AN ANDROID APPLICATION FOR BENGALI OPTICAL CHARACTER RECOGNITION FROM IMAGE

BY

MD. MASUD KARIM
ID: 141-15-3368

AND

MD. REZWANUR RAHMAN
ID: 141-15-3357

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science and Engineering

Supervised By

Md. Zahid Hasan
Assistant Professor
Department of CSE
Daffodil International University

Co-Supervised By

Mr. Abdullah Al-Mamun
Senior Lecturer
Department of CSE
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY
DHAKA, BANGLADESH
MAY 2018

APPROVAL

This Project titled **"An Android Application for Bengali Optical Character Recognition from Image"**, submitted by Md. Masud Karim, ID: 141-15-3368 and Md. Rezwanur Rahman, ID: 141-15-3357 to the Department of Computer Science and Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering (CSE) and approved as to its style and contents. The presentation has been held on May 5, 2018.

BOARD OF EXAMINERS

Dr. Syed Akhter Hossain — **Chairman**
Professor and Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Dr. Sheak Rashed Haider Noori — **Internal Examiner**
Associate Professor and Associate Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

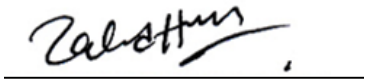
Md. Zahid Hasan — **Internal Examiner**
Assistant Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Dr. Mohammad Shorif Uddin — **External Examiner**
Professor
Department of Computer Science and Engineering
Jahangirnagar University

DECLARATION

We hereby declare that, this project has been done by us under the supervision of **Md. Zahid Hasan, Assistant Professor**, Department of Computer Science and Engineering**,** Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:



Md. Zahid Hasan
Assistant Professor
Department of Computer Science and Engineering
Daffodil International University

Submitted by:

Md. Masud Karim
ID: 141-15-3368
Department of Computer Science and Engineering
Daffodil International University

Md. Rezwanur Rahman
ID: 141-15-3357
Department of Computer Science and Engineering
Daffodil International University

ACKNOWLEDGEMENT

First we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the final year project successfully.

We really grateful and wish our profound our indebtedness to **Md. Zahid Hasan, Assistant Professor**, Department of Computer Science and Engineering, Daffodil International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of "Bengali Optical Character Recognition for Android" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior draft and correcting them at all stage have made it possible to complete this project.

We would like to express our heartiest gratitude to prof. **Dr. Syed Akhter Hossain, Head,** Department of Computer Science and Engineering for his kind help to finish our project and also to other faculty member and the staff of Computer Science and Engineering department of Daffodil International University.

We would like to thank our entire course mate in Daffodil International University, who took part in this discuss while completing the course work.

Finally, we must acknowledge with due respect the constant support and patients of our parents.

ABSTRACT

This project is on "**An Android Application for Bengali Optical Character Recognition from Image**". An Optical Character Recognition system for Bengali language is proposed here. This application can detect and recognize Bengali text that is captured by a mobile device or selected from gallery and displays recognized text onto the phone screen. You can edit the recognized text within this software and also can copy text for further use. For developing this application, we have used the Tesseract Optical Character Recognition (OCR) Engine which was developed by Google is an open source OCR application. The output of the OCR is allowed to be edited or copied recognized text for further processing like translate text via translator, save text file via different MSOffice Software. This project is useful for the foreign tourists to navigate while they are traveling around this country. This application will allow users to perform many actions in few minutes, such as copy text and modify it, instead of wasting time on retyping it. UI of this application is user friendly so that any user can use this application. This application is very much time consuming and accurate than other existing Bengali OCR.

TABLE OF CONTENTS

CONTENTS	PAGE
Board of Examiners	ii
Declaration	iii
Acknowledgements	iv
Abstract	v
List of Figures	viii
List of Tables	viii
CHAPTER 1: INTRODUCTION	**01-02**
1.1 Introduction	01
1.2 Motivation	01
1.3 Objectives of the Software	01
1.4 Expected Outcome	02
1.5 Report Layout	02
CHAPTER 2: BACKGROUND	**03-05**
2.1 Introduction	03
2.2 Related Works	03
2.3 Comparative Studies	04
2.4 Scope of the Problem	04
2.5 Challenges	05
CHAPTER 3: REQUIREMENT SPECIFICATION	**06-11**
3.1 Business Process Modeling	06
3.2 Requirement Collection and Analysis	10
3.3 Use Case Modeling and Description	11
3.4 Design Requirements	11

CHAPTER 4: DESIGN SPECIFICATION **12-16**
4.1 Front-end Design 12
4.2 Back-end Design 13
4.3 Interaction Design and UX 14
4.4 Implementation Requirements 15

CHAPTER 5: IMPLEMENTING AND TESTING **17-24**
5.1 Implementation of Database 17
5.2 Implementation of Front-end Design 17
5.3 Implementation of Interactions 20
5.4 Testing Implementation 22
5.5 Test Results and Reports 23

CHAPTER 6: CONCLUSION AND FUTURE SCOPE **25**
6.1 Discussion and Conclusion 25
6.2 Limitation 25
6.3 Scope for Further Developments 25

REFERENCES **26**
APPENDICES **26**

LISTS OF FIGURES

FIGURES	PAGE
Figure 3.1 : System Flow Chart	06
Figure 3.2 : Use Case Model	11
Figure 4.1 : Chart of User Experience	15
Figure 5.1 : Welcome Page	18
Figure 5.2 : Instruction Page	18
Figure 5.3 : Instruction Page	18
Figure 5.4 : Add Image Using Gallery or Camera	19
Figure 5.5 : Option for Add Image	19
Figure 5.6 : Display Recognized Text	20
Figure 5.7 : Image View of Selected Image	21
Figure 5.8 : Text Button for Run OCR	22

LISTS OF TABLES

TABLES	PAGE
Table 5.1 : Testing Objective of Our Application	22
Table 5.2 : Result of Test Objectives	23

CHAPTER 1
INTRODUCTION

1.1 Introduction

OCR means Optical Character Recognition that converts text image into editable text format. An example of Optical Character Recognition (OCR) is ASCII code that the computer can manipulate. We used Unicode which is considered as converted text in our project. There are many recognition systems are available in play store for android. OCR plays eminent role in computer science. Recognition system works frequently for simple languages like English. Because English language has only 26 character sets. For standard text there are 52 numbers of characters including capital and small letters in English Language. Bengali is a complex language but organized language. But OCR system for Bengali language is still in preliminary level for android. The reasons of its complexities are its character shapes, its top bars and end bars more over it has some modified, vowel and compound characters.

1.2 Motivation

Android is a very popular platform. Most of the smart phones use Android Operating system. Now-a-days there are many OCR applications for android platform. Most of them are already used. But unfortunately there is not a good application for Bengali language which can fulfill the basic requirement. Google created an OCR engine, but they didn't make any android application for Bengali language. So, we tried to build an Android based OCR application for solving this issue.

1.3 Objective

Objective of Bengali OCR for Android are given below:

- Converts an image to editable text
- Copy extracted text into clipboard for further use
- Edit extracted text

- Save extracted text as pdf/word document via MSOffice Software
- Translate extracted text via translator application

1.4 Expected Outcomes

Our application can detect Bengali text from image more accurately than other applications. Time complexity of this application will be very low. The user interface will be very simple so that any user will be able to use this application very easily.

1.5 Report Layout

The entire project is composed by six chapters. In the report, layout is summarized that five chapter. Discuss the summarized below:

Chapter 2 covers background of this project, related works, comparative studies, scope of the problem and challenges. Chapter 3 contains business process modeling, requirement collection and analysis, use case modeling and design requirements. Chapter 4 contains front-end, back-end and interaction design and UX. Chapter 5 is about implementation of project and testing results. Chapter 6 covers conclusion, limitation and future development. And at last of this report there is references.

CHAPTER 2
BACKGROUND

2.1 Introduction

OCR is the mechanical or electronic transformation of scanned digital images of handwritten, typewritten or printed text into machine-encoded text for example Unicode. It is largely used application for converting books and documents into text files. Early optical character recognition may be traced to technologies involving telegraphy and creating reading devices for the blind people. Emanuel Goldberg implemented a machine that read characters and converted them into standard telegraph code in 1914. Concurrently, Edmund Fournier d'Albe invented the Optophone which is a handheld scanner that when moved across a printed page, produced tones that corresponded to specific letters or characters. Since 1950, OCR has emerged a major research field for its usefulness. All over the world there are many widely spoken languages like English, Chinese, Arabic, Japanese, and Bengali etc. Bengali is ranked 5th as speaking language all over the world. Here we will present a total overview of Bengali OCR and its existing challenge is to know the development procedure and to estimate what more requires to do to create a complete Bengali OCR for making things easier.

2.2 Related Works

Bengali OCR isn't a recent work, but there are very few mentionable work in this field. For example: CamScanner, Text fairy etc. For Bengali OCR 'BORCA and Alpona-Pathok' was made publicly all over the world in 2006. But They were not open source. In 2007 the Center for Research on Bengali Language Processing (CRBLP) released BengaliOCR – the first open source OCR software for Bengali language. BengaliOCR is a complete OCR framework and has a recognition rate of up to 98%(in limited domains) but it also has many limitations.

2.3 Comparative Studies

In existing software there is many complications and limitations. The user interface is not much user friendly. The accuracy and time consumption of software is very annoying. But in our software we have used the most accurate OCR engine Tesseract which was developed by Google. So it's accuracy and time consumption is better than other existing softwares. And we have tried our level best to make this software user friendly.

2.4 Scope of the Problem

Each author has used their own set of data to build system. As a result, comparative analysis does not produce a really meaningful result in this project. Some authors addressed noise detection and cleaning phase in their works to make system convenient. However, a comprehensive solution for elimination of all types of noise for Bengali OCR system is not available. The reader has already understood that Bengali has not only basic characters; but also rich with modifiers and compound characters. Placement of modifiers may happen on the upper, lower, left or right side of original characters which generates a lot of complications. Rarely authors could confidently claim that a particular segmentation and classification scheme has dealt perfectly with all of them. Again lack of standard or benchmark samples do not allow one to make a comprehensive testing of their application. Existing applications are built up for printed documents where complex structure of documents is assumed not to present. Also these applications do not have any good quality page layout analyzer which could automatically identify picture and text paragraph of an input image. These have some limitations on line segmentation for newspaper document images because of the appearance of joining problem between two lines in the digital images. Existing OCR does not perform well for medium quality printed low resolution document images. Many of the existing applications are doing well but not user friendly and very complex user interface. There is some good quality application available, but they need premium subscription.

2.5 Challenges

Characters in Bengali are not alphabetical as in English where the characters largely have one-sound one-symbol characteristics. It is a mixture of syllabic and alphabetic characters. The use of modified and compound characters is also very common in Bengali language. This project presents methods for recognizing Bengali printed characters based on view-based approach.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 Business Process Modeling

This model not only provides a clear view about the software that has been developed but also helps to achieve the goal. In this model when user need he or she will be update easily in his/her system or software and in this model are also useful for the developers.

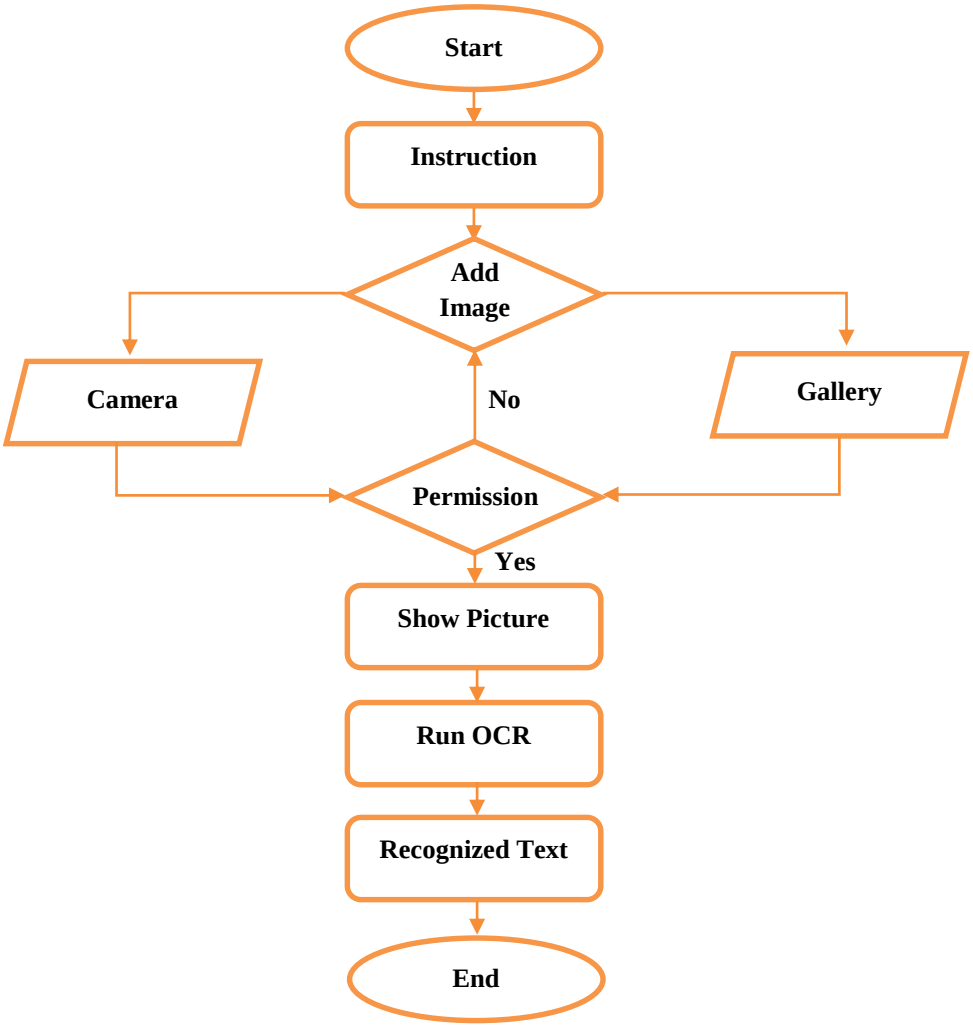


Fig 3.1: System Flow Chart

- **Architecture**

Tesseract assumes that its input is a binary image with optional polygonal text regions defined. Processing follows a traditional step-by-step pipeline, but some of the stages were unusual in their day, and possibly remain so even now. The first step is a connected component analysis in which outlines of the components are stored. This was a computationally expensive design decision at the time, but had a significant advantage: by inspection of the nesting of outlines, and the number of child and grandchild outlines, it is simple to detect inverse text and recognize it as easily as black-on-white text. Tesseract was probably the first OCR engine able to handle white-on-black text so trivially. At this stage, outlines are gathered together, purely by nesting, into Blobs. Blobs are organized into text lines, and the lines and regions are analyzed for fixed pitch or proportional text. Text lines are broken into words differently according to the kind of character spacing. Fixed pitch text is chopped immediately by character cells. Proportional text is broken into words using definite spaces and fuzzy spaces.

Recognition then proceeds as a two-pass process. In the first pass, an attempt is made to recognize each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize text lower down the page.

Since the adaptive classifier may have learned something useful too late to make a contribution near the top of the page, a second pass is run over the page, in which words that were not recognized well enough are recognized again.

A final phase resolves fuzzy spaces, and checks alternative hypotheses for the x-height to locate small-cap text.

- **Line and Word Finding**

Line Finding

The line finding algorithm is one of the few parts of Tesseract that has previously been published. The line finding algorithm is designed so that a skewed page can be recognized without having to de-skew, thus saving loss of image quality. The key parts

of the process are blob filtering and line construction. Assuming that page layout analysis has already provided text regions of a roughly uniform text size, a simple percentile height filter removes drop-caps and vertically touching characters. The median height approximates the text size in the region, so it is safe to filter out blobs that are smaller than some fraction of the median height, being most likely punctuation, diacritical marks and noise. The filtered blobs are more likely to fit a model of non-overlapping, parallel, but sloping lines. Sorting and processing the blobs by x-coordinate makes it possible to assign blobs to a unique text line, while tracking the slope across the page, with greatly reduced danger of assigning to an incorrect text line in the presence of skew. Once the filtered blobs have been assigned to lines, a least median of squares fit is used to estimate the baselines, and the filtered-out blobs are fitted back into the appropriate lines. The final step of the line creation process merges blobs that overlap by at least half horizontally, putting diacritical marks together with the correct base and correctly associating parts of some broken characters.

Baseline Fitting

Once the text lines have been found, the baselines are fitted more precisely using a quadratic spline. This was another first for an OCR system, and enabled Tesseract to handle pages with curved baselines, which are a common artifact in scanning, and not just at book bindings. The baselines are fitted by partitioning the blobs into groups with a reasonably continuous displacement for the original straight baseline. A quadratic spline is fitted to the most populous partition, (assumed to be the baseline) by a least squares fit. The quadratic spline has the advantage that this calculation is reasonably stable, but the disadvantage that discontinuities can arise when multiple spline segments are required. A more traditional cubic spline might work better.

Fixed Pitch Detection and Chopping

Tesseract tests the text lines to determine whether they are fixed pitch. Where it finds fixed pitch text, Tesseract chops the words into characters using the pitch, and disables the chopper and associator on these words for the word recognition step.

Proportional Word Finding

Non-fixed-pitch or proportional text spacing is a highly non-trivial task. Tesseract solves most of the problems by measuring gaps in a limited vertical range between the baseline and mean line. Spaces that are close to the threshold at this stage are made fuzzy, so that a final decision can be made after word recognition.

- **Word Recognition**

Part of the recognition process for any character recognition engine is to identify how a word should be segmented into characters. The initial segmentation output from line finding is classified first. The rest of the word recognition step applies only to non-fixed pitch text.

Chopping Joined Characters

While the result from a word is unsatisfactory, Tesseract attempts to improve the result by chopping the blob with worst confidence from the character classifier. Candidate chop points are found from concave vertices of a polygonal approximation of the outline, and may have either another concave vertex opposite, or a line segment. It may take up to 3 pairs of chop points to successfully separate joined characters from the ASCII set.

Chops are executed in priority order. Any chop that fails to improve the confidence of the result is undone, but not completely discarded so that the chop can be re-used later by the associator if needed.

Associating Broken Characters

When the potential chops have been exhausted, if the word is still not good enough, it is given to the associator. The associator makes an A* (best first) search of the segmentation graph of possible combinations of the maximally chopped blobs into candidate characters. It does this without actually building the segmentation graph, but instead maintains a hash table of visited states. The A* search proceeds by pulling candidate new states from a priority queue and evaluating them by classifying unclassified combinations of fragments.

It may be argued that this fully-chop-then-associate approach is at best inefficient, at worst liable to miss important chops, and that may well be the case. The advantage is that the chop-then-associate scheme simplifies the data structures that would be required to maintain the full segmentation graph.

When the A* segmentation search was first implemented in about 1989, Tesseract's accuracy on broken characters was well ahead of the commercial engines of the day. An essential part of that success was the character classifier that could easily recognize broken characters.

3.2 Requirement Collection and Analysis

Requirement collection and analysis is an important part of SDLC. Our system is based on Tesseract OCR engine. As we implemented our system for Bengali Language, so we need Bengali language data and trained data to make our system working. Google is working in this sector from 2006 and they made their source files open source for all. To make this software we need

- Tesseract OCR Engine
- Bengali language data
- Bengali trained data set

As Google already made this files open source and these files are also verified, so we used these files to make our system. Bengali trained data is the most important part of our system. So we carefully stored these data.

3.3 Use Case Modeling and Description

Use case model different types of users of a system and their fields of activities. In the following model the activities of admin and users are indicated.

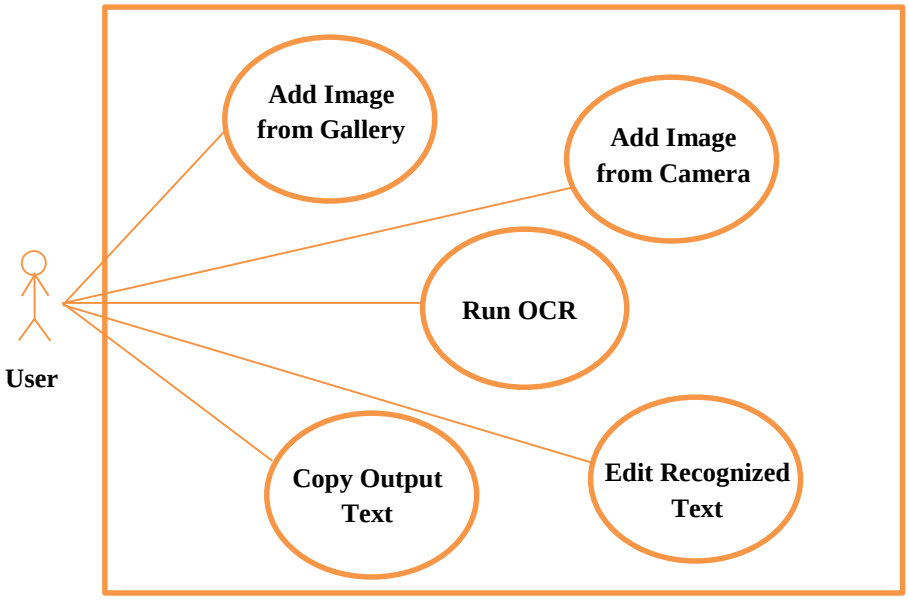


Fig 3.2 Use Case Model

3.4 Design Requirements

Design of our system will make our system unique. To expected output in an easy way we have designed our system carefully and in a decorative way.

When the system will open first time, it will show welcome note and instructions to use the system. Then next skin will show to add image. After selecting image, it will show the image and ask for continue. After pressing Run OCR, it will show the recognized text which is our system's output.

Design requirement for user:

- Welcome Skin
- Instruction Skin
- Gallery
- Camera
- Permission
- Confirm Skin
- Output Skin

CHAPTER 4
DESIGN SPECIFICATION

4.1 Front-end Design

- Linear Layout

  Linear Layout is a view group that aligns all children in a single direction, vertically or horizontally. It can be specified the layout direction with the android:orientation attribute. Linear Layout is used for viewing ImageView, TextView, Button etc.

- Relative Layout

  Relative Layout displays child views in relative positions and is a view group. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent Relative Layout area (such as aligned to the bottom, left or center). For arranging our interface in relative position we used Relative Layout.

- ImageView

  We use ImageView class to display image files in our application. Image file is easy to use but hard to master in Android, because of the various screen sizes in Android devices. An android is enriched with some of the best UI design widgets which allows us to build good looking and attractive user interface based application.

- TextView

  For displaying text to the user of the application we use TextView. Its optionally allows them to edit it to make application user friendly. A TextView is a complete text editor, however the basic class is configured to not allow editing.

- Button

  A button consists of text or an icon or both text and an icon that communicates what action occurs when the user presses it. For Selecting Image and processing image in OCR engine we used Button in this application.

- Frame Layout

  Frame Layout is designed to block out an area on the screen to display a single item in android application. Generally, Frame Layout should be used to hold a single child view, because it can be difficult to organize child views in a way that's scalable to different screen sizes without the children overlapping each other. We use Frame Layout for blocking the area of loaded image on the screen to display selected image to the user which is important in our application to confirm the image is perfectly selected. We also used this method for many other purposes in this project.

- Fragment

  We used Fragment activity for enabling more modular activity design. Because a Fragment is a piece of activity which enables more modular activity design to the application. It won't be wrong if we say, a fragment is a kind of sub-activity.

4.2 Back-end Design

- Tesseract OCR engine:

  Tesseract is an optical character recognition engine for various operating systems. It is free software and open source, released under the Apache License, Version 2.0, and development has been sponsored by Google since 2006. Tesseract was considered one of the most accurate open-source optical character recognition engine than available in 2006. We used Tesseract engine for recognizing Bengali text and display it for further use. It is very easy to install Tesseract optical character recognition (OCR) engine in android project.

- Trained dataset for Bengali Language:

  After an analysis of Tesseract, we found, that to integrate the Bengali script recognition support in Tesseract, we need a complete set of training data. This training data is dependent on the output of the script segment which will be included during test data recognition. It also depends on the nature of the segment that is included in the Tesseract engine; the engine has its own segment to detect lines, words and characters. Since Tesseract is a complete OCR package, once it is trained

with the training data, we do not need to be concerned about feature extraction and recognition in order to recognize the characters in the Bengali script.

The entire task is divided into two parts:

1.Training data generation

2.Test data processing

4.3 Interaction Design and UX

Welcome Page Design:

- Welcome Page
- User Instructions
- Go to Add Image

OCR Page Design:

- Add Image (Camera/Gallery)
- Display Image to user
- Run OCR for recognizing text
- Display recognized text to user
- Copy text

UX

UX means User Experience. This system design should be user friendly for good user experience. Now figure 4.1 can show proof that this system design satisfies its users. In figure 4.1 you can see that real users who are using this application provided their opinion about this application. Here, 19 real user's opinion is available where most of them that means 17 users voted very satisfied after using this application. Only 1 user voted that this application needs improvement or satisfactory for us. And 1 user voted not satisfied and added that this application needs more functionality.

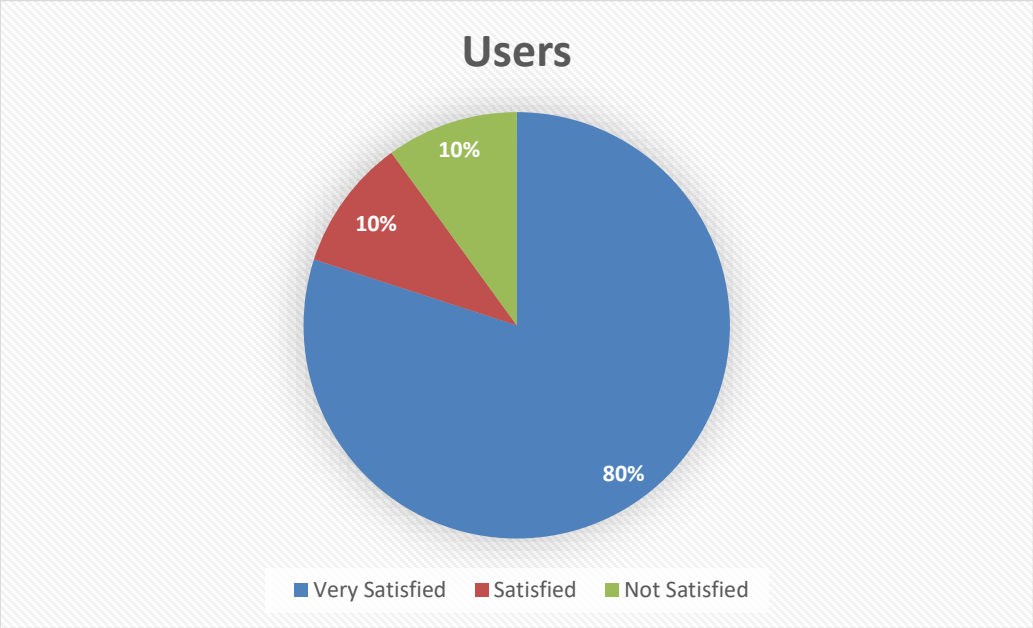


Fig 4.1 Chart of user experience

4.4 Implementation Requirement

To implement our project, we need Android Studio for Build Android application and Tesseract OCR engine and trained data set for Bengali language. We need XML code to design UI/UX for our Application. To implement other thing, we need Java programming language to do Add image and reorganizing text. We need to install Tesseract OCR engine in our project to build this application. We need all the tool to work with this project already I have discussed above.

