**1. INTRODUCTION**

**1.1 OVERVIEW**

The next generation open operating systems are not on desktops or mainframes but on the small mobile devices people carry every day. The openness of these new environments leads to new applications & markets and enables greater integration. As the demand grows for mobile phone applications, research in optical character recognition, a technology well developed for scanned documents, is shifting focus to the recognition of text embedded in digital photographs. Optical character recognition (OCR) is a powerful tool for bringing information from our analogy lives into the increasingly digital world. This technology has long seen use in building digital libraries, recognizing text from natural scenes, understanding hand-written office forms etc.

By 2014 mobile internet usage should take over Desktop internet usage. Google’s approach is to develop an operating system which can run on every mobile device and not for their specific mobile devices itself, enables them to reach as many people as possible.

Every day a Smartphone user may look for a new application dedicated for his need. Android makes it easier for consumers to get and use new content and applications on their Smart phones. Our project presents an extremely on-demand, fast and user friendly Android Application. This application is useful for native Tourists and Travelers who possess Android Smart phones. This Application has an advanced search feature so that recognized as well as displayed text can be used to copy, paste, share and search for travel related queries like museums, places, restaurants, books, culture, hotels, etc. There is no remote computing overhead because the application has built in OCR suite as well as Image Processing suite both installed in the Android device. It provides fast, robust and extremely high Quality performance because of having improved Auto focus behavior, continuous dynamic preview and improved noise tolerance feature. The motivation of a real time text translation mobile application is to help tourists navigate in a foreign language environment. The application we developed enables the users to get text translate as ease as a button click. The camera captures the text and returns the translated result in real time.

**1.2 PURPOSE**

OCR the native country language Books pages, Signboards, Banners and hotel menus etc. Translate the Recognized text into one of 4-5 languages. This application enables people to understand any native country language. The proposed application would have image processing suite as well as OCR[6] engine both installed on the smart phone implying no server communication overhead and quick and far better processing thus would not just only tackle the limited processing power and limited memory challenge but also provide improved performance.

**1.3 PROBLEM STATEMENT**

The problem here is for the software systems to recognize characters in computer system when information is scanned through paper documents as we know that we have number of newspapers and books which are in printed format related to different subjects. Whenever we scan the documents through the scanner, the documents are stored as ***images*** such as jpeg, gif etc., in the computer system. These images cannot be read or edited by the user. But to reuse this information it is very difficult to read the individual contents and searching the contents form these documents line-by-line and word-by-word. These days there is a huge demand in “storing the information available in these paper documents in to a computer storage disk and then later editing or reusing this information by searching process”.

**1.4 PROJECT SCOPE**

Every day a Smartphone user may look for a new application dedicated for his need. Android makes it easier for consumers to get and use new content and applications on their Smart phones. This paper presents an extremely on-demand, fast and user friendly Android Application ATMA.

ATMA stands for Android Travel Mate Application. This application is useful for native Tourists and Travelers who possess Android Smart phones. It enables Travelers and Tourists to easily capture the native country language Books pages, signboards, banners and hotel menus etc. The built-in OCR converts the text embedded in the captured image into Unicode text format. It also provides translation facility so that Tourists can translate the Native Language Unicode text into their own country language.

This Application has an advanced search feature so that recognized as well as translated text can be used to copy, paste, share and search for travel related queries like museums, places, restaurants, books, culture, hotels, etc.There is no remote computing overhead because the application has built in OCR suite as well as Image Processing suite both installed in the Android device. It provides fast, robust and extremely high Quality performance because of having improved Auto focus behavior, continuous dynamic preview and improved noise tolerance feature.

**1.5 EXISTING SYSTEM**

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

**1.6 DRAWBACK OF EXISTING SYSTEM**

The drawback in the early OCR systems is that they only have the capability to convert and recognize only the documents of English or a specific language only. That is, the older OCR system is uni-lingual.

**1.7 PROPOSED SYSTEM**

The project enables Travelers and Tourists to easily capture the native country language Books pages, signboards, banners and hotel menus etc. The built-in OCR converts the text embedded in the captured image into Unicode text format. It also provides translation facility so that Tourists can translate the Native Language Unicode text into their own country language. This Application has an advanced search feature so The project enables Travelers and Tourists to easily capture the native country language Books pages, signboards, banners and hotel menus etc.

The built-in OCR converts the text embedded in the captured image into Unicode text format. It also provides translation facility so that Tourists can translate the Native Language Unicode text into their own country language. This Application has an advanced search feature so that recognized as well as translated text can be used to copy, paste, share and search for travel related queries like museums, places, restaurants, books, culture, hotels, etc. This would prove enormously beneficial with respect to the aspects about localization being a common phenomenon now-a-days. Also android platform has been increasingly being common in accordance with its features like low-cost, customizable, lightweight operating system and more. In this module, The Binarization of Captured Image takes place, after that the text layout is analyzed, Blobs are detected and finally words and lines are detected. The words are sent to a number of passes. In these passes each word is chopped into characters and characters are checked for the need of joining the broken characters or the breaking of associated characters. Finally chopped characters are recognized with the help of inbuilt fuzzy features matched to language specific training.

**1.8 BENEFITS OF PROPOSED SYSTEM**

The benefit of proposed system that overcomes the drawback of the existing system is that it supports multiple functionalities such as editing and searching. It also adds benefit by providing heterogeneous characters recognition.

Save data entry costs - automatic recognition by OCR/ICR/OMR/barcode engines ensure lower manpower costs for data entry and validation. Lower licensing cost - since the product enables distributed capture licensing costs for OCR/ICR engine is much lower. For instance 5 workstations may be used for scanning and indexing but only one OCR/ICR license may be required. Export the recognized data in XML or any other standard format for integration with any application or database.

**APPLICATIONS**

* Industries and Institutions in which control of large amounts of paper work is critical
* Banking, Credit cards, Insurance industries Libraries and archives.
* To read the signs for travelers and tourists travelling in another country.

For conservation and preservation of vulnerable documents and for the provision of access to source documents OCR fonts are used for several purpose where automated systems needs a several purposes where automated systems need a standardcharacter shape defined to properly read text without the use of barcodes. Some examples of OCR font implementations include bank checks, passports, serial labels and postal mail.

**1.9 PLATFORM SPECIFICATION**

**1.9.1 HARDWARE**

* Processor : DUAL CORE
* RAM : 512 MB (Minimum)
* Interface : Touch Screen.

**1.9.2 SOFTWARE**

* Operating System – Android 3.0 or above
* Database – SQLite
* IDE – Eclipse

**1.9.3 TECHNOLOGY USED**

**JAVA:** Java is an object-oriented programming language developed by Sun Microsystems a company best known for its high end UNIX workstations. Java language was designed to be small, simple and portable across platforms, operating systems, both at the source and at the binary level, which means that Java programs (applet and application) can run on any machine that has the Java virtual machine (JVM) installed.

**XML:** XML [code](http://whatis.techtarget.com/definition/code), a formal recommendation from the [World Wide Web Consortium](http://searchsoa.techtarget.com/definition/W3C) (W3C), is similar to [Hypertext Markup Language](http://searchsoa.techtarget.com/definition/HTML) (HTML). Both XML and HTML contain [markup symbols](http://searchsoa.techtarget.com/definition/markup) to describe page or file contents. HTML code describes Web page content (mainly text and graphic images) only in terms of how it is to be displayed and interacted with.XML data is known as self-describing or self-defining, meaning that the structure of the data is embedded with the data, thus when the data arrives there is no need to pre-build the structure to store the data; it is dynamically understood within the XML. The XML format can be used by any individual or group of individuals or companies that want to share information in a consistent way. XML is actually a simpler and easier-to-use subset of the [Standard Generalized Markup Language](http://searchsoa.techtarget.com/definition/SGML) (SGML), which is the standard to create a document structure.

**ANDROID:** Android is a Linux-based operating System designed primarily for touch screen mobile devices such as smart phones and tablet computers. Initially developed by Android Inc. , which Google backed financially and later purchased in 2005. Android is open source and Google releases the code under the Apache License. This open source code and permissive licensing allows the software to be freely modified and distributed by device manufacturers, wireless carriers and enthusiast developers. Additionally, Android has a large community of developers writing applications (“apps”) that extend the functionality of devices, written primarily in a customized version of the Java programming language.

**2. SYSTEM ANALYSIS**

**2.1 IDENTIFICATION OF THE NEED**

Every day a Smartphone user may look for a new application dedicated for his need. Android makes it easier for consumers to get and use new content and applications on their Smart phones. This paper presents an extremely on-demand, fast and user friendly Android Application ATMA. ATMA stands for Android Travel Mate Application. This application is useful for native Tourists and Travelers who possess Android Smart phones. It enables Travelers and Tourists to easily capture the native country language Books pages, signboards, banners and hotel menus etc.

The built-in OCR converts the text embedded in the captured image into Unicode text format. It also provides translation facility so that Tourists can translate the Native Language Unicode text into their own country language. This Application has an advanced search feature so that recognized as well as translated text can be used to copy, paste, share and search for travel related queries like museums, places, restaurants, books, culture, hotels, etc. There is no remote computing overhead because the application has built in OCR suite as well as Image Processing suite both installed in the Android device. It provides fast, robust and extremely high Quality performance because of having improved Auto focus behavior, continuous dynamic preview and improved noise tolerance feature.

**2.2 PRELIMINARY INVESTIGATION**

The Optical Character Recognition is a mobile application. It uses smart mobile phones of android platform. This paper combines the functionality of Optical Character Recognition and speech synthesizer. The objective is to develop user friendly application which performs image to speech conversion system using android phones. The OCR takes image as the input, gets text from that image and then converts it into speech. This system can be useful in various applications like banking, legal industry, other industries, and home and office automation. It mainly designed for people who are unable to read any type of text documents. In this paper, the character recognition method is presented by using OCR technology and android phone with higher quality camera.

**2.3 FEASIBILITY SERVICE**

**2.3.1 TECHNICAL FEASIBILITY**

This application will not be too complex in terms of technology as it would be used by tourist and travelers who might have difficulties to understand the complex application. The design of application is very easy so that the user can easily understand the interface. User must only open the application and click the text they want to recognize so that they can understand the written image in their language. The user must carry an android phone that should be network enabled and should be properly charged before using the application.

**2.3.2 ECONOMICAL FEASIBILITY**

Economic feasibility attempts to weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system. A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

**2.3.3 OPERATIONAL FEASIBILITY**

The application is developed for travelers and tourist so they would not have any difficulty in operating the application. The user interface of the application has been kept simple so that it is easy to use even for the first timers.

The application can be of great use while travelling to any other place other than their home country or home town.

**2.4 FUNCTIONAL REQUIREMENTS**

Functional requirements specify all the Functions performed by any System. It includes both Hardware and Software requirements which are as follows:

**SOFTWARE REQUIREMENTS**

* Java(JDK 1.6)
* Android SDK 4.0
* IDE :Eclipse Helios

**HARDWARE REQUIREMENTS**

* Dual Core
* 512 MB RAM

**JAVA:** Java is an object-oriented programming language developed by Sun Microsystems a company best known for its high end UNIX workstations. Java language was designed to be small, simple and portable across platforms, operating systems, both at the source and at the binary level, which means that Java programs (applet and application) can run on any machine that has the Java virtual machine (JVM) installed.

**ANDROID:** Android is a Linux-based operating System designed primarily for touch screen mobile devices such as smart phones and tablet computers. Initially developed by Android Inc. which Google backed financially and later purchased in 2005. Android is open source and Google releases the code under the Apache License. This open source code and permissive licensing allows the software to be freely modified and distributed by device manufacturers, wireless carriers and enthusiast developers. Additionally, Android has a large community of developers writing applications ("apps") that extend the functionality of devices, written primarily in a customized version of the Java programming language.

**ANDROID SDK:** A [software](http://www.webopedia.com/TERM/S/software.html) development kit that enables developers to create [applications](http://www.webopedia.com/TERM/A/application.html) for the [Android platform](http://www.webopedia.com/TERM/A/Android_platform.html). The Android [SDK](http://www.webopedia.com/TERM/S/SDK.html) includes sample projects with [source code](http://www.webopedia.com/TERM/S/source_code.html), development tools, an [emulator](http://www.webopedia.com/TERM/E/emulator.html), and required libraries to build Android applications. Applications are written using the [Java](http://www.webopedia.com/TERM/J/Java.html) programming language and run on [Dalvik](http://www.webopedia.com/TERM/D/Dalvik.html), a custom [virtual machine](http://www.webopedia.com/TERM/V/virtual_machine.html) designed for embedded use which runs on top of a [Linux](http://www.webopedia.com/TERM/L/Linux.html) kernel. Android software development is the process by which new applications are created for the Android operating system. Applications are usually developed in the Java programming language using the Android Software Development Kit, but other development tools are available.

**ECLIPSE:** Eclipse is a multi-language software development environment comprising a base workspace and an extensible plug-in system for customizing the environment. It is written mostly in Java. It can be used to develop applications in Java and, by means of various plug-ins, other programming languages including Ada, C, C++, COBOL, Fortran, Haskell, Perl, PHP, Python, Ruby (including Ruby on Rails framework), and Groovy. It can also be used to develop packages for the software Mathematical. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others. The initial codebase originated from IBM Visual Age. The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

**2.4.1 PRODUCT FUNCTIONS**

Below mentioned are the functionalities provided by the system:

* Image Processing Module.
* Document Recognition Module.

**IMAGE PROCESSING MODULE**

This module is accessed by User whose role in our application is a librarian.This module perform certain activities such as scanning documents, storing them as images, recognizing characters in images to transfer them into word format. During the recognition process, this module uses the OCR methodology in support of grid infrastructure datastructure. The module supports the following services:-

* Storing the documents as snapshots or images.
* Processing those image-based documents.
* Recognizing the characters in documents.

.

**IMAGE RECOGNITION MODULE**

This module can be accessed by both the User and the end-user. Once the printed documents are converted into structured documents, any user can recognize the characters present in the document. That means the user can recognize the characters of any language he chooses which makes OCR more flexible. This flexibility is due to the adaptation of grid infrastructure. This is the module where the main functionality of OCR is tested.

Under this module, there are two types of recognition. They are ***handwritten recogniiton*** and ***scanned document recognition.***

In handwritten recognition, the handwriting of the user in any language is trained to the system only for the first time. From there on-wards, the system recognizes the characters or words written by the user. Thus handwritten document recognition recognizes the human handwriting.

In scanned document recognition, the system is first trained with the font characters in the document in the training module itself. Now in the recognition module, the system takes the scanned documents image as an input file, first crops the image and then extracts/recognizes the characters from the document and makes these documents editable and searchable. Thus the scanned document recognition recognizes the chracters from the scanned document image and makes the document editable and searchable. Hence the document recogniiton module on a whole supports the following services:-

* Recognizes the characters
* Heterogeneous character Recognition.

# 2.5 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define the overall qualities or attributes of the resulting system. Non-functional requirements place restrictions on the product being developed, the development process, and specify external constraints that the product must meet. Examples of NFR include safety, security, usability, reliability and performance requirements.

**2.5.1 PERFORMANCE REQUIREMENTS**

Performance requirements concern the speed of operationof a System. This project performs well and is running. These are the requirement which will further improve the performance it may include to deduce the response time and avoid unnecessary deterring effect .Some Important Requirement which will further enhance the functionality of app. Some performance requirements are mention below:

* **Instant information:** If information is not provided instantly then the entire app’s performance would be deterring. As we need this service to get the information about the text from the image that is captured by the user.

* **Version Of android:** Higher the version of android (3.0) better would be the performance of the app.

## 2.5.2 SAFETY REQUIREMENTS

Safety requirements are the ‘shall not’ requirements which exclude unsafe situations from the possible solution space of the system. These include the requirement which if got deter while functioning of the app the task would remain incomplete or would not be per sued for and if such things happen what precaution should be taken.

**2.5.3 SEQURITY REQUIREMENTS**

Security is the ability of the software to remain protected from unauthorized access. This includes both change access and view access. The App is well secured using multiple levels of Security Constraints.

**2.5.4 RELIABILITY**

Reliability is the ability of a system to perform its required functions under stated conditions for a specific period of time. High Reliability is the measure of how a product behaves in varying circumstances.

**2.5.5 RESPONSE TIME**

Response time is the total amount of time it takes to respond to a request for service. That service can be anything from a memory fetch, to a disk IO, to a complex database query, or loading a full web page, ignoring transmission time for a moment.

**2.5.6 ROBUSTNESS**

Robustness is defined as the ability of a software product to cope with unusual situation.

**2.5.7 SCALABILITY**

Scalability is the ability of a system, network, or process to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth.

**2.5.8 SECURITY**

Security is the ability of the software to remain protected from unauthorized access. This includes both change access and view access.

**2.5.9 STABILITY**

Stability defines the quality or State of something that is not easily changed or likely to change.

**2.5.10 SUPPORTABILITY**

Supportability is the measure of the extent to which software is composed of separate, interchangeable components, each of which accomplishes one function and contains everything necessary to accomplish this. Supportability increases cohesion and reduces coupling and makes it easier to extend the functionality and maintain the code.

**2.5.11 TESTABILITY**

Testability is the ability of software to be tested thoroughly before putting into production. Although this app does internal testing before releasing any new versions, they can never be sure to work at an Operator's production systems without testing. This gets very useful in testing the new releases before production.

**2.5.12 FAULT TOLERANCE**

Fault tolerance is the property that enables a system to continue operating properly in the event of the failure of some of its components. Fault tolerance is the way in which an operating system (OS) responds to a hardware or software failure.

**3. SOFTWARE DESIGN**

**3.1 UML DIAGRAMS**

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has 14 types of diagrams divided into two categories. Seven diagram types represent structural information, and the other seven represent general types of behavior, including four that represent different aspects of interactions. Some of these diagrams we provided to describe the design and implementation of our OCR system can be categorized hierarchically as below:-

* Use case diagram
* Class diagram
* Sequence diagram
* Activity diagram
* Component diagram
* Deployment diagram

**3.1.1 USE CASE DIAGRAM**

Our software system can be used to support library environment to create a *Digital Library* where several paper documents are converted into electronic-form for accessing by the users. For this purpose the printed documents must be recognized before they are converted into electronic-form. The resulting electronic-documents are accessed by the users like faculty and students for reading and editing. Now according to this information, the following are the different actors involved in implementing our OCR system:-

1. If we consider for virtual digital library, the User can be the Librarian and
2. The End-users can be Students or/and Faculty.

The following are the list of use diagrams that altogether form the complete or the overall use-case diagram. They are listed below:-

1. Use-case diagram.

In the use-case diagrams below we clearly explained about that particular use-case functionality. In this we provided a description about the

* Use-case name
* Details about the use-case
* Actors using this use-case
* The flow of events carried out by the use-case
* The conditions that occur in this use-case.

**Description**

The User is the only person who participates in the image processing. Here he captures the images. The captured images are read by the system. Finally the read images are recognized and can copy or share the text.

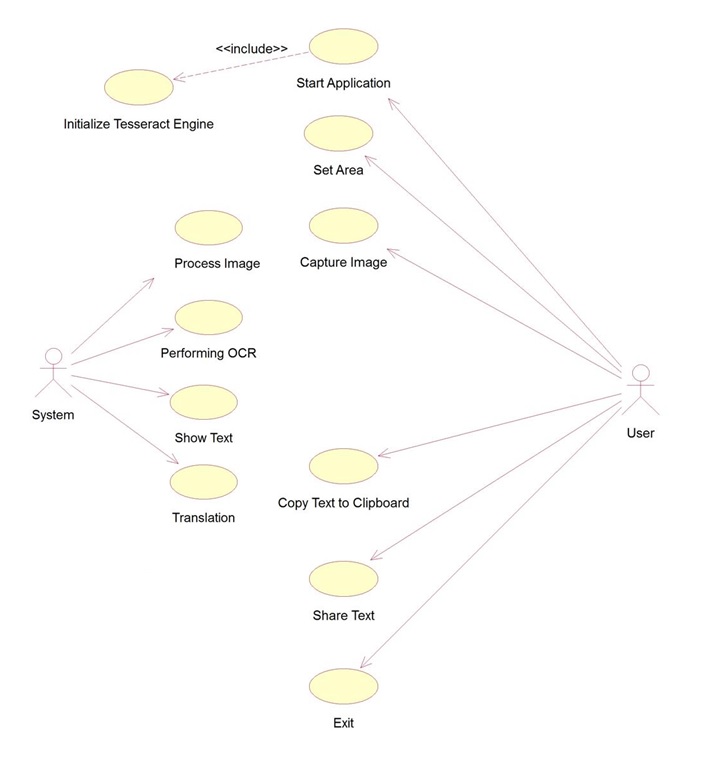
The System comes in role after the image is captured by the user. The System processes the image to generate the text from the image given by the user. The text is generated with the help of tesseract Engine and thus can be either copied or shared to by the user.

**Actors**

* **Primary Actor :** User
* **Secondary Actor :** System

**Flow of Events**

1. User starts the application.
2. As soon as the application is started, the tesseract engine is initialized.
3. User sets the area of the box in which he has to capture the image.
4. System processes the image captured by the user.
5. System performs the OCR of the given image.
6. After performing OCR, System displays the text recognized by it.
7. Translation of the text can be done if required.
8. User captures the image which he wants to read.
9. User has two options, either he can copy the text or share the text in other application.



**Figure.3.1**

**3.1.2 SEQUENCE DIAGRAM**

Sequence diagrams are sometimes called Event-trace diagrams, event scenarios, and timing diagrams. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

In sequence diagram, the class objects that are used to describe the interaction between various classes vary from one function to another function. There are five sequence diagrams short-listed below for presenting the sequence of actions performed by each of the five modules. The key class object involved in all of these module functions is Main Screen class which controls the interaction among various class objects.

**Sequence Diagram for Image Processing**

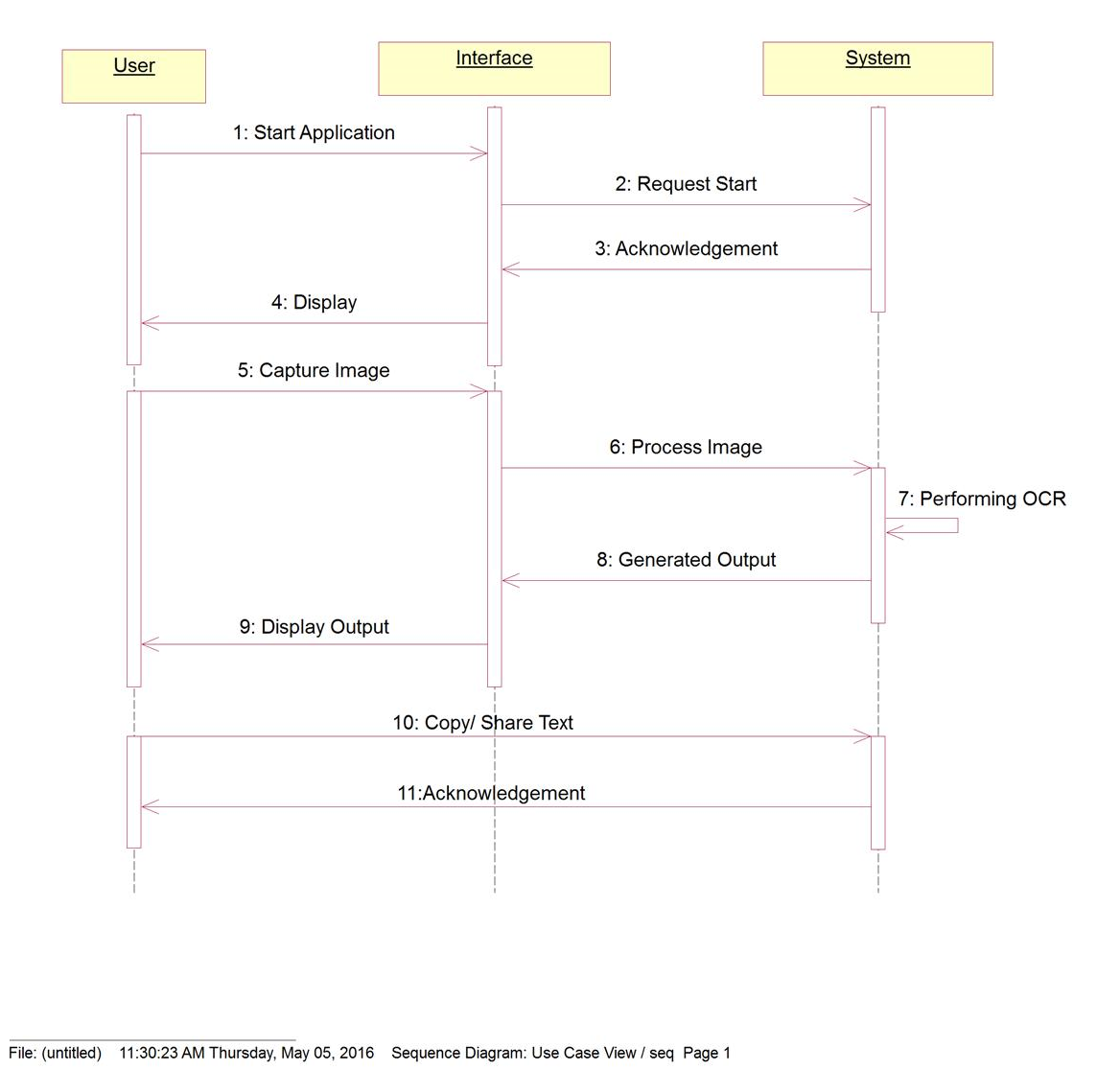
1. **Objects**

User- “u”

Interface- “i”

System - “s”

1. **Links**
2. Interface object to User object.
3. User object to Interface object.
4. Interface object to System object.
5. System object to Interface object.
6. Interface object to User object.
7. User object to System object.
8. **Messages**
9. Display
10. Capture Image
11. Process Image
12. Performing OCR.
13. Generated Output
14. Display Output
15. Copy/ Share text.



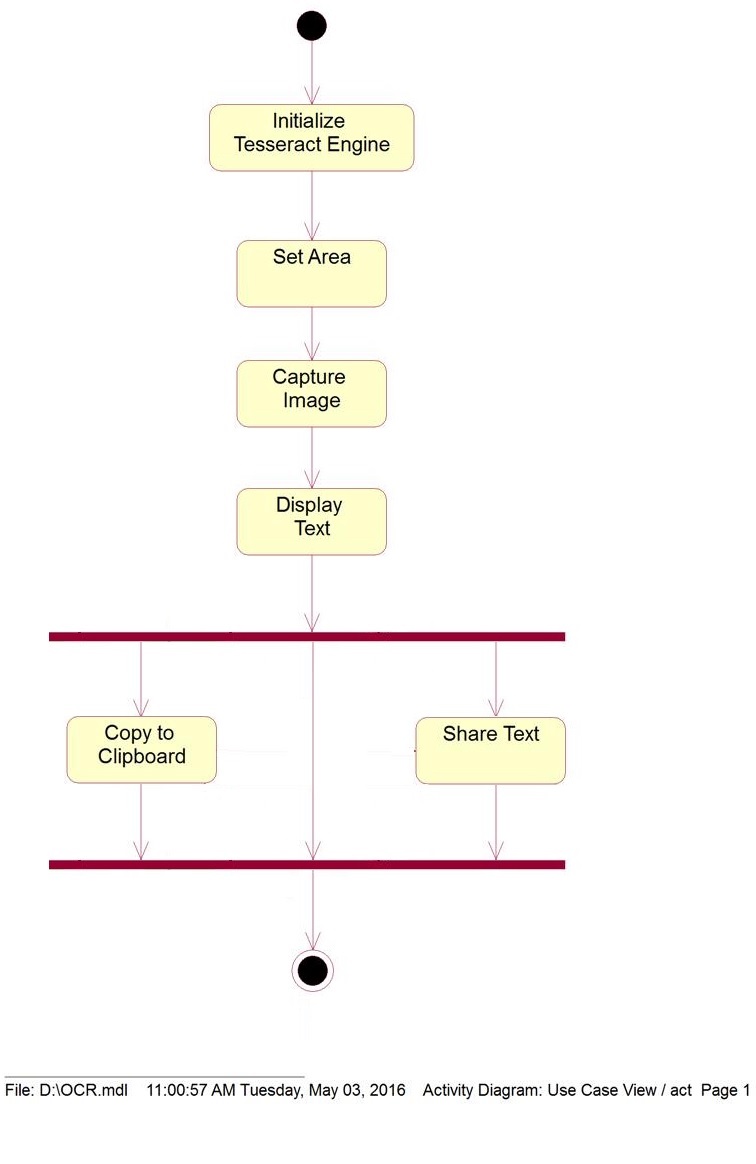
**Figure 3.2**

**3.1.3 ACTIVITY DIAGRAM**

The purpose of activity diagram is to provide a view of flows and what is going on inside a use case or among several classes. Activity diagram can also be used to represent a class’s method implementation. A token represents an operation. An activity is shown as a round box containing the name of the operation. An outgoing solid arrow attached to the end of activity symbol indicates a transition triggered by the completion.

Activity diagrams provide a way to model the workflow of a business process. Activity diagrams can also be used to model code-specific information, such as a class operation. Activity diagrams are very similar to a flowchart because of modeling a workflow from activity to activity. An activity diagram is basically a special case of a state machine in which most of the states are activities and most of the transitions are implicitly triggered by completion of the actions in the source activities.

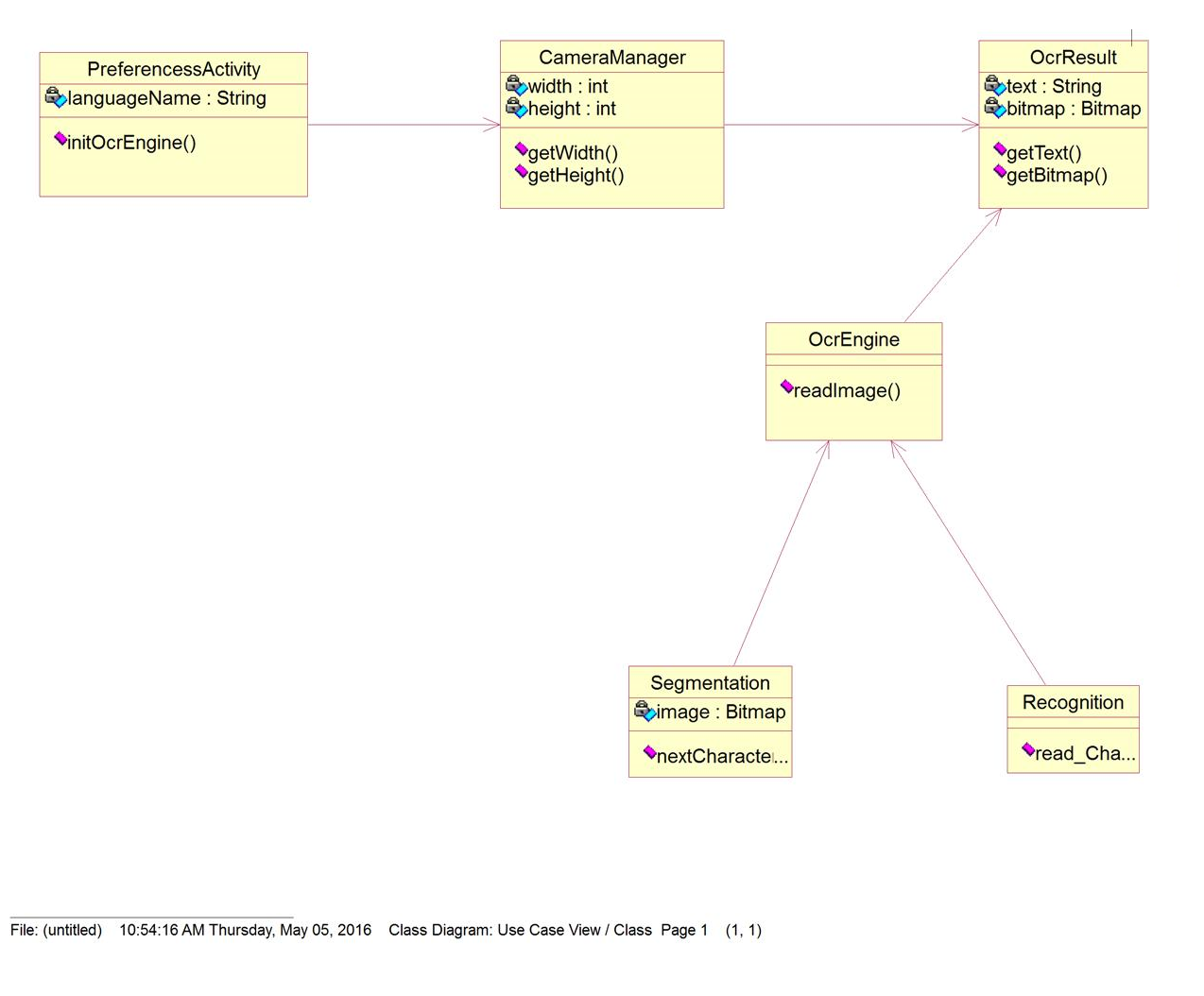
Each activity (Figure 3.4) represents the performance of a group of actions in a workflow. Once the activity is complete, the flow of control moves to the next activity or state through a transition. If an outgoing transition is not clearly triggered by an event, then it is triggered by the completion of the contained actions inside the activity. A unique activity diagram feature is a swim lane that defines who or what is responsible for carrying out the activity or state. It is also possible to place objects on activity diagrams. The workflow stops when a transition reaches an end state. It is possible to attach activity diagrams to most model elements in the use case or logical views. Activity diagrams cannot reside within the component view.

****

**Figure 3.3**

**3.1.4 CLASS DIAGRAM**

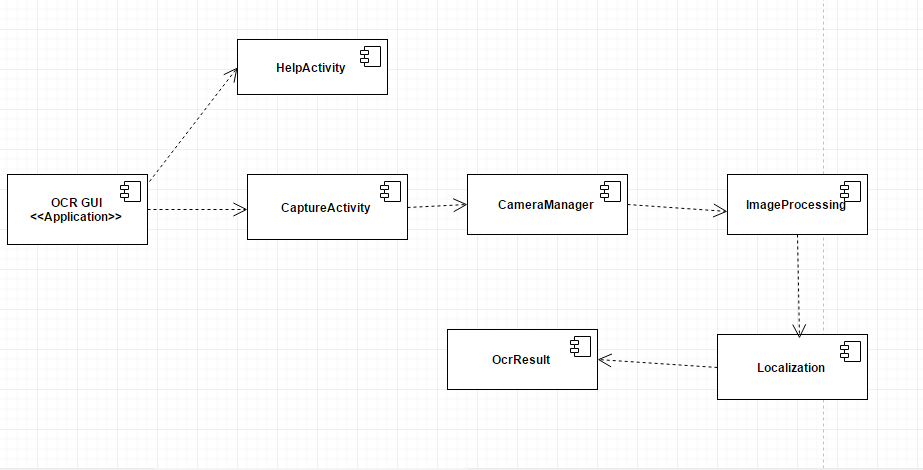
The class diagram (Figure 3.6) is the main building block of [object oriented](https://en.wikipedia.org/wiki/Object_oriented) modeling. It is used both for general [conceptual modeling](https://en.wikipedia.org/wiki/Conceptual_model) of the systematic of the application, and for detailed modeling translating the models into [programming code](https://en.wikipedia.org/wiki/Programming_code). Class diagrams can also be used for [data modeling](https://en.wikipedia.org/wiki/Data_modeling). The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

****

**Figure 3.4**

**3.1.5 COMPONENT DIAGRAM**

The crucial component in our component diagram(figure 3.8) that plays a major role in implementing the OCR system is the GUI component. All other components that is Document processing and recognition, Document editing and Document Searching depends on it. They are as follows:-

****

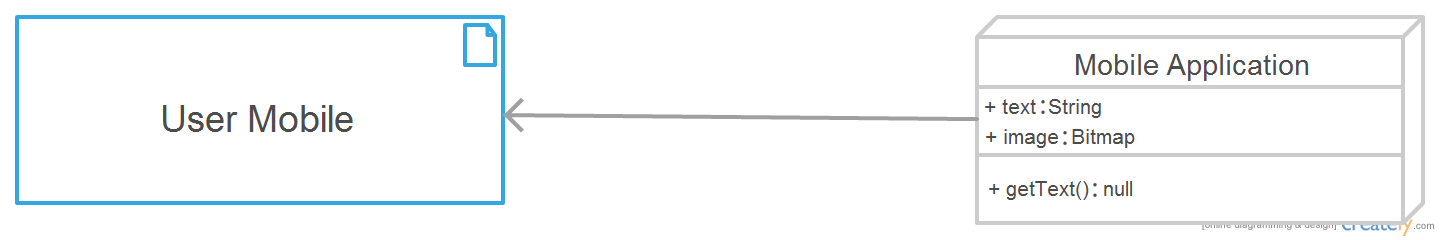
**Figure 3.5**

**3.1.6 DEPLOYMENT DIAGRAM**

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.

So deployment diagrams (Figure 3.9) are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

The name *Deployment* itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related.

****

**Figure 3.6**

**4. SOFTWARE ENGINEERING APPROACH**

**4.1 SOFTWARE ENGINEERING PARADIGM APPLIED**

**4.1.1 DESCRIPTION**

The apparent purpose of software life cycle models was to provide a conceptual scheme for rationally managing the development of software systems. Such a scheme could therefore serve as a basis for planning, organizing, staffing, coordinating, budgeting, and directing software development activities. In our project “Development of a noval real time Bus Tracking System.” we are using incremental model.

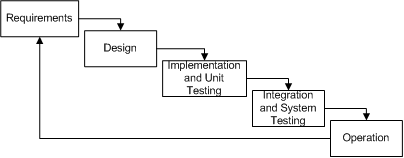
Incremental model (Figure 4.1) is an evolution of waterfall model. The product is designed, implemented, integrated and tested as a series of incremental builds. It is a popular model software evolution used many commercial software companies and system vendor. Incremental software development model may be applicable to projects where:

* + Software Requirements are well defined, but realization may be delayed.
  + The basic software functionalities are required early.

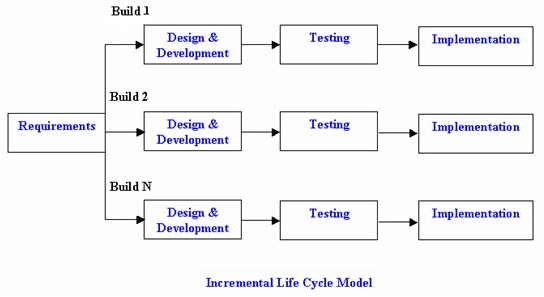
The incremental build model (Figure 4.2) is a method of software development where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements. The product is decomposed into a number of components, each of which are designed and built separately (termed as builds). Each component is delivered to the client when it is complete.

