MINOR PROJECT I

RE-FORM-IT

Submitted by:
Tripti Shukla (9917103253)
Prashant Dixit (9917103209)
Aditya Pandey (9917103039)

Submiited to:
Mr. