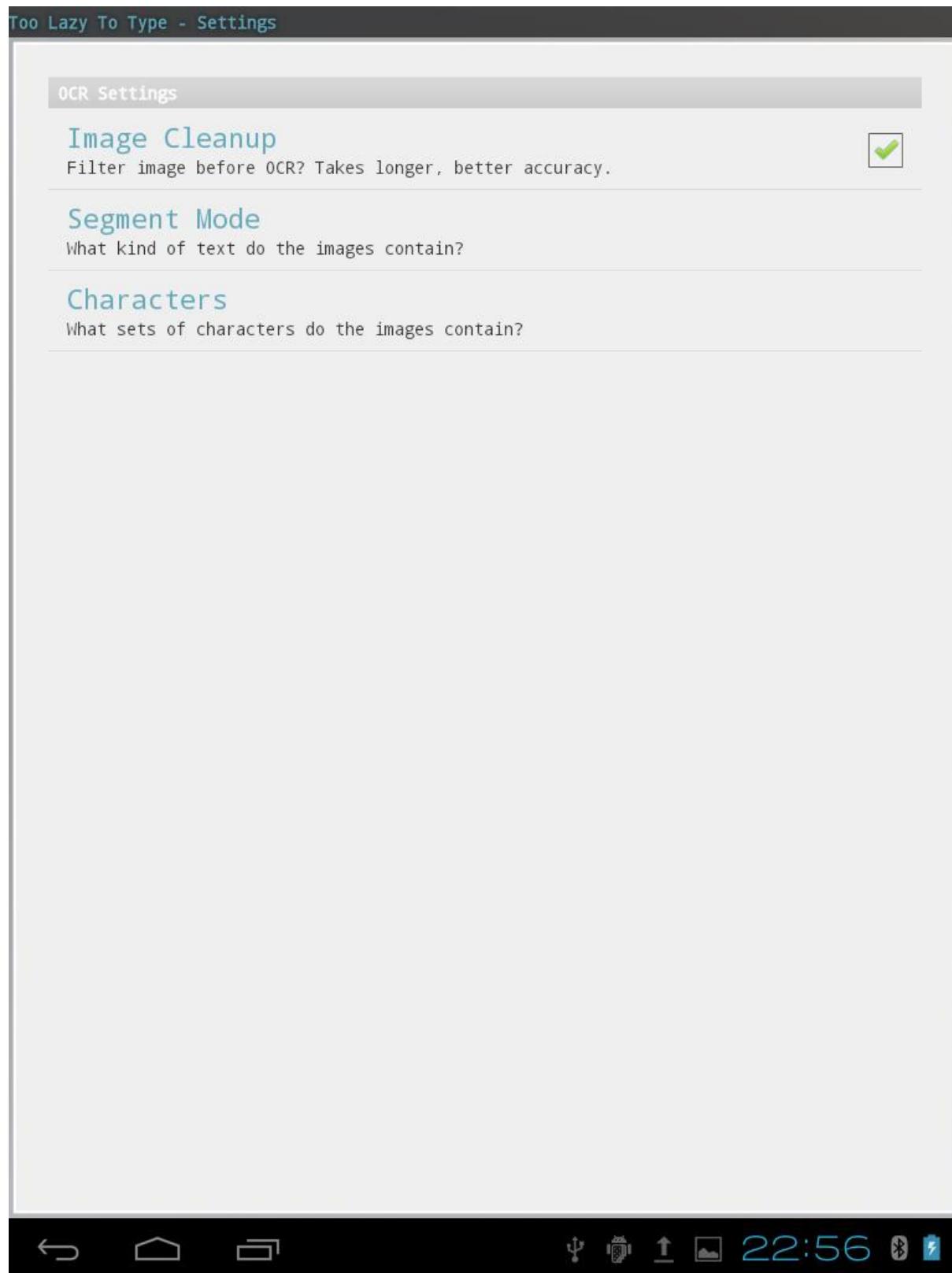


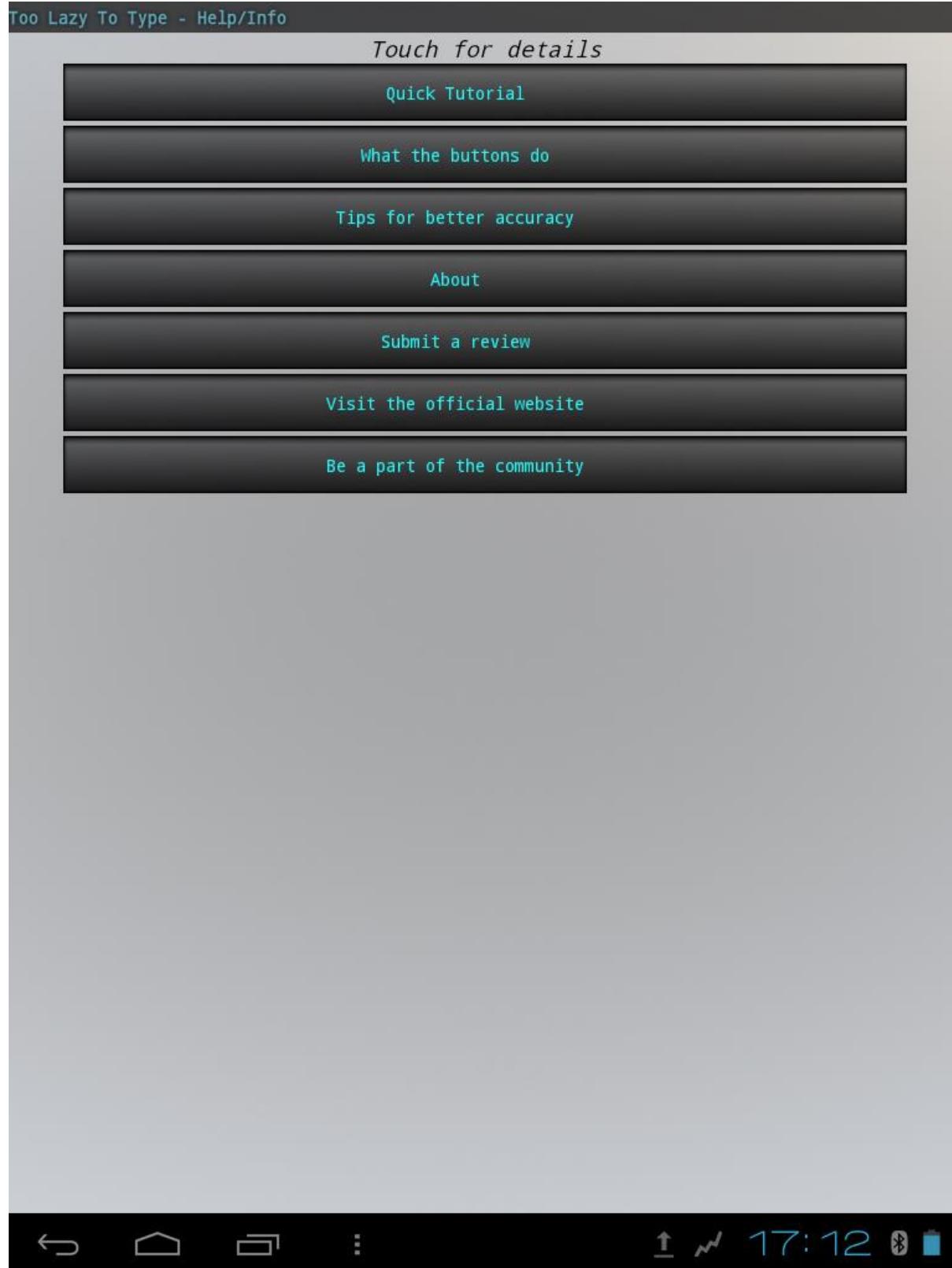


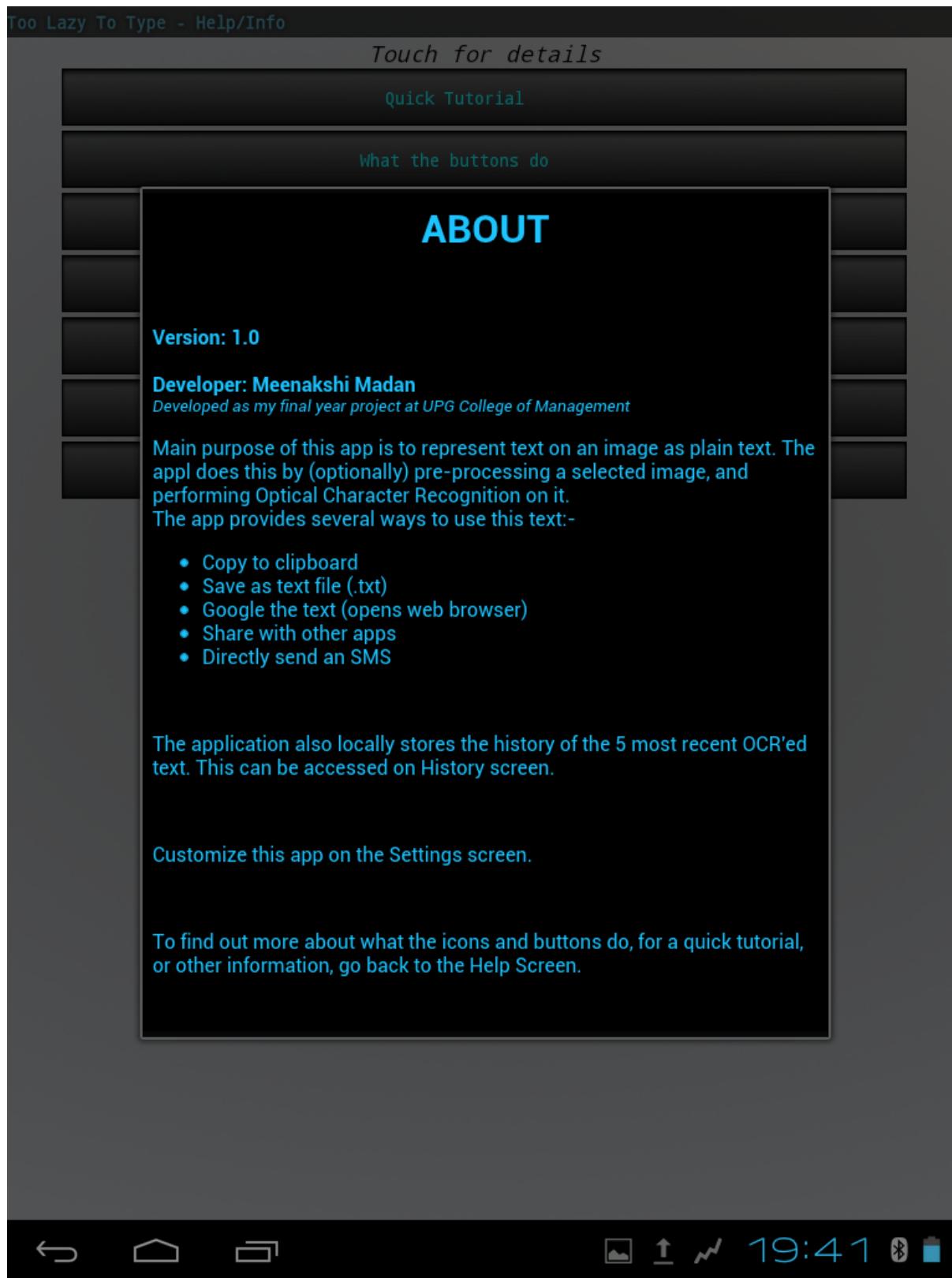


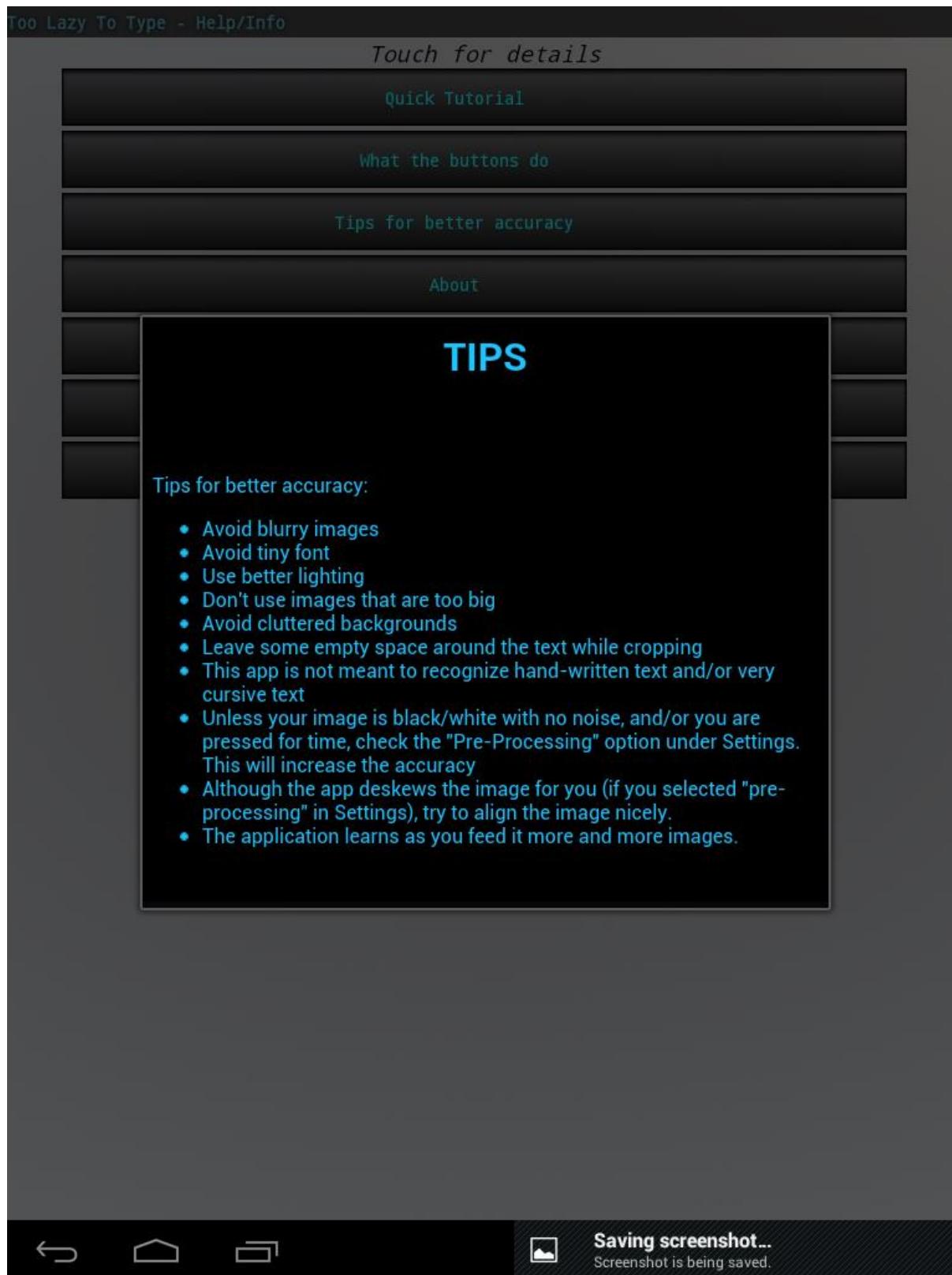








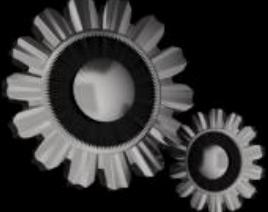




Too Lazy To Type - Help/Info

Buttons

For answers to the question "Ooo what does THIS button do?"

Button Icon	Description
	Standard icon for OCR. Follow this button (widget -> main screen -> ocr screen) as a definite path to OCR.
	Takes you to the Settings/Preferences screen. On this screen you can customize language, pre-processing, and other OCR parameters.
	Opens up History screen. 5 most recent text that has been extracted in this app. On touching/clicking any one of the entries, a pop-up selection menu will let you perform actions with this text such as delete the entry, save it, copy it, share it, directly send as sms, google it, etc.
	Click this for help and information about this app.



19:40







19:41





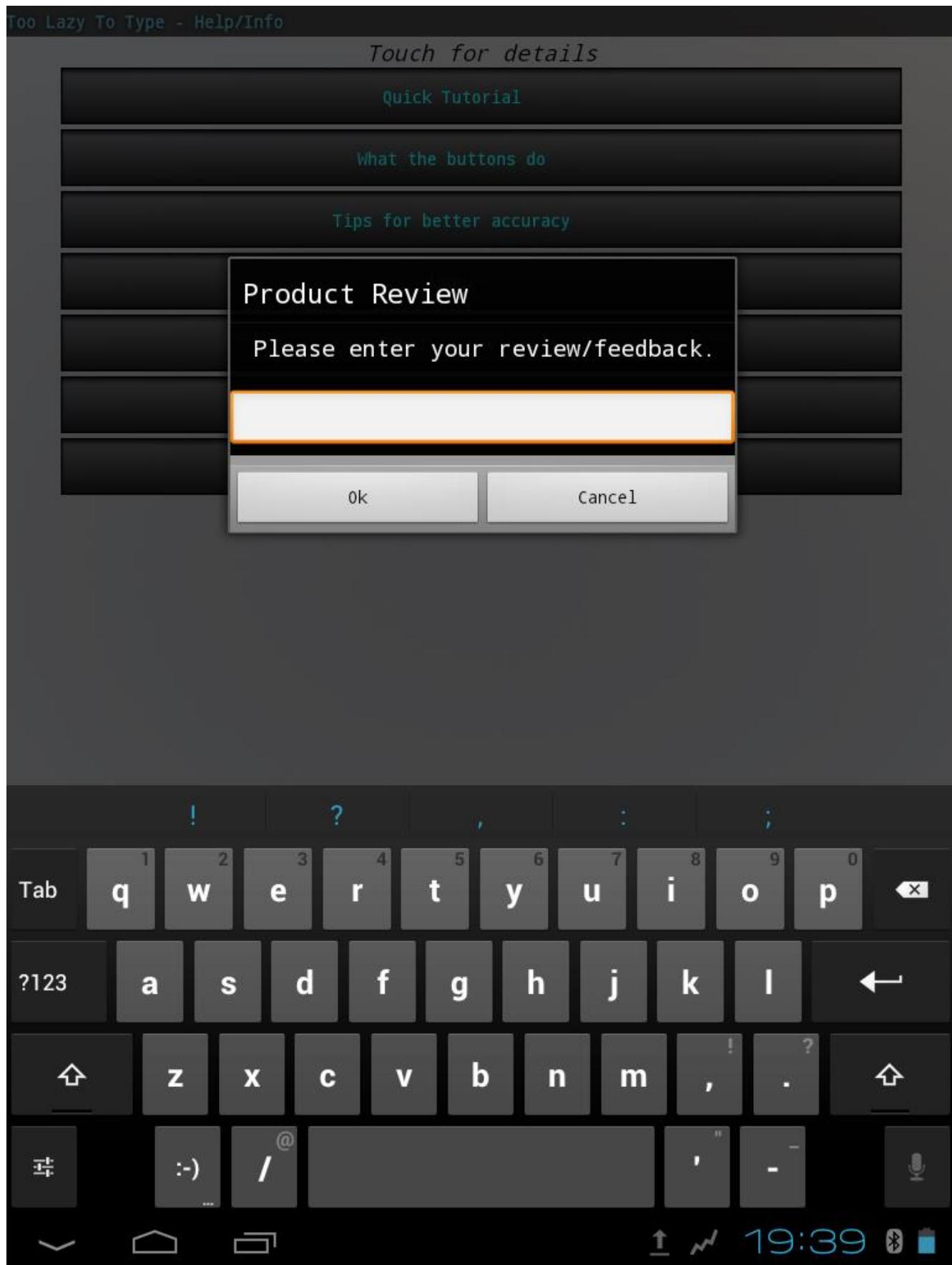
19:39



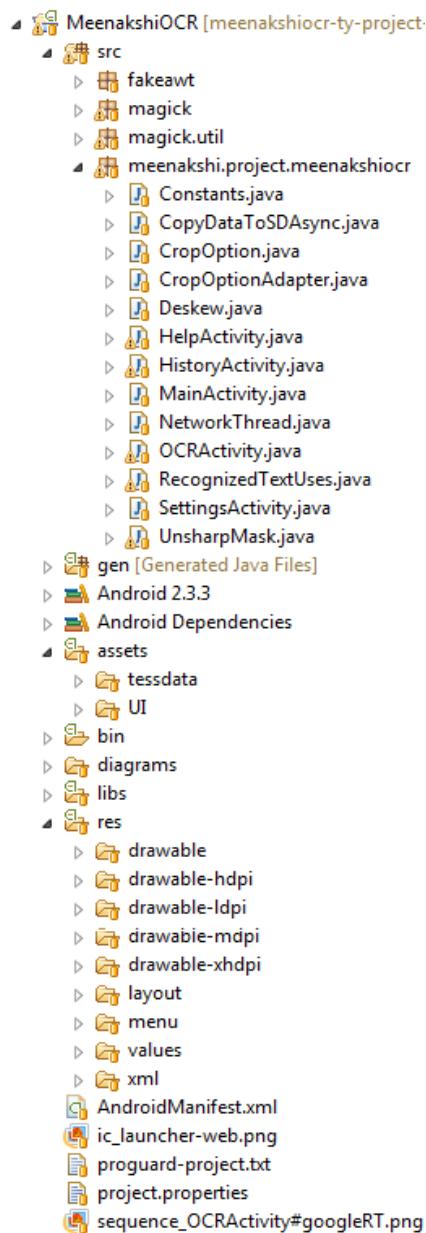








APK structure and source code



MainActivity.java

```

package meenakshi.project.meenakshiorc;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.webkit.WebView;

public class MainActivity extends Activity {

    public static String PREFS_NAME = "OCRSettings";

    private SharedPreferences mPreferences;
    private static final String TAG = "MainActivity.java";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mPreferences = getSharedPreferences("MeenakshiOCRSharedPreferences", Context.MODE_PRIVATE);
        boolean firstTime = mPreferences.getBoolean("welcome", true);
        if (firstTime) {
            Log.v(TAG, "In first time if block");
            SharedPreferences.Editor editor = mPreferences.edit();
            editor.putBoolean("welcome", false);
            editor.putString("lang", "eng");
            editor.putString("OCRTxMode", "default");
            editor.putString("DATA_PATH", getExternalFilesDir(null).getAbsolutePath() + "/");
            editor.putString("CURRENT_IMAGE_PATH", getExternalFilesDir(null).getAbsolutePath() + "/" + "/currentocr.jpg");
            editor.commit();
            Constants.initializeConstants(this);
            new CopyDataToSDAsync(this).execute();
        }

        AlertDialog.Builder imageDialog = new AlertDialog.Builder(this);
        LayoutInflater inflater = (LayoutInflater) getSystemService(LAYOUT_INFLATER_SERVICE);

        View layout = inflater.inflate(R.layout.about_layout,
                ( ViewGroup ) findViewById(R.id.layout_root));

        WebView webView = (WebView) layout.findViewById(R.id.wvabout);
        if(webView==null)
        {
            Log.v("help", "webview is null o_o");
        }
        webView.loadUrl("file:///android_asset/UI/welcome.html");

        imageDialog.setView(layout);
        imageDialog.create();
        imageDialog.show();
    }

    Constants.initializeConstants(this);

}

public void goToOCR(View view)
{
    Intent intent = new Intent(this, OCRAActivity.class);
    startActivity(intent);
}

```

```

public void goToSettings(View view)
{
    Intent intent = new Intent(this, SettingsActivity.class);
    startActivity(intent);
}

public void goToHistory(View view)
{
    Intent intent = new Intent(this, HistoryActivity.class);
    startActivity(intent);
}

public void goToHelp(View view)
{
    Intent intent = new Intent(this, HelpActivity.class);
    startActivity(intent);
}

}

```

OCRActivity.java

```

package meenakshi.project.meenakshioocr;

import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.PendingIntent;
import android.content.ActivityNotFoundException;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.content.pm.ResolveInfo;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.telephony.SmsManager;
import android.text.ClipboardManager;
import android.text.method.ScrollingMovementMethod;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;

public class OCRActivity extends Activity {

    private Uri mImageCaptureUri;

```

```

private static final int PICK_FROM_CAMERA = 1;
private static final int CROP_FROM_CAMERA = 2;
private static final int PICK_FROM_FILE = 3;
private static final int SEND_SMS = 4;
private static final String TAG = "OCRActivity.java";
protected TextView _field;
public String recognizedText;

private SharedPreferences mPreferences;

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

    _field = (TextView) findViewById(R.id.recogText);
    _field.setMovementMethod(new ScrollingMovementMethod());
    ImageButton button = (ImageButton) findViewById(R.id.btn_startOCR);

    final String [] items = new String [] {"Take from camera", "Select from gallery"};
    ArrayAdapter<String> adapter = new ArrayAdapter<String> (this, android.R.layout.select_dialog_item,items);
    AlertDialog.Builder builder = new AlertDialog.Builder(this);

    builder.setTitle("Select Image");
    builder.setAdapter(adapter, new DialogInterface.OnClickListener() {
        @Override
        public void onClick( DialogInterface dialog, int item ) { //pick from camera
            if (item == 0) {
                Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

                File file = new File(Constants.CURRENT_IMAGE_PATH);
                mImageCaptureUri = Uri.fromFile(file);

                intent.putExtra(android.provider.MediaStore.EXTRA_OUTPUT, mImageCaptureUri);

                try {
                    intent.putExtra("return-data", true);

                    startActivityForResult(intent, PICK_FROM_CAMERA);
                } catch (ActivityNotFoundException e) {
                    e.printStackTrace();
                }
            } else { //pick from file
                Intent intent = new Intent();

                intent.setType("image/*");
                intent.setAction(Intent.ACTION_GET_CONTENT);

                startActivityForResult(Intent.createChooser(intent, "Complete action using"), PICK_FROM_FILE);
            }
        }
    });

    final AlertDialog dialog = builder.create();
    dialog.setCanceledOnTouchOutside(true);

    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            dialog.show();
        }
    });
}

public void copyRTToClipBoard(View v)
{
    RecognizedTextUses.copyRTToClipBoard(recognizedText, this);
}

```

```

public void googleRT(View v)
{
    RecognizedTextUses.googleRT(recognizedText, this);
}

public void share(View v)
{
    RecognizedTextUses.share(recognizedText, this);
}

public void saveRTToFile(View v)
{
    Log.v(TAG, "In save to file button call");
    mPreferences = getSharedPreferences("MeenakshiOCRSharedPreferences", Context.MODE_PRIVATE);

    try
    {
        File root = new File(Environment.getExternalStorageDirectory(), "OCRNNotes");
        if (!root.exists())
        {
            root.mkdirs();
        }
        int count = mPreferences.getInt("textFileCounter", 1);
        File gpxfile = new File(root, count + ".txt");
        FileWriter writer = new FileWriter(gpxfile);
        writer.append(recognizedText);
        writer.flush();
        writer.close();

        Toast.makeText(this, "Saved to OCRNotes/" + count + ".txt", Toast.LENGTH_SHORT).show();
        Log.v(TAG, gpxfile.getAbsolutePath());

        SharedPreferences.Editor editor = mPreferences.edit();
        editor.putInt("textFileCounter", ++count);
        editor.commit();
    }
    catch(IOException e)
    {
        e.printStackTrace();
        Log.v(TAG, e.toString());
    }
}
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode != RESULT_OK) return;

    switch (requestCode) {
        case PICK_FROM_CAMERA:
            doCrop();

            break;

        case PICK_FROM_FILE:
            mImageCaptureUri = data.getData();

            doCrop();

            break;

        case CROP_FROM_CAMERA:
            Bundle extras = data.getExtras();

            if (extras != null) {
                Bitmap photo = extras.getParcelable("data");

                try {
                    FileOutputStream out = new FileOutputStream(Constants.CURRENT_IMAGE_PATH);
                    photo.compress(Bitmap.CompressFormat.JPEG, 100, out);
                }
                catch (IOException e)
                {
                    e.printStackTrace();
                }
            }

            break;
    }
}

```

```

        File file = new File(Constants.CURRENT_IMAGE_PATH);
        mImageCaptureUri = Uri.fromFile(file);
    } catch (Exception e) {
        e.printStackTrace();
        Log.v(TAG, "Oh noes, couldn't save cropped file to _path" + e.toString());
    }

    new UnsharpMask(this).execute();
}

break;

}

private void doCrop() {
    final ArrayList<CropOption> cropOptions = new ArrayList<CropOption>();

    Intent intent = new Intent("com.android.camera.action.CROP");
    intent.setType("image/*");

    List<ResolveInfo> list = getPackageManager().queryIntentActivities( intent, 0 );

    int size = list.size();

    if (size == 0) {
        Toast.makeText(this, "Can not find image crop app", Toast.LENGTH_SHORT).show();
        return;
    } else {
        intent.setData(mImageCaptureUri);

        intent.putExtra("scale", true);
        intent.putExtra("return-data", true);

        if (size == 1) {
            Intent i = new Intent(intent);
            ResolveInfo res = list.get(0);

            i.setComponent( new ComponentName(res.activityInfo.packageName, res.activityInfo.name));
            startActivityForResult(i, CROP_FROM_CAMERA);
        } else {
            for (ResolveInfo res : list) {
                final CropOption co = new CropOption();

                co.title = getPackageManager().getApplicationLabel(res.activityInfo.applicationInfo);
                co.icon = getPackageManager().getApplicationIcon(res.activityInfo.applicationInfo);
                co.appIntent = new Intent(intent);

                co.appIntent.setComponent( new ComponentName(res.activityInfo.packageName, res.activityInfo.name));

                cropOptions.add(co);
            }
        }
    }

    CropOptionAdapter adapter = new CropOptionAdapter(getApplicationContext(), cropOptions);

    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Choose Crop App");
    builder.setAdapter( adapter, new DialogInterface.OnClickListener() {
        @Override
        public void onClick( DialogInterface dialog, int item ) {
            startActivityForResult( cropOptions.get(item).appIntent, CROP_FROM_CAMERA);
        }
    });

    builder.setOnCancelListener( new DialogInterface.OnCancelListener() {
        @Override
        public void onCancel( DialogInterface dialog ) {

            if (mImageCaptureUri != null ) {
                getContentResolver().delete(mImageCaptureUri, null, null );
            }
        }
    });
}