This allows partial utilization of product and avoids a long development time. It also creates a large initial capital outlay with the subsequent long wait avoided. This model of development also helps ease the traumatic effect of introducing completely new system all at once.

There are some problems with this model. One is that each new build must be integrated with previous builds and any existing systems. The task of decomposing product into builds is not trivial either. If there are few builds and each build degenerates this turns into Build-And-Fix model. However if there are too many builds then there is little added utility from each build.

****

**Figure 4.1: Incremental Life Cycle Model**

****

**Figure 4.2: Adapted Incremental Model**

**4.1.2 ADVANTAGES AND DISADVANTAGES**

**Advantages of Incremental model**

* Generates working software quickly and early during the software life cycle.
* This model is more flexible – less costly to change scope and requirements.
* It is easier to test and debug during a smaller iteration.
* In this model customer can respond to each built.
* Lowers initial delivery cost.
* Easier to manage risk because risky pieces are identified and handled during it’d iteration.

**Disadvantages of Incremental model**

* Needs good planning and design.
* Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
* Total cost is higher than [waterfall](http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/).

**4.1.3 REASON FOR USE**

* This model can be used when the requirements of the complete system are clearly defined and understood.
* Major requirements must be defined; however, some details can evolve with time.
* There is a need to get a product to the market early.
* A new technology is being used
* Resources with needed skill set are not available
* There are some high risk features and goals.

**4.2 REQIUREMENT ANALYSIS**

This project is aimed at developing fully integrated mobile-based Application. A common environment can be developed for accessing and exchanging the placement related information.

**4.2.1 INTENDED USERS**

* **Developer:** A software developer is a person concerned with facts of the software development process. In short, developers "make software for the world to use." Their work includes researching, designing, implementing, and testing software.
* **Project Manager:** A project manager is a professional in the field of project management. Project managers can have the responsibility of the planning, execution and closing of any project.
* **Users:** All the School Going Students who wait for bus in their respective Bus Stops.

### Tester: A technician who conducts prescribed tests on software programs and applications prior to their implementation to ensure quality, design, integrity and proper functionality. They apply right testing methods including extensive end-user simulations to uncover program “bugs” which are then eliminated by programmers.

### 4.2.2 PROBLEM SPECIFICATION

Present system does not allow user to get any notification of the insurance date, renewal, fixed deposits etc. People get difficulties in gathering all the information at the same time at the same place.

**4.2.3 SYSTEM FEATURE**

The android app helps users to get notifications about their last date of bus registration, renewals details, banking information like, last date of insurance payments and fixed deposits information. After installing this app on android mobile phone notifications are displayed on status bar for above mentioned features.

**4.2.4 PURPOSE**

The application is a user friendly one that anyone can access. The basic idea for this application is to guide the bus travelers with the timings of the last stops where the bus has been so that they can estimate at what time the bus will reach their stop. The aim is to overcome all the drawbacks faced in all the previous applications and generate fast and accurate results.

**4.3 IMPLEMENTATION PHASE**

**4.3.1 CODING**

1. **CameraManager.java**

**package** edu.sfsu.cs.orange.ocr.camera;

**import** android.content.Context;

**import** android.content.SharedPreferences;

**import** android.graphics.Point;

**import** android.graphics.Rect;

**import** android.hardware.Camera;

**import** android.os.Handler;

**import** android.preference.PreferenceManager;

**import** android.util.Log;

**import** android.view.SurfaceHolder;

**import** edu.sfsu.cs.orange.ocr.PlanarYUVLuminanceSource;

**import** edu.sfsu.cs.orange.ocr.PreferencesActivity;

**import** java.io.IOException;

**public final class** CameraManager {

**private static final** String ***TAG*** = CameraManager.**class**.getSimpleName();

**private static final int *MIN\_FRAME\_WIDTH*** = 50; *// originally 240*

**private static final int *MIN\_FRAME\_HEIGHT*** = 20; *// originally 240*

**private static final int *MAX\_FRAME\_WIDTH*** = 800; *// originally 480*

**private static final int *MAX\_FRAME\_HEIGHT*** = 600; *// originally 360*

**private final** Context **context**;

**private final** CameraConfigurationManager **configManager**;

**private** Camera **camera**;

**private** AutoFocusManager **autoFocusManager**;

**private** Rect **framingRect**;

**private** Rect **framingRectInPreview**;

**private boolean initialized**;

**private boolean previewing**;

**private boolean reverseImage**;

**private int requestedFramingRectWidth**;

**private int requestedFramingRectHeight**;

**private final** PreviewCallback **previewCallback**;

**public** CameraManager(Context context) {

**this**.**context** = context;

**this**.**configManager** = **new** CameraConfigurationManager(context);

**previewCallback** = **new** PreviewCallback(**configManager**);

}

**public synchronized void** openDriver(SurfaceHolder holder) **throws** IOException {

Camera theCamera = **camera**;

**if** (theCamera == **null**) {

theCamera = Camera.*open*();

**if** (theCamera == **null**) {

**throw new** IOException();

}

**camera** = theCamera;

}

**camera**.setPreviewDisplay(holder);

**if** (!**initialized**) {

**initialized** = **true**;

**configManager**.initFromCameraParameters(theCamera);

**if** (**requestedFramingRectWidth** > 0 && **requestedFramingRectHeight** > 0) {

adjustFramingRect(**requestedFramingRectWidth**, **requestedFramingRectHeight**);

**requestedFramingRectWidth** = 0;

**requestedFramingRectHeight** = 0;

}

}

**configManager**.setDesiredCameraParameters(theCamera);

SharedPreferences prefs = PreferenceManager.*getDefaultSharedPreferences*(**context**);

**reverseImage** = prefs.getBoolean(PreferencesActivity.***KEY\_REVERSE\_IMAGE***, **false**);

}

**public synchronized void** closeDriver() {

**if** (**camera** != **null**) {

**camera**.release();

**camera** = **null**;

**framingRect** = **null**;

**framingRectInPreview** = **null**;

}

}

**public synchronized void** startPreview() {

Camera theCamera = **camera**;

**if** (theCamera != **null** && !**previewing**) {

theCamera.startPreview();

**previewing** = **true**;

**autoFocusManager** = **new** AutoFocusManager(**context**, **camera**);

}

}

**public synchronized void** stopPreview() {

**if** (**autoFocusManager** != **null**) {

**autoFocusManager**.stop();

**autoFocusManager** = **null**;

}

**if** (**camera** != **null** && **previewing**) {

**camera**.stopPreview();

**previewCallback**.setHandler(**null**, 0);

**previewing** = **false**;

}

}

**public synchronized void** requestOcrDecode(Handler handler, **int** message) {

Camera theCamera = **camera**;

**if** (theCamera != **null** && **previewing**) {

**previewCallback**.setHandler(handler, message);

theCamera.setOneShotPreviewCallback(**previewCallback**);

}

}

**public synchronized void** requestAutoFocus(**long** delay) {

**autoFocusManager**.start(delay);

}

**public synchronized** Rect getFramingRect() {

**if** (**framingRect** == **null**) {

**if** (**camera** == **null**) {

**return null**;

}

Point screenResolution = **configManager**.getScreenResolution();

**if** (screenResolution == **null**) {

*// Called early, before init even finished*

**return null**;

}

**int** width = screenResolution.**x** \* 3/5;

**if** (width < ***MIN\_FRAME\_WIDTH***) {

width = ***MIN\_FRAME\_WIDTH***;

} **else if** (width > ***MAX\_FRAME\_WIDTH***) {

width = ***MAX\_FRAME\_WIDTH***;

}

**int** height = screenResolution.**y** \* 1/5;

**if** (height < ***MIN\_FRAME\_HEIGHT***) {

height = ***MIN\_FRAME\_HEIGHT***;

} **else if** (height > ***MAX\_FRAME\_HEIGHT***) {

height = ***MAX\_FRAME\_HEIGHT***;

}

**int** leftOffset = (screenResolution.**x** - width) / 2;

**int** topOffset = (screenResolution.**y** - height) / 2;

**framingRect** = **new** Rect(leftOffset, topOffset, leftOffset + width, topOffset + height);

}

**return framingRect**;

}

**public synchronized** Rect getFramingRectInPreview() {

**if** (**framingRectInPreview** == **null**) {

Rect rect = **new** Rect(getFramingRect());

Point cameraResolution = **configManager**.getCameraResolution();

Point screenResolution = **configManager**.getScreenResolution();

**if** (cameraResolution == **null** || screenResolution == **null**) {

*// Called early, before init even finished*

**return null**;

}

rect.**left** = rect.**left** \* cameraResolution.**x** / screenResolution.**x**;

rect.**right** = rect.**right** \* cameraResolution.**x** / screenResolution.**x**;

rect.**top** = rect.**top** \* cameraResolution.**y** / screenResolution.**y**;

rect.**bottom** = rect.**bottom** \* cameraResolution.**y** / screenResolution.**y**;

**framingRectInPreview** = rect;

}

**return framingRectInPreview**;

}

**public synchronized void** adjustFramingRect(**int** deltaWidth, **int** deltaHeight) {

**if** (**initialized**) {

Point screenResolution = **configManager**.getScreenResolution();

*// Set maximum and minimum sizes*

**if** ((**framingRect**.width() + deltaWidth > screenResolution.**x** - 4) || (**framingRect**.width() + deltaWidth < 50)) {

deltaWidth = 0;

}

**if** ((**framingRect**.height() + deltaHeight > screenResolution.**y** - 4) || (**framingRect**.height() + deltaHeight < 50)) {

deltaHeight = 0;

}

**int** newWidth = **framingRect**.width() + deltaWidth;

**int** newHeight = **framingRect**.height() + deltaHeight;

**int** leftOffset = (screenResolution.**x** - newWidth) / 2;

**int** topOffset = (screenResolution.**y** - newHeight) / 2;

**framingRect** = **new** Rect(leftOffset, topOffset, leftOffset + newWidth, topOffset + newHeight);

**framingRectInPreview** = **null**;

} **else** {

**requestedFramingRectWidth** = deltaWidth;

**requestedFramingRectHeight** = deltaHeight;

}

}

**public** PlanarYUVLuminanceSource buildLuminanceSource(**byte**[] data, **int** width, **int** height) {

Rect rect = getFramingRectInPreview();

**if** (rect == **null**) {

**return null**;

}

**return new** PlanarYUVLuminanceSource(data, width, height, rect.**left**, rect.**top**,

rect.width(), rect.height(), **reverseImage**);

}

}

1. **PreferencesActivity.java**

**package** edu.sfsu.cs.orange.ocr;

**import** android.content.SharedPreferences;

**import** android.content.SharedPreferences.OnSharedPreferenceChangeListener;

**import** android.os.Bundle;

**import** android.preference.EditTextPreference;

**import** android.preference.ListPreference;

**import** android.preference.PreferenceActivity;

**import** android.preference.PreferenceManager;

**import** edu.sfsu.cs.orange.ocr.language.LanguageCodeHelper;

**import** edu.sfsu.cs.orange.ocr.language.TranslatorBing;

**import** edu.sfsu.cs.orange.ocr.language.TranslatorGoogle;

**public class** PreferencesActivity **extends** PreferenceActivity **implements**

OnSharedPreferenceChangeListener {

*// Preference keys not carried over from ZXing project*

**public static final** String ***KEY\_SOURCE\_LANGUAGE\_PREFERENCE*** = **"sourceLanguageCodeOcrPref"**;

**public static final** String ***KEY\_TARGET\_LANGUAGE\_PREFERENCE*** = **"targetLanguageCodeTranslationPref"**;

**public static final** String ***KEY\_TOGGLE\_TRANSLATION*** = **"preference\_translation\_toggle\_translation"**;

**public static final** String ***KEY\_CONTINUOUS\_PREVIEW*** = **"preference\_capture\_continuous"**;

**public static final** String ***KEY\_PAGE\_SEGMENTATION\_MODE*** = **"preference\_page\_segmentation\_mode"**;

**public static final** String ***KEY\_OCR\_ENGINE\_MODE*** = **"preference\_ocr\_engine\_mode"**;

**public static final** String ***KEY\_CHARACTER\_BLACKLIST*** = **"preference\_character\_blacklist"**;

**public static final** String ***KEY\_CHARACTER\_WHITELIST*** = **"preference\_character\_whitelist"**;

**public static final** String ***KEY\_TOGGLE\_LIGHT*** = **"preference\_toggle\_light"**;

**public static final** String ***KEY\_TRANSLATOR*** = **"preference\_translator"**;

*// Preference keys carried over from ZXing project*

**public static final** String ***KEY\_AUTO\_FOCUS*** = **"preferences\_auto\_focus"**;

**public static final** String ***KEY\_DISABLE\_CONTINUOUS\_FOCUS*** = **"preferences\_disable\_continuous\_focus"**;

**public static final** String ***KEY\_HELP\_VERSION\_SHOWN*** = **"preferences\_help\_version\_shown"**;

**public static final** String ***KEY\_NOT\_OUR\_RESULTS\_SHOWN*** = **"preferences\_not\_our\_results\_shown"**;

**public static final** String ***KEY\_REVERSE\_IMAGE*** = **"preferences\_reverse\_image"**;

**public static final** String ***KEY\_PLAY\_BEEP*** = **"preferences\_play\_beep"**;

**public static final** String ***KEY\_VIBRATE*** = **"preferences\_vibrate"**;

**public static final** String ***TRANSLATOR\_BING*** = **"Bing Translator"**;

**public static final** String ***TRANSLATOR\_GOOGLE*** = **"Google Translate"**;

**private** ListPreference **listPreferenceSourceLanguage**;

**private** ListPreference **listPreferenceTargetLanguage**;

**private** ListPreference **listPreferenceTranslator**;

**private** ListPreference **listPreferenceOcrEngineMode**;

**private** EditTextPreference **editTextPreferenceCharacterBlacklist**;

**private** EditTextPreference **editTextPreferenceCharacterWhitelist**;

**private** ListPreference **listPreferencePageSegmentationMode**;

**private static** SharedPreferences *sharedPreferences*;

@Override

**protected void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

addPreferencesFromResource(R.xml.***preferences***);

*sharedPreferences* = PreferenceManager.*getDefaultSharedPreferences*(**this**);

**listPreferenceSourceLanguage** = (ListPreference) getPreferenceScreen().findPreference(***KEY\_SOURCE\_LANGUAGE\_PREFERENCE***);

**listPreferenceTargetLanguage** = (ListPreference) getPreferenceScreen().findPreference(***KEY\_TARGET\_LANGUAGE\_PREFERENCE***);

**listPreferenceTranslator** = (ListPreference) getPreferenceScreen().findPreference(***KEY\_TRANSLATOR***);

**listPreferenceOcrEngineMode** = (ListPreference) getPreferenceScreen().findPreference(***KEY\_OCR\_ENGINE\_MODE***);

**editTextPreferenceCharacterBlacklist** = (EditTextPreference) getPreferenceScreen().findPreference(***KEY\_CHARACTER\_BLACKLIST***);

**editTextPreferenceCharacterWhitelist** = (EditTextPreference) getPreferenceScreen().findPreference(***KEY\_CHARACTER\_WHITELIST***);

**listPreferencePageSegmentationMode** = (ListPreference) getPreferenceScreen().findPreference(***KEY\_PAGE\_SEGMENTATION\_MODE***);

*// Create the entries/entryvalues for the translation target language list.*

initTranslationTargetList();

}

@Override

**public void** onSharedPreferenceChanged(SharedPreferences sharedPreferences,

String key) {

*// Update preference summary values to show current preferences*

**if** (key.equals(***KEY\_TRANSLATOR***)) {

**listPreferenceTranslator**.setSummary(sharedPreferences.getString(key, CaptureActivity.***DEFAULT\_TRANSLATOR***));

} **else if**(key.equals(***KEY\_SOURCE\_LANGUAGE\_PREFERENCE***)) {

*// Set the summary text for the source language name*

**listPreferenceSourceLanguage**.setSummary(LanguageCodeHelper.*getOcrLanguageName*(getBaseContext(), sharedPreferences.getString(key, CaptureActivity.***DEFAULT\_SOURCE\_LANGUAGE\_CODE***)));

*// Retrieve the character blacklist/whitelist for the new language*

String blacklist = OcrCharacterHelper.*getBlacklist*(sharedPreferences, **listPreferenceSourceLanguage**.getValue());

String whitelist = OcrCharacterHelper.*getWhitelist*(sharedPreferences, **listPreferenceSourceLanguage**.getValue());

*// Save the character blacklist/whitelist to preferences*

sharedPreferences.edit().putString(***KEY\_CHARACTER\_BLACKLIST***, blacklist).commit();

sharedPreferences.edit().putString(***KEY\_CHARACTER\_WHITELIST***, whitelist).commit();

*// Set the blacklist/whitelist summary text*

**editTextPreferenceCharacterBlacklist**.setSummary(blacklist);

**editTextPreferenceCharacterWhitelist**.setSummary(whitelist);

} **else if** (key.equals(***KEY\_TARGET\_LANGUAGE\_PREFERENCE***)) {

**listPreferenceTargetLanguage**.setSummary(LanguageCodeHelper.*getTranslationLanguageName*(**this**, sharedPreferences.getString(key, CaptureActivity.***DEFAULT\_TARGET\_LANGUAGE\_CODE***)));

} **else if** (key.equals(***KEY\_PAGE\_SEGMENTATION\_MODE***)) {

**listPreferencePageSegmentationMode**.setSummary(sharedPreferences.getString(key, CaptureActivity.***DEFAULT\_PAGE\_SEGMENTATION\_MODE***));

} **else if** (key.equals(***KEY\_OCR\_ENGINE\_MODE***)) {

**listPreferenceOcrEngineMode**.setSummary(sharedPreferences.getString(key, CaptureActivity.***DEFAULT\_OCR\_ENGINE\_MODE***));

} **else if** (key.equals(***KEY\_CHARACTER\_BLACKLIST***)) {

*// Save a separate, language-specific character blacklist for this language*

OcrCharacterHelper.*setBlacklist*(sharedPreferences,

**listPreferenceSourceLanguage**.getValue(),

sharedPreferences.getString(key, OcrCharacterHelper.*getDefaultBlacklist*(**listPreferenceSourceLanguage**.getValue())));

*// Set the summary text*

**editTextPreferenceCharacterBlacklist**.setSummary(sharedPreferences.getString(key, OcrCharacterHelper.*getDefaultBlacklist*(**listPreferenceSourceLanguage**.getValue())));

} **else if** (key.equals(***KEY\_CHARACTER\_WHITELIST***)) {

*// Save a separate, language-specific character blacklist for this language*

OcrCharacterHelper.*setWhitelist*(sharedPreferences,

**listPreferenceSourceLanguage**.getValue(),

sharedPreferences.getString(key, OcrCharacterHelper.*getDefaultWhitelist*(**listPreferenceSourceLanguage**.getValue())));

*// Set the summary text*

**editTextPreferenceCharacterWhitelist**.setSummary(sharedPreferences.getString(key, OcrCharacterHelper.*getDefaultWhitelist*(**listPreferenceSourceLanguage**.getValue())));

}

**if** (key.equals(***KEY\_TRANSLATOR***)) {

initTranslationTargetList();

