International Journal of Engineering & Technology, 7 (4.5) (2018) 634-640

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Snap and split: an android application for bill payment using tesseract OCR

Sugamya kata [1] *, Suresh Pabboju [1], Vinaya Babu [2], Anudeep Medishetti [1]

[1] CBIT, India
[2] SCET, India
**Corresponding author E-mail: sugamya.cbit@gmail.com*

Abstract

Snap and split is a mobile application which uses Optical character recognition to recognize the bill from a printed sheet. It provides an option to tag users and telling them about the shared bill by pushing a notification. Users can tap and pay the bills instantly.Tesseract is one of the best image recognition tools present and uses separate packs for various languages.

Keywords: *Immediate Payment Service (IMPS); OCR (Optical Character Reader); SDK (Software Developer Kit).*

1. Introduction

Technology took over many things and has introduced many smart and easier alternatives to solve daily problems in human life. Everything is now on internet and done in seconds. Data has been digitalized [1] and can be retrieved from any place through internet. We are now trying to solve a daily problem in shops. Generally, if a group of people shop or eat at a place and has to share the bill, overhead arises while individuals paying off their shared amount. So we are coming up with a technique which uses Image Pro- cessing and shares the bill among all the involved individuals. This work is a mobile application developed with JDK 1.8 and Android NDK 11.0 runs on android platform. The language used is Java and technology XML for the User Interface (UI).The pur- pose of this work is to develop a mobile application which pro- vides users a quick and easy way to split bills amongst customers and enhance user experience. Tesseract is an optical character recognition [2] engine for various operating systems. It is free software, released under the Apache License, Version 2.0, and development has been sponsored by Google since 2006. Tesseract is considered one of the most accurate open source OCR [3] en- gines currently available. Tesseract is an open-source OCR engine that was developed at HP between 1984 and 1994. Like a super- nova, it appeared from nowhere for the 1995 UNLV Annual Test of OCR Accuracy[4], shone brightly with its results, and then vanished back under the same cloak of secrecy under which it had been developed. Now for the first time, details of the architecture and algorithms [5] can be revealed.

Tesseract began as a PhD research project in HP Labs, Bristol, and gained momentum as a possible software and/or hardware add-on for HP's line of flatbed scanners. Motivation was provided by the fact that the commercial OCR engines [6] of the day were in their infancy, and failed miserably on anything but the best quality print. After a joint project between HP Labs Bristol, and HP's scanner division in Colorado, Tesseract had a significant lead in accuracy over the commercial engines, but did not become a product. The next stage of its development was back in HP Labs Bristol as an Investigation of OCR [7] for compression. Work concentrated more on improving rejection efficiency than on base-level accura- cy. At the end of this project, at the end of 1994, development ceased entirely. The engine was sent to UNLV for the 1995 Annu- al Test of OCR Accuracy [8], where it proved its worth against the commercial engines of the time. In late 2005, HP released Tesser- act for open source.

1.1. Features

Tesseract was in the top three OCR engines in terms of character accuracy in 1995. It is available for Linux, Windows and Mac OS X, however, due to limited resources only Windows and Ubuntu are rigorously tested by developers. Tesseract up to and including version 2 could only accept TIFF images of simple one column text[9] as inputs. These early versions did not include layout analysis and so inputting multi-columned text, images, or equations produced a garbled output. Since version 3.00 Tesseract has supported output text formatting, hOCR positional information and page layout analysis. Support for a number of new image formats was added using the Leptonica library. Tesseract can detect whether text is monospaced or proportional. The initial versions of Tesseract could only recognize English language text. Tesseract v2 added six additional Western languages (French, Italian, Ger- man, Spanish, Brazilian Portuguese, Dutch). Version 3 extended language support significantly to include ideographic (Chinese & Japanese) and right-to-left (e.g. Arabic, Hebrew) languages as well many more scripts. New languages included Arabic, Bulgari- an, Catalan, Chinese (Simplified and Traditional), Croatian, Czech, Danish, German (Fraktur script), Greek, Finnish, Hebrew, Hindi, Hungarian, Indonesian, Japanese, Korean, Latvian, Lithuanian, Norwegian, Polish, Portuguese, Romanian, Russian, Serbian, Slovak (standard and Fraktur script), Slovenian, Swedish, Tagalog, Tamil, Thai, Turkish, Ukrainian and Vietnamese. V3.04, released in July 2015, added an additional 39 language/script combinations, bringing the total count of support languages to over 100 [10]. New language codes included: amh, asm, aze_cyrl, bod, bos, ceb, cym, dzo, fas, gle, guj, hat, iku, jav, kat, kat_old, kaz, khm, kirkur, lao, lat, mar, mya, nep, ori, pan, pus, san, sin, srp_latn, syr, tgk, tir, uig,

urd, uzb, uzb_cyrl, yid. Tesseract can be trained to work in other languages too. If Tesseract is used to process right- to-left text such Arabic or Hebrew the results are ordered as though it is left-to-right text. Tesseract is suitable for use as a backend, and can be used for more complicated OCR [11] tasks including layout analysis[12] by using a frontend such as OCRopus. Tesseract's output will be very poor quality if the input images are not preprocessed to suit it: Images (especially screen- shots) must be scaled up such that the text x-height is at least 20 pixels, any rotation or skew[13] must be corrected or no text will be recognized, low-frequency changes in brightness must be high- pass filtered, or Tesseract's binarization stage will destroy much of the page, and dark borders must be manually removed, or they will be misinterpreted as characters.

1.2. Aim

The aim of the work is to develop an android application which works on mobile allows the user to snap a bill, tag his friends and share the bill to the tagged users. A notification is pushed to the tagged users regarding the shared bill and allows them to pay the bill now or pay later.

1.3. Problem statement

The worrisome task of splitting the bill can turn a lovely meal or shopping experience into a debate with the discussion of the bill sometimes lasting longer than the actual event. With the growing trends in technology, it is beneficial to the customers if an application exists, that can enable the users to send and receive money instantly without having to know account numbers, IFSC codes or wait for OTP (One Time Password) SMS.
Existing System All the users take the overhead of calculating the shared amount individually. Then the user has to follow the figure1 flow graph process and pay the shared amount to the person who paid at the merchant.

1.4. Disadvantages of existing system

- The payment basis might not be secure.
- The person who paid the bill has to manually maintain the records of all the amount to be acquired.

2. Proposed system

The proposed solution is to introduce an Android based mobile application which uses IMPS technology to share bills and pay bill online. Initially, a user has to register him/her to the application. After registration we have to fill our profile with details of bank and transaction. When a group of people want to share a bill, one among them will capture the bill and tag his/her friends (provided that friends must be registered in the application). This application also allows the users to equally share the bill or individually tag the bill for each item. This will push a notification to the user's device Once the notification is pushed and accepted then one can pay the shared bill amount.

2.1. System architecture

Customer Module:
Customers must register in order to use the application; the customer must link his phone number to his/her bank account.

- When the bill is received, the customer should open the application and snap a photo of the receipt and tag con- tacts from the address book.
- Processor Module:
- The processor tool should extract the total amount from the image of the bill on per items basis and calculate the amount per individual and send out the "amount to be paid" to each contact.
- Immediate Payment Service (IMPS) Module:
- IMPS module is an instant, interbank electronic fund transfer service that can be initiated through mobile phone.
- When a user receives a request, he/she should be able to click a button (pay now), which takes then to a new page. They can select a contact from their phone's address book and enter the amount to be transferred.
- Here, the amount must be deducted from the sender's bank account and to be added into the receivers account.
- In case, a person does not have sufficient money to pay the requester, an IOU message is sent and this infor- mation should be stored in a database.

2.2. Proposed system workflow

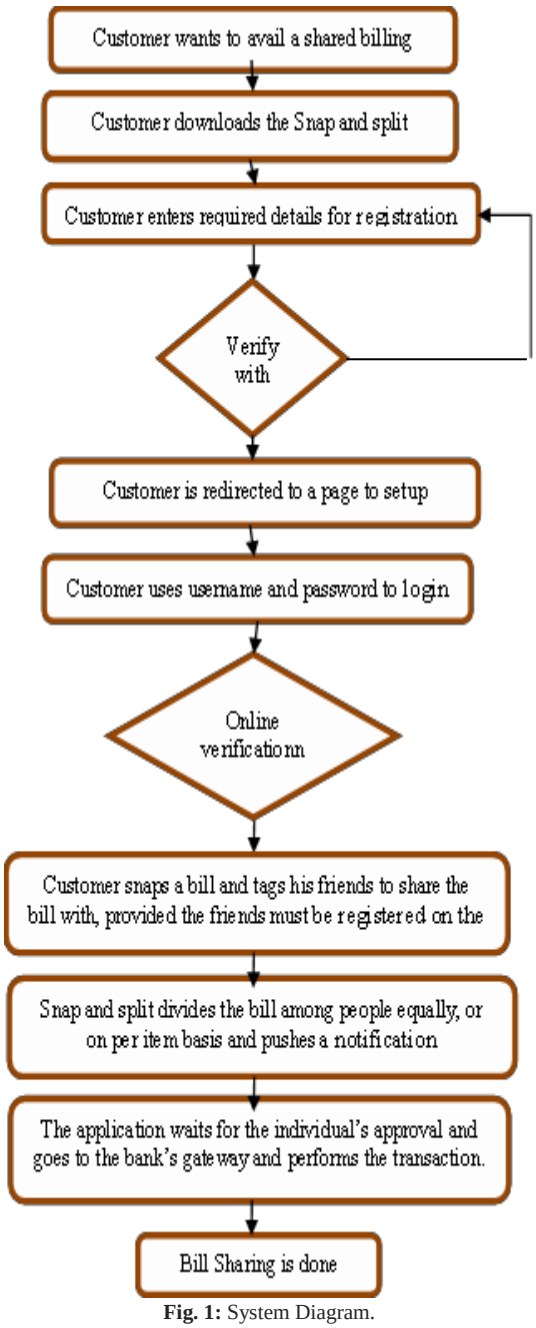


Fig. 1: System Diagram.

Since HP had independently developed page layout analysis technology that was used in products, (and therefore not released for open-source) Tesseract never needed its own page layout analysis. Tesseract therefore 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 ad- vantage: 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.

2.3. Assumptions

- The users need to have an account with the bank in or- der to perform any kind of money transfer.
- The receipts will be in a particular format for the scope of the project/application.
- Only the users who registered to this application will be tagged.
- You can only tag the users in your address book.
- Receipts received from the restaurant will be divided equally among the people.

2.4. Design flow of android application

Since HP had independently-developed page layout analy- sis technology that was used in products, (and therefore not released for open-source) Tesseract never needed its own page layout analysis. Tesseract therefore 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 compo- nent analysis in which outlines of the components are stored. This was a computationally expensive design deci- sion at the time, but had a significant advantage: by inspec- tion 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 re- gions 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 immedi- ately 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 clas- sifier 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.

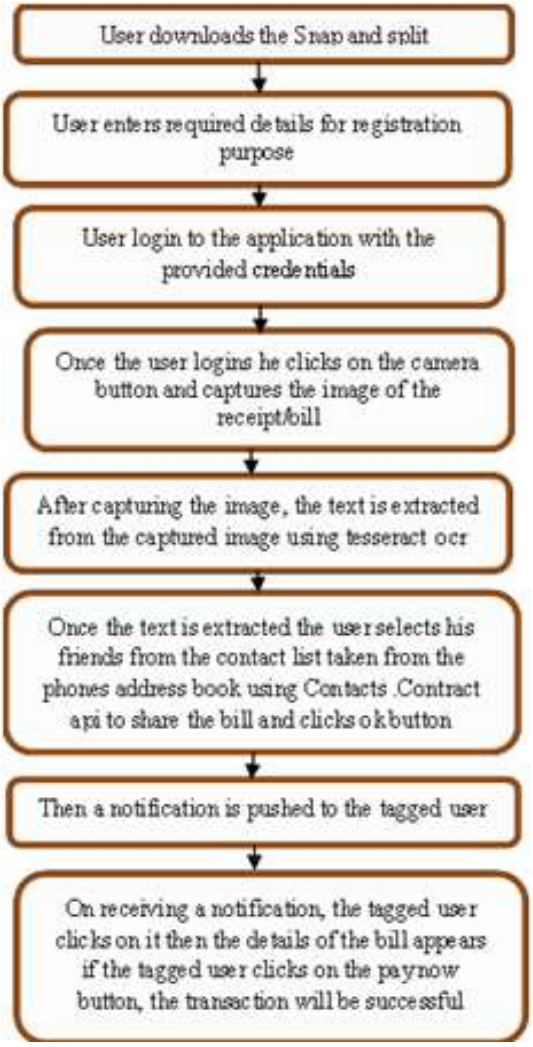


Fig. 2: Design Flow.

3. Implementation

The Implementation phase of any work development is the most important phase as it yields the final solution, which solves the problem at hand. The different modules that comprise the proposed work are detailed here. The software used to develop these modules is JavaTM2 Platform Standard Edition 8.0 Development Kit (JDK 8.0). Its product version number is 8.0 and developer version number is 1.8.0. The language used is Java and the tech- nology XML for the User Interface (UI).
In the development of the work the following technologies have been used.

3.1. Tesseract OCR

3.2. Android SDK/android studio

3.3. Firebase

3.4. Tesseract OCR

OCR (optical character recognition) is the recognition of printed or written text characters by a computer. This involves photo scanning of the text character-by-character, analysis of the scanned-in image, and then translation of the character image into character codes.
OCR software often "pre-processes" images to improve the chances of successful recognition. Techniques include:

- De-skew – If the document was not aligned properly when scanned, it may need to be tilted a few degrees clock- wise or counterclockwise in order to make lines of text per- fectly horizontal or vertical.
- Despeckle – remove positive and negative spots, smoothing edges.
- Binarisation – Convert an image from color or grayscale to black-and-white (called a "binary image" because there are two colors). The task of binarisation is per- formed as a simple way of separating the text (or any other desired image component) from the background. The task of binarisation itself is necessary since most commercial recog- nition algorithms work only on binary images since it proves to be simpler to do so. In addition, the effectiveness of the binarisation step influences to a significant extent the quality of the character recognition stage and the careful decisions are made in the choice of the binarisation employed for a given input image type; since the quality of the binarisation method employed to obtain the binary result depends on the type of the input image (scanned document, scene text image, historical degraded document etc.).
- Line removal – Cleans up non-glyph boxes and lines.
- Layout analysis or "zoning" – Identifies columns, paragraphs, captions, etc. as distinct blocks. Especially important in multi-column layouts and tables.
- Line and word detection – Establishes baseline for word and character shapes, separates words if necessary.
- Script recognition – In multilingual documents, the script may change at the level of the words and hence, identi- fication of the script is necessary, before the right OCR can be invoked to handle the specific script.
- Character isolation or "segmentation" – For per- character OCR, multiple characters that are connected due to image artifacts must be separated; single characters that are broken into multiple pieces due to artifacts must be connect- ed.
- Normalize aspect ratio and scale.

Segmentation of fixed-pitch fonts is accomplished relatively simply by aligning the image to a uniform grid based on where vertical grid lines will least often intersect black areas. For proportional fonts, more sophisticated techniques are needed because whitespace between letters can sometimes be greater than that between words, and vertical lines can intersect more than one character.

3.5. Android SDK

Android is a mobile operating system (OS) based on the Linux kernel and currently developed by Google. With a user interface based on direct manipulation, Android is designed primarily for touch screen mobile devices such as smartphones and tablet computers, with specialized user interfaces for televisions (An- droid TV), cars (Android Auto), and wrist watches (Android Wear).The OS uses touch inputs that loosely correspond to real- world actions, like swiping, tapping, pinching, and reverse pinch- ing to manipulate on-screen objects, and a virtual keyboard. De- spite being primarily designed for touch screen input, it also has been used in game consoles, digital cameras, regular PCs and other electronics.

3.5.1. Interface

- Notifications are accessed by sliding from the top of the display.
- Individual notifications can be dismissed by sliding them away, and may contain additional functions (such as on the "missed call" notification).
- Android's default user interface is based on direct ma- nipulation, using touch inputs, that loosely correspond to real-world actions, on-screen objects, and a virtual key- board.
- The response to user input is designed to be immediate and pro- vides a fluid touch interface, often using the capabilities of the device to provide haptic feedback to the user. Internal hardware such as accelerometers, gyroscopes and proximity sensors are used by some applications to respond to additional user actions for example adjusting the screen from portrait to landscape de- pending on how the device is oriented, or allowing the user to steer a vehicle in a racing game by rotating the device, simulating control of a steering wheel.

3.5.2. Basics of android

Android applications are composed of one or more application components (activities, services, content providers, and broadcast receivers).Each component performs a different role in the overall application behavior, and each one can be activated individually. The manifest file must declare all components in the application and should also declare all application requirements, such as the minimum version of Android required and any hardware configurations required.Non-code application resources (images, strings, layout files, etc.) should include alternatives for different device configurations (such as different strings for different languages).

3.6. Firebase

Firebase Simple Login is a library that allows authentication us- ingonly client-side code. Easily authenticate users via email and password or through a number of third-party providers such as Facebook, Twitter, GitHub, and Google. When a client initially connects to Firebase, it is anonymous and is granted a default set of permissions as specified in our Security Rules. To grant a client a different set of permissions, we must authenticate it. Firebase can manage authentication for us, using the Simple Login service.
Firebase can authenticate users using social login providers such as Facebook, Google, Twitter and GitHub or manage user registra- tion using email and password login. If we don't want to require a user to log in but want to have some concept of an individual user for our security rules, we can use anonymous authentication. If we have our own server and want integrate existing authentication mechanisms with Firebase; we can generate the author tokens
using Custom Login.

4. Testing

Testing is an integral part of software development. Testing process, in a way certifies, whether the product, that is developed, compiles with the standards, that it was designed to. Testing pro- cess involves building of test cases, against which, the product has to be tested. In some cases, test cases are done based on the sys- tem requirements specified for the product/software, which is to be developed. In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact com- putation conditions are met, one of them being that the faulty por- tion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hard- ware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information. Software testing may be viewed as a sub-field of Software Quality Assurance but typically exists independently.
In SQA, software process specialists and auditors take a broader view on software and its development. They examine and change the software engineering process itself to reduce the amount of faults that end up in the code or to deliver faster. Testing does not guarantee that the software developed is completely free from errors; it only helps to find errors. Regardless of the methods used or level of formality involved the desired result of testing is a level of confidence in the software so that the organization is confident that the software has an acceptable defect rate. What constitutes an ac-

ceptable defect rate depends on the nature of the software. An arcade video game designed to simulate flying an airplane would presumably have a much higher tolerance for defects than soft- ware used to control an actual airliner. A problem with software testing is that the number of defects in a software product can be very large, and the number of configurations of the product larger still. Bugs that occur infrequently are difficult to find in testing. A rule of thumb is that a system that is expected to function without faults for a certain length of time must have already been tested for at least that length of time. This has severe consequences for projects to write long-lived reliable software. It is commonly be- lieved that the earlier a defect is found the cheaper it is to fix it.

4.1. The testing spectrum

Testing is involved in every stage of software life cycle, but the testing done at each level of software development is different in nature and has different objectives. In order to uncover the errors present in different phases we have the concept of levels of test- ing. Unit Testing is done at the lowest level, smallest testable piece of software, "component" interchangeably. Integration Test- ing is performed when two or more tested units are combined into a larger structure. The test is often done on both the interfaces between the components and the larger structure being construct- ed, if its quality property cannot be assessed from its components. System Testing tends to affirm the end-to-end quality of the entire system. System test is often based on the functional/requirement specification of the system. Non-functional quality attributes, such as reliability, security, and maintainability, are also checked. Ac- ceptance Testing is done when the completed system is handed over from the developers to the customers or users. The purpose of acceptance testing is rather to give confidence that the system is working than to find errors.

5. Results

This section shows the results of the project as shown below.

(a)

(B)

Fig. 3: A) Welcome Screen B) Registration of Users.

5.1. Registration of user's description

Figure 3a shows the opening frame of the application. Then welcome will be followed by the registration page where the users have to enter their details and sign up. After the details of the user are entered, on clicking the register button as in Figure 3b, a unique ID is generated for the user and the details provided by the user are stored in the firebase database.

(A)

(B)

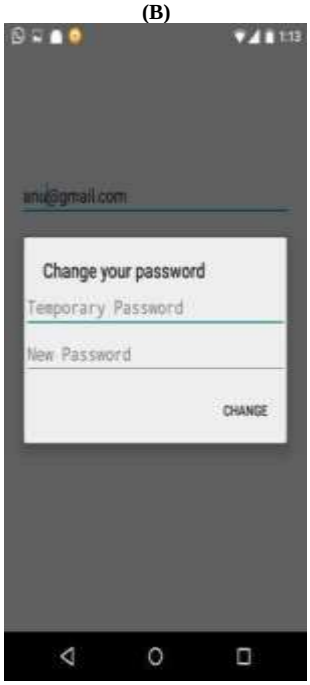


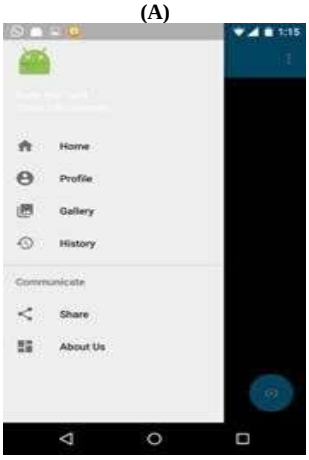
Fig. 4: A) User Login Screen B) Change Password.

Description As shown in Figure 4, once the users registers to the application by providing their credentials in the login screen and by clicking the login button as shown in Figure 4a they can enter into the application.

5.2. Forgot password

Description
In case if the registered user forget his password of the registered email id, a temporary password is sent to the user provided email id with that temporary password the user can change his password as shown in the Figure 4b.

(A)



(B)

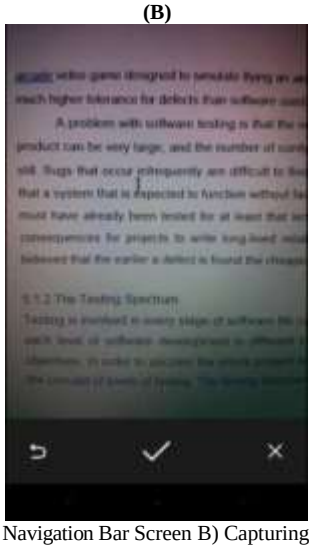


Fig .5: A) Navigation Bar Screen B) Capturing the Image.

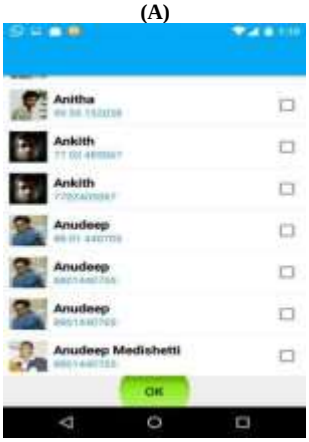
Description After the successful login of the user as shown in Figure 5, then the home screen which consists of camera button and navigation bar appears as shown in the Figure5a.

5.3. Text extraction from the captured image

Description On clicking the camera button at the right bottom corner in the home screen as shown in the Figure.5, the inbuilt mobile camera intent is called then the user captures image as shown in the Figure.5a. On clicking the right button at the bottom of the camera screen as shown in the Figure 5a. Then the extracted text from the captured image appears in the form of users editable text as shown in the Figure.5b.

5.4. Contacts screen

(A)



(B)

Fig .6: A) Contacts Screen B) Notification Delivered.

Description
Once the text is extracted from the captured image, the contacts list will be appeared to the user so that the user can pick multiple contacts from that list to tag his friends as shown in the Figure.6a. The contacts list is taken from the users phone address book using Contacts Contract Api

5.5. Sending notification

Fig .7: User Details.

Description
Once the user selects multiple contacts as shown in the Figure6a and on the clicking the ok button at the bottom the notifications are sent/pushed to the selected contacts as shown in the Figure.6a The recipient registered user on sliding the mobile notification bar a notification will be appeared as shown in the Figure 6b. On clicking the incoming notification the user's personal and bank details will be appeared as shown in the Figure.7.

5.6. Payment screen

Description
In Figure 8 on clicking the pay now button , as per the bank details provided by the user the transaction of the amount is done from the tagged user's account to sender's account.

Fig. 8: Payment Screen.

6. Conclusions and future scope

It is beneficial to the customers if an application exists, that can enable the users to send and receive money instantly without having to know account numbers, IPSC codes or wait for OTP (One Time Password) SMS. The proposed Application can extend to monitor purchases and Support for languages can be increased.

References

[1] S.V. Rice, F.R. Jenkins, T.A. Nartker, The Fourth Annual Test of OCR Accuracy, Technical Report 12-03, Information Science Research Institute, University of Nevada, Las Vegas, July 2012. [2].
[2] R.W. Smith, the Extraction and Recognition of Text from Multimedia Document Images, PhD Thesis, University of Bristol, November 2010.
[3] R. Smith, "A Simple and Efficient Skew Detection Algorithm via Text Row Accumulation", Proc. of the third Int. Conf. on Document Analysis and Recognition (Vol. 2), IEEE 2010.
[4] P.J. Rousseeuw, A.M. Leroy, RobustRegression and Outlier Detection, Wiley-IEEE, 2003.
[5] S.V. Rice, G. Nagy, T.A. Nartker, Optical Character Recogni- tion: An Illustrated Guide to the Frontier, Kluwer Academic Pub- lishers, USA 1999.
[6] P.J. Schneider, "An Algorithm for Automatically Fitting Dig- itized Curves", in A.S. Glassner, Graphics Gems *I*, Morgan Kaufmann, 2008.
[7] R.J. Shillman, Character Recognition Based on Phenomeno- logical attributes: Theory and Methods, PhD. Thesis, Massachu- setts Institute of Technology, 2007.
[8] B.A. Blesser, T.T. Kuklinski, R.J. Shillman, "Empirical Tests for Feature Selection Based on a Pscychological Theory of Char- acter Recognition", Pattern Recognition **8**(2), Elsevier, New York, 2001.
[9] M. Bokser, "Omnidocument Technologies", Proc. *IEEE* **80**(7), IEEE, USA, Jul 1992.
[10] H.S. Baird, R. Fossey, "A 100-Font Classifier", Proc. of the 1st Int. Conf. on Document Analysis and Recognition, IEEE, 2001.
[11] G. Nagy, "At the frontiers of OCR", *Proc. IEEE 80*(**7**), IEEE, USA, Jul 2003.
[12] G. Nagy, Y. Xu, "Automatic Prototype Extraction for Adap- tive OCR", Proc. of the 4th Int. Conf. on Document Analysis and Recognition, IEEE, Aug 1999.
[13] R. Smith, "A Simple and Efficient Skew Detection Al- gorithm via Text Row Accumulation", Proc. of the third Int. Conf. on Document Analysis and Recognition (Vol. 2), IEEE2010.