```

```
        mImageCaptureUri = null;
    }
}
});
```

AlertDialog alert = builder.create()

```
        alert.show();
```

}

```
}
```

}

```
}
```

UnsharpMask.java

```
package meenakshi.project.meenakshioocr;

/**This class hands the OCR processing and unsharp-masking for processing the image
 * author: Meenakshi Madan
 */

import java.io.IOException;

import magick.CompressionType;
import magick.ImageInfo;
import magick.MagickImage;
import magick.PixelPacket;
import magick.util.MagickBitmap;
import android.app.AlertDialog;
import android.content.Context;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.media.ExifInterface;
import android.os.AsyncTask;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;
import android.widget.TableRow;
import android.widget.Toast;

import com.googlecode.tesseract.android.TessBaseAPI;

public class UnsharpMask extends AsyncTask<Void, Integer, Void> {

    ProgressDialog pg;

    /** Tag for logging purposes */
    String TAG = "UnsharpMask";

    /** To check if OCR needs to be performed again on the same image - if the confidence value is very low
     boolean checkOnceForFurtherProcessing = true;

    /** Number of times ocr has been performed in this transaction */
    int tessRepeatCount = 0;

    /** Maximum number of times that OCR can be performed on the image in this transaction */
    int tessRepeatMAXCOUNT = 5;

    /** Mean confidence as returned by tesseract on the recognized text */
    int meanConfidence_original, meanConfidence_processed=0;
    String text_original, text_processed="";

    static int LEVEL_ORIGINAL = 0, LEVEL_PROCESSED=1;

    private SharedPreferences ocrPref;
    private SharedPreferences mPreferences;
```

```

private String BLACK_LIST_AUTOMATIC = "#$%^&+=;;{}[]/.!@\\|><~`\"*()";
private String WHITE_LIST_AUTOMATIC = "1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";

/** Object of OCRActivity, to access variables such as DATA_PATH and view elements */
private OCRActivity act;

public UnsharpMask(OCRActivity act) {
    Log.v(TAG, "Begin constructor");
    this.act = act;

    ocrPref = act.getSharedPreferences(MainActivity.PREFS_NAME, Context.MODE_PRIVATE);
}

@Override
protected void onPreExecute() {
    Log.v("CopData AsyncTask Mein", "Entered onPreExecute");
    pg = ProgressDialog.show(act, "", "Processing...", true);
}

/** Displays text to the user, hides progress bar
 */
@Override
protected void onPostExecute(Void result) {
    Log.v("AsyncTask Mein", "Entered onPostExecute");

    if(meanConfidence_original > meanConfidence_processed)
    {
        act.recognizedText = text_original;
    }
    else {
        act.recognizedText = text_processed;
    }

    pg.dismiss();

    if ( act.recognizedText.length() != 0 ) {
        act._field.setText(act.recognizedText);

        ((TableRow)act.findViewById(R.id.tableRow3)).setVisibility(View.VISIBLE);
        ((TableRow)act.findViewById(R.id.tableRow4)).setVisibility(View.VISIBLE);
        ((ImageView)act.findViewById(R.id.dbrobotarms)).setImageResource(R.drawable.ocrscreen20);

        mPreferences = act.getSharedPreferences("MeenakshiOCRSharedPreferences", Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = mPreferences.edit();
        for(int i=4; i>=1; i--)
        {
            editor.putString("his" + i, mPreferences.getString("his" + (i-1), " "));
        }
        editor.putString("his0", act.recognizedText);
        editor.commit();
    }
    else
    {
        Toast.makeText(act, "Oops, no text found!", Toast.LENGTH_SHORT).show();
    }
}

/** Performs OCR on the bitmap at _path on SDCARD. Also performs any orientation required
 */

```

```

protected void performOCR(int level)
{
    BitmapFactory.Options options = new BitmapFactory.Options();
    options.inSampleSize = 1;

    Bitmap bitmap = BitmapFactory.decodeFile(Constants.CURRENT_IMAGE_PATH, options);

    // Getting width & height of the given image.
    int w = bitmap.getWidth();
    int h = bitmap.getHeight();

    try {
        ExifInterface exif = new ExifInterface(Constants.CURRENT_IMAGE_PATH);
        int exifOrientation = exif.getAttributeInt(
            ExifInterface.TAG_ORIENTATION,
            ExifInterface.ORIENTATION_NORMAL);

        Log.v(TAG, "Orient: " + exifOrientation);

        int rotate = 0;

        switch (exifOrientation) {
            case ExifInterface.ORIENTATION_ROTATE_90:
                rotate = 90;
                break;
            case ExifInterface.ORIENTATION_ROTATE_180:
                rotate = 180;
                break;
            case ExifInterface.ORIENTATION_ROTATE_270:
                rotate = 270;
                break;
        }

        Log.v(TAG, "Rotation: " + rotate);

        if (rotate != 0) {

            // Setting pre rotate
            Matrix mtx = new Matrix();
            mtx.preRotate(rotate);

            // Rotating Bitmap
            bitmap = Bitmap.createBitmap(bitmap, 0, 0, w, h, mtx, false);
        }

        // Convert to ARGB_8888, required by tess
        bitmap = bitmap.copy(Bitmap.Config.ARGB_8888, true);
    } catch (IOException e) {
        Log.e(TAG, "Couldn't correct orientation: " + e.toString());
    }

    Log.v(TAG, "Before baseApi");
    TessBaseAPI baseApi = new TessBaseAPI();
    if(ocrPref.getString("whiteilst", "None").equals("None"))
    {
        baseApi.setVariable(TessBaseAPI.VAR_CHAR_BLACKLIST, BLACK_LIST_AUTOMATIC);
        baseApi.setVariable(TessBaseAPI.VAR_CHAR_WHITELIST, WHITE_LIST_AUTOMATIC);

        Log.v(TAG, "whitelist preferences returned None");
    }
    else
    {
        baseApi.setVariable(TessBaseAPI.VAR_CHAR_WHITELIST, ocrPref.getString("whitelist", "None"));

        Log.v(TAG, "whitelist preferences returned " + ocrPref.getString("whitelist", "None"));
    }

    if(!ocrPref.getString("psm", "None").equals("None"))

```

```

    {
        baseApi.setPageSegMode(Integer.parseInt(ocrPref.getString("psm", "None")));
        Log.v(TAG, "PSM preferences returned " + ocrPref.getString("psm", "None"));
    }

    Log.v(TAG, "PSM preferences returned None");

    baseApi.setDebug(true);
    Log.v(TAG, "After setting variables");
    baseApi.init(Constants.DATA_PATH, Constants.LANG); //, TessBaseAPI.OEM_CUBE_ONLY
    Log.v(TAG, "After init and before setting bitmap");
    baseApi.setImage(bitmap);
    Log.v(TAG, "After init and before getUTF8Text");
    if(level == UnsharpMask.LEVEL_ORIGINAL)
    {
        text_original = baseApi.getUTF8Text();
        meanConfidence_original = baseApi.meanConfidence();

        Log.v(TAG, "OCRED TEXT: " + text_original);
        Log.v(TAG, "Mean Confidence: " + meanConfidence_original);
    }
    else if(level == UnsharpMask.LEVEL_PROCESSED)
    {
        text_processed = baseApi.getUTF8Text();
        meanConfidence_processed = baseApi.meanConfidence();

        Log.v(TAG, "OCRED TEXT: " + text_processed);
        Log.v(TAG, "Mean Confidence: " + meanConfidence_processed);
    }

    baseApi.end();

    if (baseApi != null) {
        baseApi.clear();
    }
}

void performProcessing()
{
    try{
        ImageInfo mi = new ImageInfo(Constants.CURRENT_IMAGE_PATH);
        MagickImage m = new MagickImage(mi);
        if(m.normalizeImage()) Log.v(TAG, "normalize conversion successful");
        else Log.v(TAG, "normalize conversion unsuccessful");

        Deskew d = new Deskew(MagickBitmap.ToBitmap(m));
        double skew = d.GetSkewAngle();
        Log.v(TAG, "After Deskew, skew = " + skew);

        m = m.rotateImage(-skew); // /57.295779513082320876798154814105
        m.setDepth(8);

        m = m.sharpenImage(10, 8);

        if(m.negateImage(0)) Log.v(TAG, "negate conversion successful");
        else Log.v(TAG, "negate conversion unsuccessful");
        PixelPacker pp = m.getBackgroundColor();
        int bg = pp.getBlue(), thresh;
        Log.v(TAG, "BG color return by getBackgroundColor is: " + bg);
        if (bg<32757) thresh = 60000;
        else thresh = 10000;
        if(m.thresholdImage(32757)) Log.v(TAG, "thresh conversion successful"); //15000
        else Log.v(TAG, "thresh conversion unsuccessful");
        if(m.negateImage(0)) Log.v(TAG, "negate conversion successful");
        else Log.v(TAG, "negate conversion unsuccessful");

        m = m.scaleImage(m.getWidth() + 100, m.getHeight() + 100);

        mi.setDensity("300");
        m.setCompression(CompressionType.NoCompression);
    }
}

```

```

        m.setFileName(Constants.CURRENT_IMAGE_PATH); //give new location
        if(m.writeImage(mi)) Log.v(TAG, "Successfully wrote image to path"); //save
        else Log.v(TAG, "Image save unsuccessful");
    }
    catch(Exception e)
    {
        Log.v(TAG, "exception occured performing magick functions: " + e.toString());
    }

    Log.v(TAG, "In runnable thread, after processing");

}

/**
 * Calls functions to perform required preprocessing and OCR
 */

@Override
protected Void doInBackground(Void... params) {
    // TODO Auto-generated method stub
    //afterProcess = bitmap_Source;
    Log.v(TAG, "In runnable thread, before processing");
    performOCR(UnsharpMask.LEVEL_ORIGINAL);

    if(ocrPref.getBoolean("processimage", true)){
        performProcessing();
        performOCR(UnsharpMask.LEVEL_PROCESSED);

        Log.v(TAG, "Processimage preferences returned true");
    }

    Log.v(TAG, "Processimage preferences returned false");

    Log.v("AsyncTask Mein", "End of do In Background");

    return null;
}

}

```

SettingsActivity.java

```

package meenakshi.project.meenakshiorc;

import android.os.Bundle;
import android.preference.PreferenceActivity;

public class SettingsActivity extends PreferenceActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getPreferenceManager().setSharedPreferencesName(
            MainActivity.PREFS_NAME);
        addPreferencesFromResource(R.xml.prefs);
    }
}

```

HelpActivity.java

```
package meenakshi.project.meenakshior;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.webkit.WebView;
import android.widget.EditText;
import android.widget.Toast;

public class HelpActivity extends Activity {

    static String TAG="HelpActivity";

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

    public void about(View v) {

        AlertDialog.Builder imageDialog = new AlertDialog.Builder(this);
        LayoutInflater inflater = (LayoutInflater) getSystemService(LAYOUT_INFLATER_SERVICE);

        View layout = inflater.inflate(R.layout.about_layout,
                (ViewGroup) findViewById(R.id.layout_root));
        WebView webView = (WebView)layout.findViewById(R.id.wvabout);
        if(webView==null)
        {
            Log.v("help", "webview is null o_o");
        }
        webView.loadUrl("file:///android_asset/UI/about.html");

        imageDialog.setView(layout);
        imageDialog.create();
        imageDialog.show();
    }

    public void tips(View v) {

        AlertDialog.Builder imageDialog = new AlertDialog.Builder(this);
        LayoutInflater inflater = (LayoutInflater) getSystemService(LAYOUT_INFLATER_SERVICE);

        View layout = inflater.inflate(R.layout.about_layout,
                (ViewGroup) findViewById(R.id.layout_root));

        WebView webView = (WebView)layout.findViewById(R.id.wvabout);
        if(webView==null)
        {
            Log.v("help", "webview is null o_o");
        }
        webView.loadUrl("file:///android_asset/UI/tips.html");

        imageDialog.setView(layout);
    }
}
```

```

imageDialog.create();
imageDialog.show();

}

public void buttons(View v) {

AlertDialog.Builder imageDialog = new AlertDialog.Builder(this);
LayoutInflater inflater = (LayoutInflater) getSystemService(LAYOUT_INFLATER_SERVICE);

View layout = inflater.inflate(R.layout.about_layout,
        (ViewGroup) findViewById(R.id.layout_root));

WebView webView = (WebView)layout.findViewById(R.id.wvabout);
if(webView==null)
{
    Log.v("help", "webview is null o_o");
}
webView.loadUrl("file:///android_asset/UI/buttons.html");

imageDialog.setView(layout);
imageDialog.create();
imageDialog.show();

}

public void tutorial(View v) {

AlertDialog.Builder imageDialog = new AlertDialog.Builder(this);
LayoutInflater inflater = (LayoutInflater) getSystemService(LAYOUT_INFLATER_SERVICE);

View layout = inflater.inflate(R.layout.about_layout,
        (ViewGroup) findViewById(R.id.layout_root));

WebView webView = (WebView)layout.findViewById(R.id.wvabout);
if(webView==null)
{
    Log.v("help", "webview is null o_o");
}
webView.loadUrl("file:///android_asset/UI/tutorial.html");

imageDialog.setView(layout);
imageDialog.create();
imageDialog.show();

}

public void review(View v)
{
    final SharedPreferences mPreferences = getSharedPreferences("MeenakshiOCRSharedPreferences", Context.MODE_PRIVATE);
    final String userName = mPreferences.getString("userName1", "1");
    if(userName.equals("1"))
    {
        AlertDialog.Builder alert = new AlertDialog.Builder(this);

        alert.setTitle("New User");
        alert.setMessage("Please enter an alias.");

        // Set an EditText view to get user input
        final EditText input = new EditText(this);
        alert.setView(input);

        alert.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton) {
                String value = input.getText().toString();
                if(!value.matches("[a-zA-Z]*$") || value.equals(" "))
                {
                    Toast.makeText(getApplicationContext(), "That does not look like a name!",
```

```

Toast.LENGTH_SHORT).show();
        }
    }
    else
    {
        SharedPreferences.Editor editor = mPreferences.edit();
        editor.putString("userName1", value);
        editor.commit();
        content(value);
    }
}
});

alert.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int whichButton) {
        // Canceled.
    }
});

alert.show();
}

else
{
    content(userName);
}

}

public void content(final String user)
{
    AlertDialog.Builder alert = new AlertDialog.Builder(this);

    alert.setTitle("Product Review");
    alert.setMessage("Please enter your review/feedback.");

    // Set an EditText view to get user input
    final EditText input = new EditText(this);
    alert.setView(input);

    alert.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
            String value = input.getText().toString();
            if(value.length()>0)
            {
                Log.v(TAG, "In value.length()>0");
                submit(user, value);
            }
        }
    });
}

alert.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int whichButton) {
        // Canceled.
    }
});

alert.show();
}

public void submit(String user, String msg)
{
    new NetworkThread(this, user, msg).execute();
}

public void website(View v)
{
    Uri uriUrl = Uri.parse("http://meenakshi-ocr.appspot.com");
    Intent launchBrowser = new Intent(Intent.ACTION_VIEW, uriUrl);
    startActivity(launchBrowser);
}

```

```

public void community(View v)
{
    Uri uriUrl = Uri.parse("http://meenakshi-ocr-fofou.appspot.com/Too-Lazy-To-Type");
    Intent launchBrowser = new Intent(Intent.ACTION_VIEW, uriUrl);
    startActivity(launchBrowser);
}

```

HistoryActivity.java

```

package meenakshi.project.meenakshioocr;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.TextView;

public class HistoryActivity extends Activity {

    String recognizedText="";

    TextView tv[];
    int clicked = 0;

    private String TAG="HistoryActivity";

    private SharedPreferences mPreferences;

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

        tv = new TextView[5];

        tv[0] = (TextView)findViewById(R.id.his0);
        tv[1] = (TextView)findViewById(R.id.his1);
        tv[2] = (TextView)findViewById(R.id.his2);
        tv[3] = (TextView)findViewById(R.id.his3);
        tv[4] = (TextView)findViewById(R.id.his4);

        mPreferences = getSharedPreferences("MeenakshiOCRSharedPreferences", Context.MODE_PRIVATE);
        tv[0].setText(mPreferences.getString("his0", " "));
        tv[1].setText(mPreferences.getString("his1", " "));
        tv[2].setText(mPreferences.getString("his2", " "));
        tv[3].setText(mPreferences.getString("his3", " "));
        tv[4].setText(mPreferences.getString("his4", " "));

        final String [] items
        = new String [] {"Copy to clipboard", "Google it", "Directly send SMS", "Share",
        "Delete"};
        ArrayAdapter<String> adapter
        = new ArrayAdapter<String> (this, android.R.layout.select_dialog_item,items);
        AlertDialog.Builder builder
        = new AlertDialog.Builder(this);

        builder.setTitle("Select Action");
        builder.setAdapter( adapter, new DialogInterface.OnClickListener() {
            @Override

```

```

        public void onClick( DialogInterface dialog, int item ) { //pick from camera
            if (item == 0)
                copyRTToClipBoard();
            else if(item==1)
                googleRT();
            else if(item == 2)
                sendSMS();
            else if(item == 3)
                share();
            else
                delete();

        }

    });

final AlertDialog dialog = builder.create();
dialog.setCanceledOnTouchOutside(true);

tv[0].setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        clicked = 0;
        recognizedText = (String) tv[0].getText();
        if(!recognizedText.equals("") && !recognizedText.equals(" "))
            dialog.show();
    }
});

tv[1].setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        clicked = 1;
        recognizedText = (String) tv[1].getText();
        if(!recognizedText.equals("") && !recognizedText.equals(" "))
            dialog.show();
    }
});

tv[2].setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        clicked = 2;
        recognizedText = (String) tv[2].getText();
        if(!recognizedText.equals("") && !recognizedText.equals(" "))
            dialog.show();
    }
});

tv[3].setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        clicked = 3;
        recognizedText = (String) tv[3].getText();
        if(!recognizedText.equals("") && !recognizedText.equals(" "))
            dialog.show();
    }
});

tv[4].setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        clicked = 4;
        recognizedText = (String) tv[4].getText();
        if(!recognizedText.equals("") && !recognizedText.equals(" "))
            dialog.show();
    }
});

}

public void delete()

```

```

{
    SharedPreferences.Editor editor = mPreferences.edit();

    for(int i=clicked; i<=3; i++)
    {
        tv[i].setText(tv[i+1].getText());
        editor.putString("his" + i, (String) tv[i+1].getText());
    }
    tv[4].setText(" ");
    editor.putString("his4", " ");
    editor.commit();
}

public void copyRTToClipBoard()
{
    RecognizedTextUses.copyRTToClipBoard(recognizedText, this);
}

public void googleRT()
{
    RecognizedTextUses.googleRT(recognizedText, this);
}

public void share()
{
    RecognizedTextUses.share(recognizedText, this);
}

public void sendSMS()
{
    RecognizedTextUses.sendSMS(recognizedText, this);
}
}