Software Requirements for our application

- Windows 7 or higher/ Linux (Ubuntu, Fedora)
- Android Studio 2.3 or higher
- Java Development Kit 1.5 or higher
- Java Runtime Environment 1.5 or higher
- Android Operating System

Hardware Requirements for our application

- Windows
  - ✓ Microsoft Windows 7/8/10 (32/64bit)

- ✓ 8GB RAM recommended, Min: 3GB and 1 GB for the Android Emulator
- ✓ Minimum 2 GB of available disk space, Recommended 4 GB
- ✓ Screen Resolution: 1280 x 800 minimum

- Linux
  - ✓ GNOME or KDE Desktop
  - ✓ Tested on Ubuntu 14.04 LTS,
  - ✓ 64-bit distribution capable of running 32-bit applications
  - ✓ GNU C Library 2.19 or later
  - ✓ 8 GB RAM recommended; 3 GB minimum and 1 GB for the Android Emulator
  - ✓ Minimum 2 GB of available disk space, Recommended 4 GB
  - ✓ Screen Resolution: 1280 x 800 minimum

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Implementation of Database

Database implementation is a very difficult part of a system. But this application is a OCR application which is implemented by using Tesseract OCR engine. So here we don't need any kind of database in this application. For Bengali OCR we need some trained data set to recognize text from image using Tesseract OCR engine. Google already released a verified trained data set for Bengali language which will be used in this application.

5.2 Implementation of Front-end Design

Figure 5.1, 5.2, 5.3 is welcome, description and instruction page of our application. It will be shown once when the application will be installed. After first time use this page won't be shown to user. These pages will make the user interface easy to understand and use. There is also skip button in the pages. If users think these pages are boring, then he/she can skip these pages by pressing the skip button. After using this application, user added in their opinion that for using these pages this application become very attractive and user friendly.

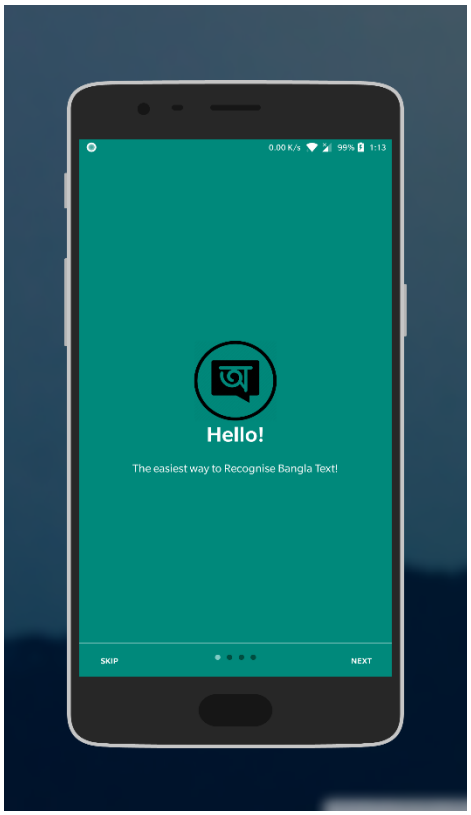


Fig 5.1: Welcome Screen

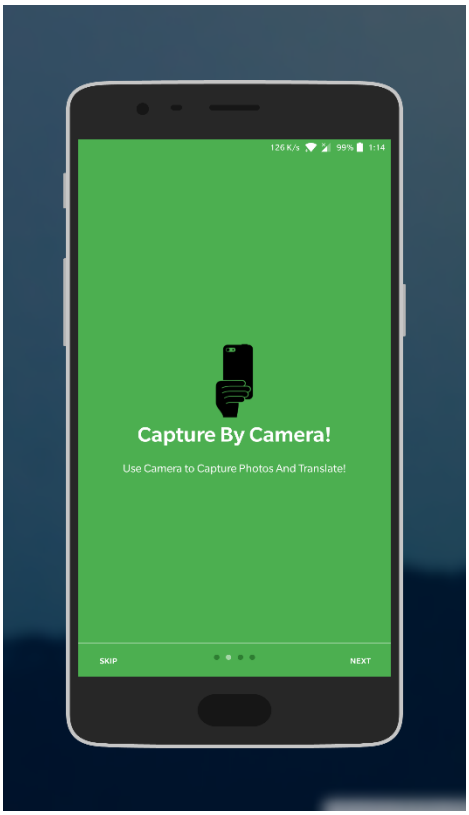


Fig 5.2: Camera Instruction Screen

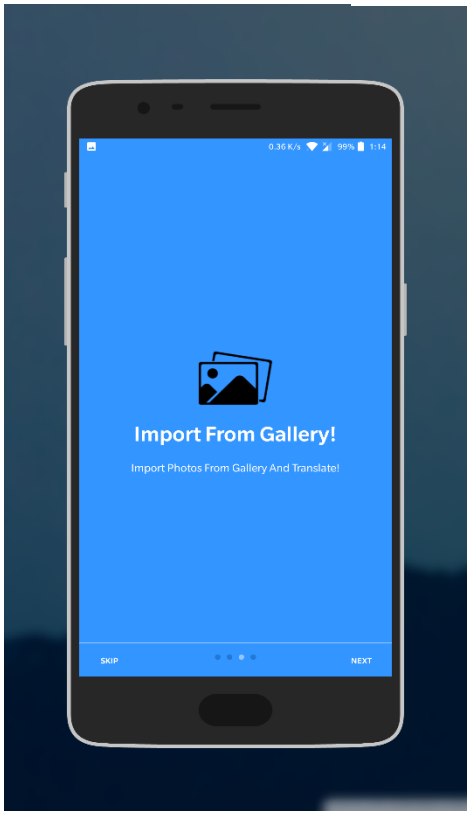


Fig 5.3: Gallery Instruction Screen

Figure 5.4, 5.5 is the selecting image page of this application. User can add image using camera or gallery from it. After click on Add image, a pop up will show two options. One is gallery and another one is camera. If user click on camera, then the phone camera will be opened and he/she can take pictures which he/she want to process. In other hand, if user click on gallery, the phone gallery will be opened and he/she can be able to select picture from storage.

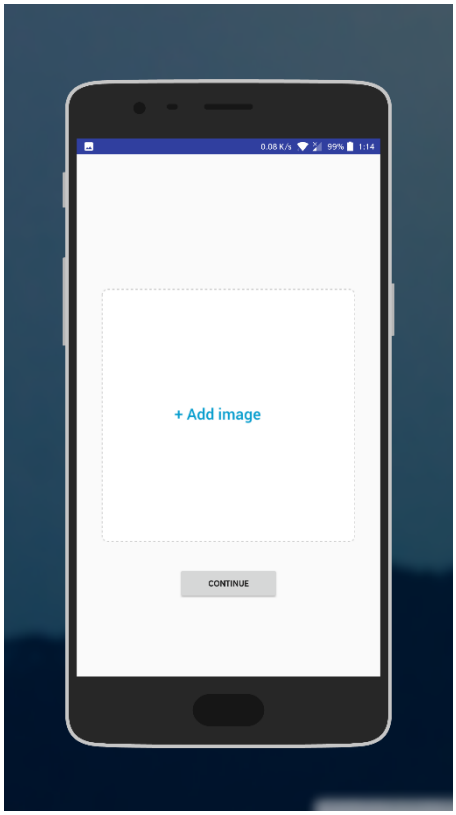


Fig 5.4: Add Image Screen

Fig 5.5: Camera/Gallery Option Screen

Figure 5.6 is the final output page of this application. After pressing the Run OCR button, it will display the recognized text in the below of the page. It will take a very small amount of time to process the image depends on the text size, noise, different shapes, resolution and other things in the image. Then user can edit or copy text from here.

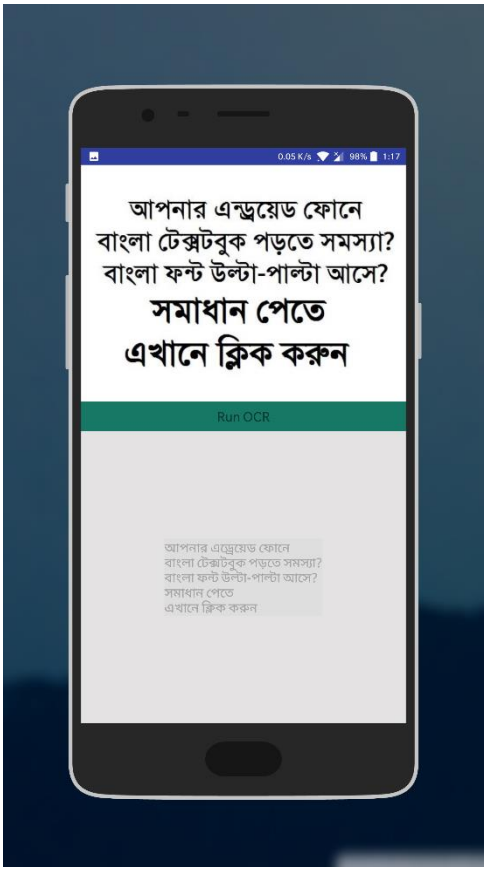


Fig 5.6: Show Recognized Text Screen

5.3 Implementation of Interactions

To make this application interactive we have implemented responsive UI for better user experience. To make things easy we have used image view, Text button, confirm page. The UI design of this application is user friendly.

Figure 5.7 is the image view of selected image of this application to make sure that user have selected the right image.

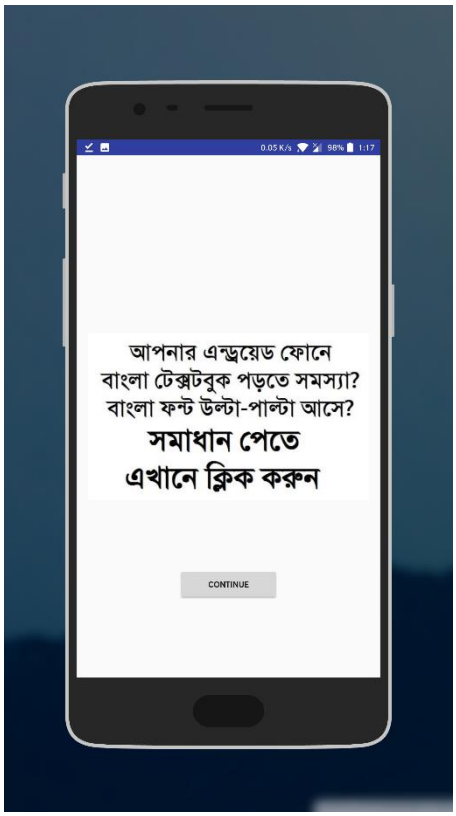


Fig 5.7: Image View Screen

Figure 5.8 is the page where text button is used as Run OCR to make it easy to the user to understand use this application.