}

}

**void** initTranslationTargetList() {

}

@Override

**protected void** onResume() {

**super**.onResume();

**listPreferenceTranslator**.setSummary(*sharedPreferences*.getString(***KEY\_TRANSLATOR***, CaptureActivity.***DEFAULT\_TRANSLATOR***));

**listPreferenceSourceLanguage**.setSummary(LanguageCodeHelper.*getOcrLanguageName*(getBaseContext(), *sharedPreferences*.getString(***KEY\_SOURCE\_LANGUAGE\_PREFERENCE***, CaptureActivity.***DEFAULT\_SOURCE\_LANGUAGE\_CODE***)));

**listPreferenceTargetLanguage**.setSummary(LanguageCodeHelper.*getTranslationLanguageName*(getBaseContext(), *sharedPreferences*.getString(***KEY\_TARGET\_LANGUAGE\_PREFERENCE***, CaptureActivity.***DEFAULT\_TARGET\_LANGUAGE\_CODE***)));

**listPreferencePageSegmentationMode**.setSummary(*sharedPreferences*.getString(***KEY\_PAGE\_SEGMENTATION\_MODE***, CaptureActivity.***DEFAULT\_PAGE\_SEGMENTATION\_MODE***));

**listPreferenceOcrEngineMode**.setSummary(*sharedPreferences*.getString(***KEY\_OCR\_ENGINE\_MODE***, CaptureActivity.***DEFAULT\_OCR\_ENGINE\_MODE***));

**editTextPreferenceCharacterBlacklist**.setSummary(*sharedPreferences*.getString(***KEY\_CHARACTER\_BLACKLIST***, OcrCharacterHelper.*getDefaultBlacklist*(**listPreferenceSourceLanguage**.getValue())));

**editTextPreferenceCharacterWhitelist**.setSummary(*sharedPreferences*.getString(***KEY\_CHARACTER\_WHITELIST***, OcrCharacterHelper.*getDefaultWhitelist*(**listPreferenceSourceLanguage**.getValue())));

getPreferenceScreen().getSharedPreferences().registerOnSharedPreferenceChangeListener(**this**);

}

@Override

**protected void** onPause() {

**super**.onPause();

getPreferenceScreen().getSharedPreferences().unregisterOnSharedPreferenceChangeListener(**this**);

}

}

1. **capture.xml**

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

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<SurfaceView android:id="@+id/preview\_view"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:layout\_centerInParent="true"/>

<edu.sfsu.cs.orange.ocr.ViewfinderView

android:id="@+id/viewfinder\_view"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:background="#0000"/>

<LinearLayout android:id="@+id/result\_view"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:background="@color/result\_view"

android:visibility="gone"

android:padding="4dip">

<LinearLayout

android:orientation="horizontal"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:gravity="center"

android:padding="12dip">

<LinearLayout

android:orientation="vertical"

android:layout\_width="wrap\_content"

android:layout\_height="fill\_parent"

android:gravity="right|center\_vertical">

<ImageView android:id="@+id/image\_view"

android:layout\_width="160dip"

android:layout\_height="wrap\_content"

android:maxWidth="160dip"

android:maxHeight="160dip"

android:layout\_marginBottom="4dip"

android:adjustViewBounds="true"

android:scaleType="centerInside"/>

<LinearLayout

android:orientation="horizontal"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

<TextView android:id="@+id/source\_language\_text\_view\_label"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="OCR"

android:textColor="@color/result\_minor\_text"

android:textStyle="bold"

android:textSize="14sp"

android:paddingRight="4dip"/>

<TextView android:id="@+id/source\_language\_text\_view"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textColor="@color/result\_minor\_text"

android:textSize="14sp"/>

</LinearLayout>

<LinearLayout

android:orientation="horizontal"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

<TextView android:id="@+id/translation\_language\_label\_text\_view"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Translation"

android:textColor="@color/translation\_text"

android:textStyle="bold"

android:textSize="14sp"

android:paddingRight="4dip"/>

<TextView android:id="@+id/translation\_language\_text\_view"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textColor="@color/translation\_text"

android:textSize="14sp"

android:text="[Target language]"/>

</LinearLayout>

</LinearLayout>

<ScrollView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

<LinearLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<TextView android:id="@+id/ocr\_result\_text\_view"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textColor="@color/result\_text"

android:textColorLink="@color/result\_text"

android:textSize="22sp"

android:paddingLeft="12dip"

android:autoLink="web"/>

<TextView android:id="@+id/translation\_text\_view"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textColor="@color/translation\_text"

android:textColorLink="@color/translation\_text"

android:textSize="22sp"

android:paddingLeft="12dip"

android:autoLink="web"

android:clickable="true"/>

<LinearLayout android:id="@+id/indeterminate\_progress\_indicator\_view"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

>

<ProgressBar android:id="@+id/progress\_small"

style="?android:attr/progressBarStyleSmall"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:paddingLeft="12dip"

android:paddingTop="10dip"/>

</LinearLayout>

</LinearLayout>

</ScrollView>

</LinearLayout>

</LinearLayout>

<RelativeLayout android:id="@+id/camera\_button\_view"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<TextView android:id="@+id/status\_view\_top"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentTop="true"

android:background="#0000"

android:text=""

android:textColor="@color/status\_text"

android:textSize="14sp"

android:autoLink="web"

android:clickable="true" />

<edu.sfsu.cs.orange.ocr.camera.ShutterButton

android:id="@+id/shutter\_button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:background="@drawable/shutter\_button"

android:clickable="true"

android:focusable="true"

android:layout\_alignParentRight="true"

android:gravity="center\_vertical"

android:layout\_marginRight="8dip"

android:layout\_marginTop="8dip"

android:scaleType="center"

android:visibility="gone" />

<TextView android:id="@+id/status\_view\_bottom"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:background="#0000"

android:text=""

android:textColor="@color/status\_text"

android:textSize="14sp"

android:autoLink="web"

android:clickable="true" />

</RelativeLayout>

</FrameLayout>

1. **Manifest.xml**

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

<**manifest xmlns:android="http://schemas.android.com/apk/res/android"**

**package="edu.sfsu.cs.orange.ocr"**

**android:installLocation="preferExternal"**

**android:versionName="1.0"**

**android:versionCode="2"**

>

<**uses-sdk android:minSdkVersion="8" android:targetSdkVersion="19"**/>

<**supports-screens android:xlargeScreens="true"**

**android:largeScreens="true"**

**android:normalScreens="true"**

**android:smallScreens="true"** />

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

<**uses-feature android:name="android.hardware.camera.flash" android:required="false"** />

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

<**uses-feature android:name="android.hardware.screen.landscape"**/>

<**application android:icon="@drawable/ic\_launcher"**

**android:label="@string/app\_name"**>

<**activity android:name=".CaptureActivity"**

**android:screenOrientation="landscape"**

**android:configChanges="orientation|keyboardHidden|screenSize"**

**android:theme="@android:style/Theme.NoTitleBar.Fullscreen"**

**android:windowSoftInputMode="stateAlwaysHidden"**

>

<**intent-filter**>

<**action android:name="android.intent.action.MAIN"**/>

<**category android:name="android.intent.category.LAUNCHER"**/>

</**intent-filter**>

</**activity**>

<**activity android:name=".HelpActivity"**

**android:screenOrientation="user"**>

<**intent-filter**>

<**action android:name="android.intent.action.VIEW"**/>

<**category android:name="android.intent.category.DEFAULT"**/>

</**intent-filter**>

</**activity**>

<**activity android:name=".PreferencesActivity"** />

</**application**>

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

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

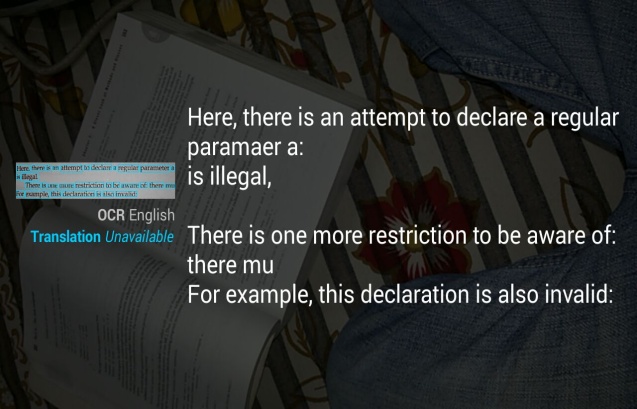
<**uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"**/>