```

main.py

```

import webapp2
import re
import os
import jinja2
import urllib
import urllib2
import json
from google.appengine.ext import db

jinja_environment=jinja2.Environment(loader=jinja2.FileSystemLoader(os.path.join(os.path.dirname(__file__),'templates')), autoescape=True)

class MainHandler(webapp2.RequestHandler):
    def get(self):
        reviews=db.GqlQuery("Select * from Reviewss order by created desc")
        self.write(reviews=reviews)

    def write(self, **params):
        template = jinja_environment.get_template('home6.html')
        self.response.out.write(template.render(params))

    def post(self):
        reviews=db.GqlQuery("Select * from Reviewss order by created desc")
        self.write(reviews=reviews)

```

```

class ReviewsHandler(webapp2.RequestHandler):
    def get(self):
        reviews=db.GqlQuery("Select * from Reviewss order by created desc")
        self.write(reviews=reviews)

    def write(self, **params):
        template = jinja_environment.get_template('reviews2.html')
        self.response.out.write(template.render(params))

    def post(self):
        self.response.headers['Content-Type']='text/plain'
        content=self.request.get("content")
        user = self.request.get("user")
        if content:
            a=Reviewss(content=content, user=user)
            a.put()
            self.response.out.write("success")
        else:
            self.response.out.write("failure")

class DevHandler(webapp2.RequestHandler):
    def get(self):
        self.write()

    def write(self, **params):
        template = jinja_environment.get_template('dev.html')
        self.response.out.write(template.render(params))

def datatetimeformat(value, format="%H:%M / %d-%m-%Y"):
    return value.strftime(format)

jinja_environment.filters['datatetimeformat'] = datatetimeformat

class Reviewss(db.Model):
    user=db.StringProperty()
    content=db.TextProperty(required=True)
    created=db.DateTimeProperty(auto_now_add=True)

app = webapp2.WSGIApplication([
    ('/', MainHandler),
    ('/reviews', ReviewsHandler),
    ('/dev', DevHandler)
], debug=True)

```

Transition

Tasks

- Final documentation
- Prepare for presentation

Objectives

- Wrap up and present project

Deliverables

- Black book
- Final product

Deployment Pre-requisites and Summary

Pre-requisites

- Minimum Android version required ⇒ 2.2 Froyo
- Camera app (if you don't have this, you can always use pictures from gallery)
- Gallery app (if you don't have this, you can always click new pictures using the built-in camera)
- Permissions required:-
 - Write on SDCard
 - Send data to other apps
 - Send SMS directly
- Internet connection ONLY IF you want to use some of the optional features. Not required otherwise

Summary

- ✓ Java was the primary coding language
- ✓ Some parts of the code were implemented in native C. Doing this improved the performance of the app.

- ✓ The help screen is static for the most part and thus I created and formatted individual help topics as local html files
- ✓ I used the Eclipse IDE along with the ADT plugin
- ✓ Git was used for distributed revision control
- ✓ I trained and used the Tesseract engine for OCR
- ✓ image-magick was used for image processing. A process of trial and error was required on my part to develop a working combination of filters
- ✓ I designed the Graphical User Interface for the app using Blender (3D modelling) and Gimp
- ✓ The website was developed using Python as the backend language
- ✓ The front-end pages were developed using html, css, and jinja2
- ✓ I made the website graphics in blender, just as I did for the app

Future enhancements and releases

- ★ There is always scope for improvement in accuracy and performance
- ★ Batch processing of images for example processing a few pages of a book and saving the file with page breaks
- ★ Handling of various file formats for saving the text such as PDFs and Documents
- ★ But really, most of what could be done to enhance the app while maintaining harmony and staying true to its basic purpose, has been done.

TOO LAZY TO TYPE
Android OCR app



User Manual

TOO LAZY TO TYPE

User Manual

Author:
Meenakshi Madan

Table of Contents

Configuring your device to install 3rd party apps ..	1
Downloading and installing the app	3
5 steps to OCR.....	4
What can the app do?.....	6
What can I click and what do the buttons do?.....	8
Tips for optimal usage	19
Getting help.....	20
The website	22

Configuring your device to install 3rd party apps

This is required as you'll be downloading the app from my website and not Google Play

As a safety precaution, all Android devices come with the option to install “non-market” apps disabled by default. If you want to allow the installation of non-market, third-party apps such as **Too Lazy To Type** on your smartphone then you’ll have to follow the steps below to get your settings configured properly.

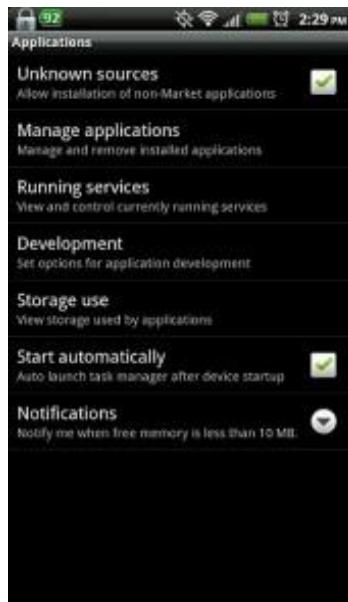
Step 1: Click the MENU button

Step 2: Go to SETTINGS

Step 3: Click on APPLICATIONS

Step 4: Make sure that “UNKNOWN SOURCES” is checked

Step 5: Read the warning message and acknowledge if you are OK to proceed with the changes



That's it!

Downloading and installing the app

Step 1: Simply go to meenakshi-ocr.appspot.com using a Web browser on your Android device and click on "Download".

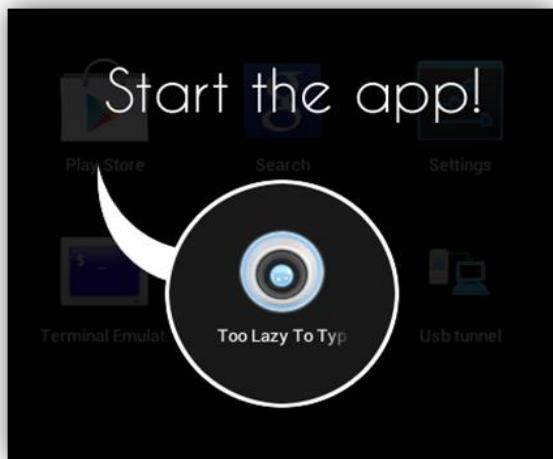
Make sure you read the application description so you'll know whether the app is compatible with your Android device.



Step 2: Locate the APK (Android Package) file on your device and install it (usually the default action when you tap on the APK in your file browser)

5 steps to OCR

Step 1



Step 2



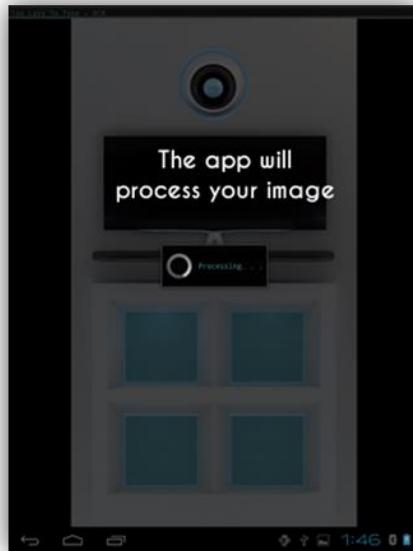
Step 3



Step 4

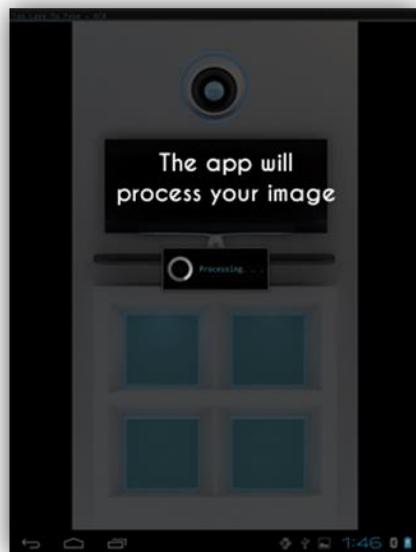


Step 5



Processing...

Done!



What can the app do?

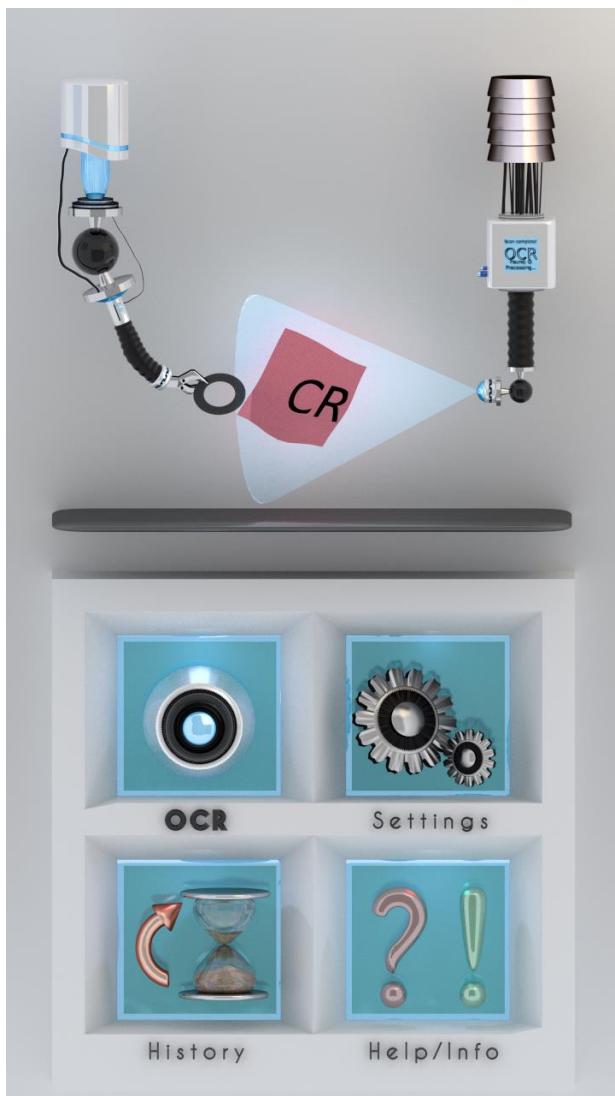
The current version (v1.0) of the app supports the following features.