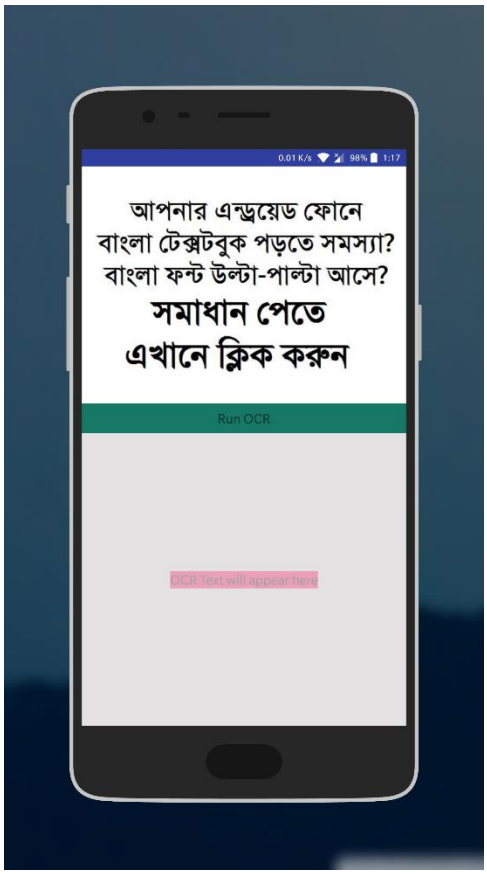


Fig 5.8: OCR Processing Screen

5.4 Testing Implementation

System Testing is a level of the software testing where complete and integrated software is tested. To evaluate the system's compliance with the specified requirements is the purpose of this test.

Table 5.1 : Testing Objective of Our Application

Serial No	Test Objective
01	To check whether the programs run or not
02	To check display all option
03	To check all option are working or not
04	To check whether asked for media permission or not
05	To check whether camera option working or not
06	To check whether gallery option working or not

07	To check whether the OCR engine working or not
08	To check whether image change stopped it or not
09	To check output text can be edited or not
10	To check output text can be copied or not

Table 5.1 is the testing objective of this application. When we tested this application by real time user, we asked these questions to every user. Based on these objectives we have created our test results and reports.

5.5 Test Results and Reports

Table 5.2 shows the test results depending on the test cases implemented in the previous section of this chapter.

Table 5.2 : Result of Test Objectives

Serial No	**Test Objective**	**Results**
01	To check whether the programs run or not	Successfully
02	To check display all option	Successfully
03	To check all option are working or not	Successfully
04	To check whether asked for media permission or not	Successfully
05	To check whether camera option working or not	Successfully
06	To check whether gallery option working or not	Successfully
07	To check whether the OCR engine working or not	Successfully
08	To check whether image change stopped it or not	Successfully
09	To check output text can be edited or not	Successfully
10	To check output text can be copied or not	Successfully

We found the test results quite successful. This application is satisfied by the user. Usability testing examines the following features of this application.

- How easy it is to use this application?
- How easy it is to learn this application?

- How convenient it is to the user?
- How accurate it is to recognize text?

So at the end of the development we can carry out the results as the benefits of usability testing to the end of the user.

- Better application
- User friendly with great UI
- More understandability accepted by users
- Good accuracy

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Discussion and Conclusion

We have successfully met the aims and objectives of our project. Our project "**An Android Application for Bengali Optical Character Recognition from Image**" is completed by using Android Studio, Tesseract OCR engine and trained data set for Bengali language. The application will take image as an input and return recognized Bengali text from image as output. This application is more accurate than other existing application. We have tried our level best to make this application simple so that any user can use this application easily. Basically this application is made to make our daily life easy. It is difficult to type Bengali language in mobile. Using this application, it will be very easy to copy text from existing photo. It will help tourists a lot to make their journey easy in Bangladesh. This application will reduce complexity of Bengali language. It is very simple and time consuming. Hopefully it will be a breakthrough in Bengali optical character recognition system for android operating system.

6.2 Limitations

As we are introducing a beta version of this application, it has some limitations.

- It doesn't support all Bengali fonts.
- It works slowly in big images where is so much text.
- Its accuracy is low for multi-language images.

Our team is working very hard to overcome these limitations in future.

6.3 Scope for Further Developments

We are planning to release this application in Play Store. So we have already figure out some future development features to make this application stable and popular to user.

- It will support all Bengali fonts.
- It will support image cropping. So it will be more time consuming and accurate.
- One click copy recognized text.
- Translate recognized text in other languages.
- Save recognized text in different format like text file, pdf, word document etc.
- Instant live translate from camera.

References:

[1] Learn about Tesseract (software), available at https://en.wikipedia.org/wiki/Tesseract_(software) , last accessed on 25 November 2017 at 11:30pm.

[2] Learn about android, available at https://www.tutorialspoint.com/android , last accessed on 1 December 2017 at 10:20pm.

[3] Learn about Android development, available at https://developer.android.com/develop/index.html , last accessed on 25 February 2018 at 12:45am.

[4] Learn about Android Studio, available at https://developer.android.com/studio/index.html , last accessed on 22 March 2018 at 09:17pm.

[5] Mohit Gupt, Optical Character Recognition On Android – OCR, available at http://www.truiton.com/2016/11/optical-character-recognition-android-ocr/ , last accessed on 15 March 2018 at 01:32am.

[6] Ravi Tamada, Android How to Build Intro Slider for your App, available at https://www.androidhive.info/2016/05/android-build-intro-slider-app/ , last accessed on 20 Februany 2018 at 11:47pm.

[7] Pick Image From Gallery or Camera On Android, available at http://droidmentor.com/pick-image-from-gallery-or-camera/ , last accessed on 20 February 2018 at 03.22am.

Plagiarism

Checked by: https://plagramme.com