</**manifest**>

**4.3.2 SCREENSHOTS:**



**Figure 4.1**



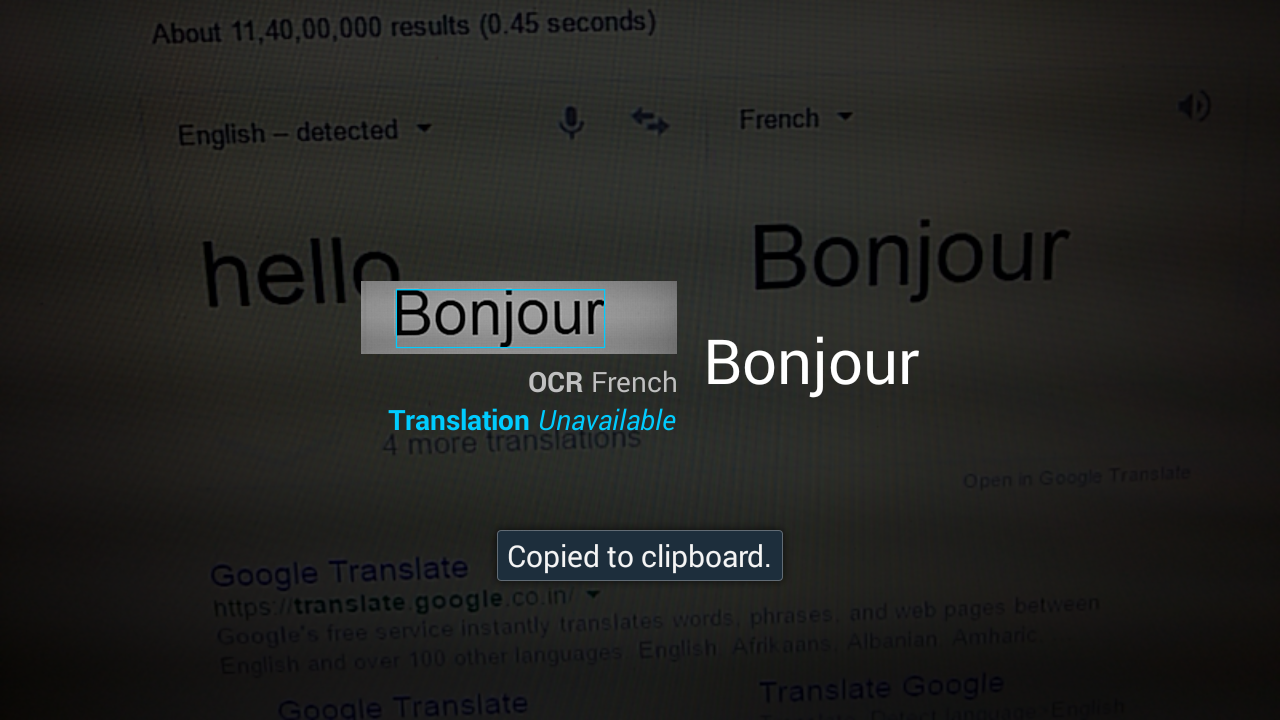
**Figure 4.2**

****

**Figure 4.3**

****

**Figure 4.4**

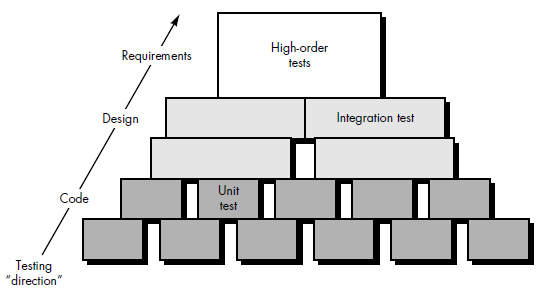
****

**Figure 4.5**

**5. TESTING**

The development of software systems involves a series of production activities where opportunities for injection of human fallibilities are enormous. Errors may begin to occur at the very inception of the process where the objectives may be erroneously or imperfectly specified, as well as in later design and development stages. Because of human inability to perform and communicate with perfection, software development is accompanied by a quality assurance activity.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and code generation (Figure 5.1). The increasing visibility of software as a system element and the attendant "costs" associated with a software failure are motivating forces for well-planned, thorough testing. It is not unusual for a software development organization to expend between 30 and 40 percent of total project effort on testing. In the extreme, testing of human-rated software (e.g., flight control, nuclear reactor monitoring) can cost three to five times as much as all other software engineering steps combined.

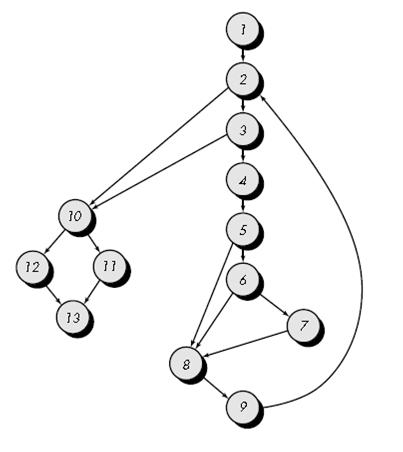


**Figure 5.1: Software testing steps**

**5.1 TESTING METHODS AND STRATEGIES USED**

**WHITE BOX TESTING**

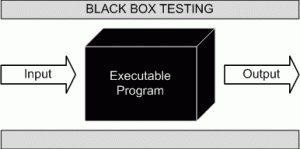
White-box testing is when the tester has access to the internal data structures (Figure 5.2) and algorithms including the code that implements these. White-box testing methods can also be used to evaluate the completeness of a test suite that was created with black-box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important [function points](http://en.wikipedia.org/wiki/Function_points) have been tested.



**Figure 5.2: White Box Testing**

**BLACK BOX TESTING**

Black-box testing treats the software as a "black box"—without any knowledge of internal implementation. Black-box testing (Figure 5.3) methods include: [equivalence partitioning](http://en.wikipedia.org/wiki/Equivalence_partitioning), [boundary value analysis](http://en.wikipedia.org/wiki/Boundary_value_analysis), [all-pairs testing](http://en.wikipedia.org/wiki/All-pairs_testing), [fuzz testing](http://en.wikipedia.org/wiki/Fuzz_testing), [model-based testing](http://en.wikipedia.org/wiki/Model-based_testing), [exploratory testing](http://en.wikipedia.org/wiki/Exploratory_testing) and specification-based testing.

****

**Figure 5.3: Black Box Testing**

**5.2 TEST CASES**

**TEST CASES IN SOFTWARE TESTING**

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. The mechanism for determining whether a software program or system has passed or failed such a test is known as a test oracle. In some settings, an oracle could be a requirement or use case, while in others it could be a heuristic. It may take many test cases to determine that a software program or system is considered sufficiently scrutinized to be released. Test cases are often referred to as test scripts, particularly when written. Written test cases are usually collected into test suites.

**FORMAL TEST CASES**

In order to fully test that all the requirements of an application are met, there must be at least two test cases for each requirement: one positive test and one negative test. If a requirement has sub-requirements, each sub-requirement must have at least two test cases. Keeping track of the link between the requirement and the test is frequently done using a traceability matrix. Written test cases should include a description of the functionality to be tested, and the preparation required to ensure that the test can be conducted. A formal written test-case is characterized by a known input and by an expected output, which is worked out before the test is executed. The known input should test a pre condition and the expected output should test a post condition.

**INFORMAL TEST CASES**

For applications or systems without formal requirements, test cases can be written based on the accepted normal operation of programs of a similar class. In some schools of testing, test cases are not written at all but the activities and results are reported after the tests have been run.

In scenario testing, hypothetical stories are used to help the tester think through a complex problem or system. These scenarios are usually not written down in any detail. They can be as simple as a diagram for a testing environment or they could be a description written in prose. The ideal scenario test is a story that is motivating, credible, complex, and easy to evaluate. They are usually different from test cases in that test cases are single steps while scenarios cover a number of steps of the key.

**TEST DATA AND RESULTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No** | **TEST CASES** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** |
| 1. | OCR English | Hello | Hello | Hello |
| 2. | OCR French | Bonjour | Bonjour | Bonjour |
| 3. | OCR Hindi | नमस्ते | नमस्ते | नमस्ते |
| 4. | OCR Chinese | 你好 | 你好 | 你好 |

**Table 5.1: Test Case**

**5.3 TEST ANALYSIS**

While testing we realized that on giving desired inputs we are getting desired outputs

* When user clicks the image, he gets proper recognized text.
* The recognized text is being properly copied as well as shared.
* User gets the multiple language support as desired.

Therefore one can say that the application is running successfully and have no bugs or working issues

**6. CONCLUSION**

A future forecast of an OCR is also discussed. We focus on using orientation sensor, embedded high resolution camera and digital image processing techniques to solve OCR related issues on camera captured images.

This Application provides fast, robust and extremely high Quality performance because of having improved Auto focus behavior, continuous dynamic preview, improved noise tolerance feature and no remote computing overhead.

**6.1** **LIMITATIONS**

* User must have an android based smart phone.
* Mobile phone should be optimally charged.
* Mobile phone should have an adequate camera to capture the images.
* User must keep the phone steady while taking the image.
* User has to set the area of frame before capturing the image.

**6.2** **FUTURE ENHANCEMENTS AND SUGGESTIONS**

* Text recognition on video based image.
* 3-D images, text can be recognized.
* Application can be enhanced by minimal amount of error.
* Backup’s can be generated by using servers.
* Application could read the text without setting the framearea.

**REFERENCES**

[1] Cormac O’Connell, “Optical Character Recognition”.

[2] The Book “Android in Action third edition” by [Chris King](https://www.google.co.in/search?espv=2&q=chris+king&stick=H4sIAAAAAAAAAOPgE-LRT9c3NErKS0vJSTFVgvCSC03MSsyStGSyk630k_Lzs_XLizJLSlLz4svzi7KtEktLMvKLALGLR-o8AAAA&sa=X&ved=0ahUKEwjC2fboq73JAhWTjo4KHTDyC3YQmxMInAEoATAP), [Robi Sen](https://www.google.co.in/search?espv=2&q=robi+sen&stick=H4sIAAAAAAAAAOPgE-LRT9c3NErKS0vJSTFVgvDyKgwMizIMtGSyk630k_Lzs_XLizJLSlLz4svzi7KtEktLMvKLAIgWZsI8AAAA&sa=X&ved=0ahUKEwjC2fboq73JAhWTjo4KHTDyC3YQmxMInQEoAjAP).

[3] Available at Website ‘www.developer.android.com’ and ‘www.androidhive.com’.

[4] UML Diagrams: ‘www.smartdraw.com’.

[5] Roger S. Pressman, “Software Engineering”.

[6] Available at http://developer.android.com/training/maps/index.html

[7] Available at https://en.wikipedia.org/wiki/Ocr.

[8] Available at http://www.androidhive.info/2012/01/android-working-with-ndk/

[9] Available at http://www.codeguru.com/

**APPENDIX A**

**USER MANUAL**

User Manuals are the basis of any project. Because until and unless we don’t know “how to operate the software?” It is of no use. So here are some of the useful tips for the user.

1. Opening of APK File

After clicking on the .apk file, installation from unknown sources and install the application.

2. How to Access the Software:

* Install the APK file
* Opening the Application
* Enjoy & Run