1. Quick and easy image-to-text **on the go**
2. Allows users to perform OCR on **new images** (taken using camera) or **images already saved** on phone
3. When taking new images, users can **define a specific rectangular area** to analyse
4. OCR'ed text can be **copied to Android clipboard**, from where users can paste it in whichever app they want (SMS/text, e-mail, document editor, etc.)
5. OCR'ed text can be **saved as a text file**
6. OCR'ed text can be quickly **Googled** (only feature that understandably requires an Internet connection)
7. OCR'ed text can be directly sent as **SMS** (without having to open the Messaging app) (only available on recent history screen)
8. OCR'ed text can be **shared** with any other app installed on your phone that accepts text like **Facebook, SMS, Bluetooth, Gmail, Twitter, Whatsapp, Dropbox, Google talk**, etc.
9. User can view OCR'ed text **history** (5 most recent)
10. When taking new images, users can use phone's **flash** (if applicable)

11. Recognizes **line-breaks**
12. Doesn't save any information on 3rd party systems so **user's data is completely private**
13. Users can **submit a review** of the app through the help screen
14. Direct link to the **website and community** from the app
15. **Dedicated lightweight** application without unnecessary extra features that use up resources.
16. Supported by highest possible percentage of **Android phones**
17. Reasonably high (95+%) and consistent **accuracy** of OCR
18. Reasonably less (a few seconds at most) **response time**
19. Support for a reasonable variety of **font types and sizes on images**
20. Attractive but easy to use **interface**
21. Self-contained application requiring **no internet and data connection charges**
22. Basic **help screen** for detailed usage help

What can I click and what do the buttons do?

1. Main Screen





OCR

The standard icon for OCR throughout the app. A tap on this button will open up the OCR screen.

Settings

Tapping on screen where



this opens up Settings and Preferences you can customize the app



History

Tapping on this button opens up Recent History screen where the 5 most recent text that you have OCR'ed are displayed.



Help/Information

A tap on this will open the Help / Information screen where you can find out more about this app, interact with other users of this app, submit a review, etc.

2. OCR Screen





OCR

A tap opens a dialog where you choose whether you want to select an image from gallery, or use the camera to click a new picture. Once you've selected your image, you can then select and crop the piece of text you're interested in.



Copy To Clipboard

A tap on this icon will copy the OCR'ed text to your device's clipboard



Google the text

Tapping this will open up your default web browser and google search the OCR'ed text for you

Save to SDCard

Tapping on this the folder



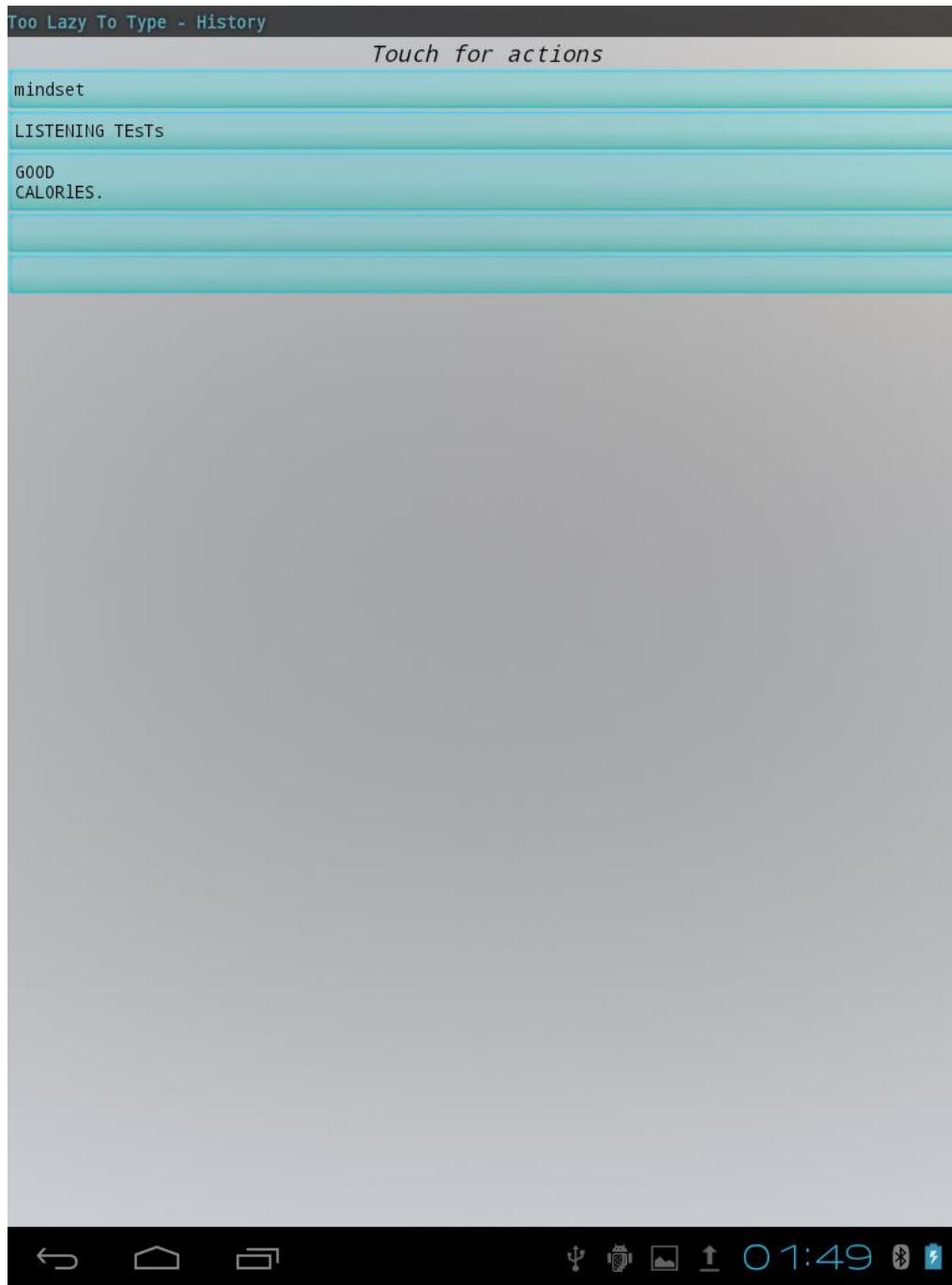
will save the OCR'ed text to your sdcard in OCRNotes. The filename will be displayed as a pop-up message and changes every time you click it. (<number>.txt where <number> is the integer that increments with every save)



Share

A tap on this will open up a dialog from which you can select an app installed on your device to share the text with. For example Twitter, Facebook, Whatsapp, Dropbox, Google+, Bluetooth, Gmail, Messaging, etc

3. History Screen



Tapping any entry will open up a dialog from which you can select an action to perform with the text such as Copy to clipboard, Share, Google it. These do exactly the same actions as on the OCR screen. Two additional actions available only on the history screen are:

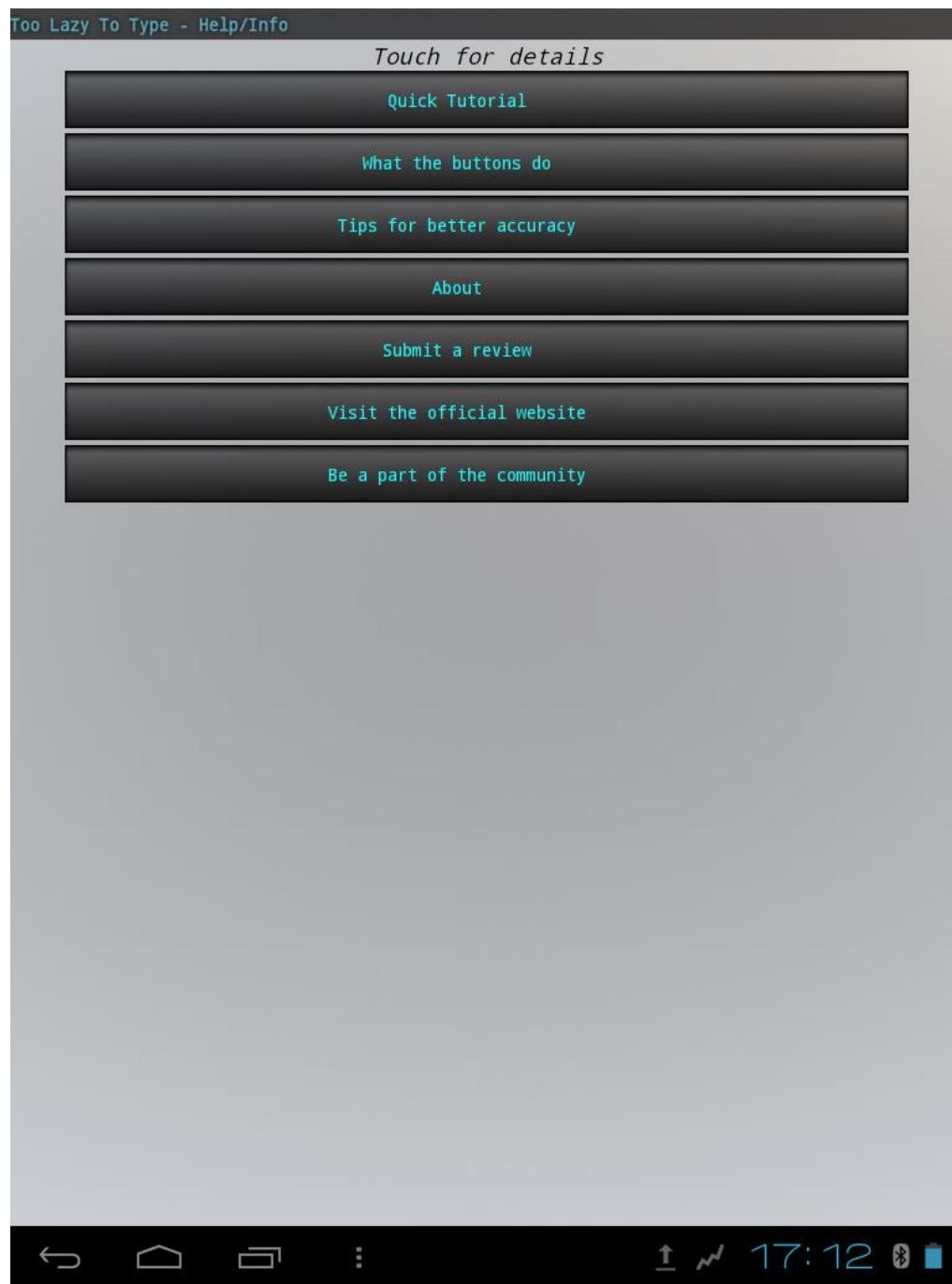
→ Delete

Deletes the entry from recent history. It does not delete the corresponding file on SDCard if you had saved it

→ Directly send an SMS

Prompts you for a phone number, sends the SMS directly, and reports the status as a pop-up. All of this without opening the message app. If you want to select a number from your list of contacts, click on “share” and choose your messaging app. This will open up your default SMS manager.

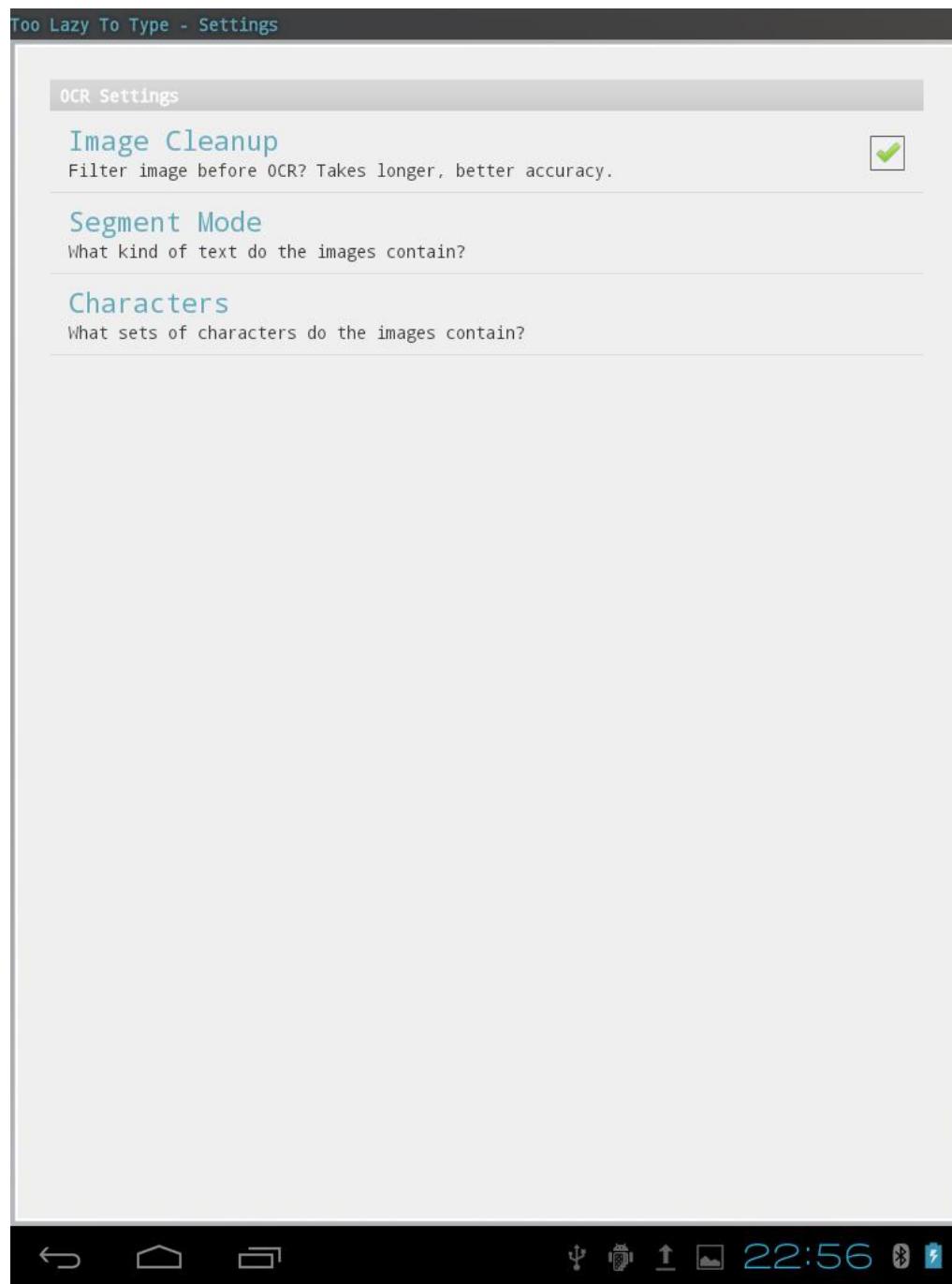
4. Help/Information Screen



- Quick Tutorial
 - Displays the 5 step tutorial to OCR, same as in this book.
- What the buttons do
 - Displays the buttons on the Main and OCR screens and gives a brief description of what each of them do.
- Tips for better accuracy
 - Lists a few tricks that will optimize usage of this app and give you better results.
- About
 - Lists the version number, the developer, and general information about the app
- Submit a review
 - Once you've played around with the app, you can submit a review to let other users know how you liked the app.
 - If you're submitting a review for the first time, it will prompt you to enter an alias. This can be your name, your favourite word, or anything you like as long as it consists of alphabets only. Subsequent submissions of reviews will not open this dialog. The name you entered the first time will be used for every review you submit after that.
 - The app will prompt you to enter a review/feedback.
 - The reviews are submitted to the website so submission will require an Internet connection.
 - You can view your and other people's reviews on the website.
- Visit the official website
 - Opens up the website (meenakshi-ocr.appspot.com) in your default web browser.
- Be a part of the community

- Opens up the community forum in your default web browser. Here, you can interact with other people using this app, post questions, ask for help, share your experience, chit-chat, etc.

5. Preferences / Settings screen



→ Image clean-up

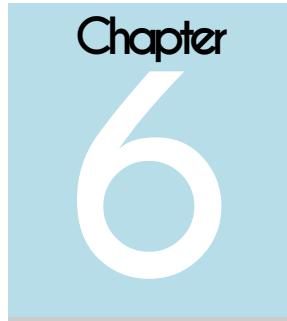
- If checked/ticked, the app will perform pre-processing on your image to remove noise, clean up the background, etc to help you get better results and accuracy.
- This may mean that the app takes slightly longer to process and display the text, but the accuracy will be better.
- Unless your images are simple images with a white background and clear black text, or you want your text faster, you should check this option.

→ Segment Mode

- You can leave this as automatic, or you can specify exactly what kind of images you intend to feed the app. For example, if you usually only select a single word for OCR, you can select the "Single Word" option from the menu that is displayed for this option, and remember to crop only the word you're interested in.

→ Characters

- No big harm will be done if you leave this as automatic, but you can specify if your images contain only numbers, only alphabets, etc.



Tips for optimal usage

These are just some simple tricks and tips that will get you better accuracy and quicker results.

- Avoid blurry images
- Avoid tiny font
- Use better lighting
- Don't use images that are too big
- Avoid cluttered backgrounds
- Leave some empty space around the text while cropping
- This app is not meant to recognize hand-written text and/or very cursive text
- Unless your image is black/white with no noise, and/or you are pressed for time, check the "Pre-Processing" option under Settings. This will increase the accuracy
- Although the app deskews the image for you (if you selected "pre-processing" in Settings), try to align the image nicely.
- The application learns as you feed it more and more images.

Getting help

If you ever find yourself stuck on a feature of the app, or if the app has had some sort of failure/crash/error, help is just a few clicks away.

→ Help with functioning of the app

- The Help/Info screen contains plenty of information to help you with the UI and functionality of this app.
- Additionally, you can always go on to the forum (Main Screen → Help/Info → Be a part of the community) and ask questions. Either I or another fellow user of this app will then be able to help you with your issue.

→ Application crash

- In the unlikely event of an application crash, try running the app again. The crash may have been caused by another application running or some external problem.
- You may have accidentally deleted important files needed for this app to run. Try uninstalling and re-installing the app. This will copy back all the important files and your app will work again.
- If the problem persists, visit meenakshi-ocr-fofou.appspot.com/Too-Lazy-To-Type and post about your problem. Be as descriptive as possible. This will help us solve your problem.

- Before downloading the app, make sure your device meets the requirements for using the app. Minimum version required is 2.2 Froyo
- Sometimes the app will not work with certain images. It will display a pop-up in such an event. The image is probably too big for your device to perform complex processing on, or there may be another reason for this. Try again with another image and it should work fine.

Chapter
8

The website

This app has a dedicated website for all things **Too Lazy To Type**. Go ahead and type meenakshi-ocr.appspot.com into your browser or use the button on your help/info screen on the app (Main screen → Help/Info → Visit the official website)

On this website you will find links to download the app, all reviews submitted by people using this app, developer tools (for fellow developers or students looking to play around with the code), the forum, latest updates, the user manual, the facebook page, etc. You can also “Like” us on facebook.

Bibliography

- ★ www.coursera.org
- ★ www.udacity.com
- ★ developer.android.com
- ★ stackoverflow.com
- ★ code.google.com/p/tesseract-ocr/
- ★ github.com/puelocesar/android-lib-magick
- ★ http://kurup87.blogspot.in/2012/03/android-ocr-tutorial-image-to-text.html
- ★ vietocr.sourceforge.net/training.html
- ★ developers.google.com/appengine/
- ★ blog.kowalczyk.info/software/fofou/
- ★ fancyapps.com/fancybox/
- ★ www.objectaid.com/
- ★ marketplace.eclipse.org/content/modelgoon-uml4java#.UTMqJjBDS7W
- ★ plantuml.sourceforge.net/
- ★ mobiforge.com/developing/story/sms-messaging-android
- ★ yuml.me/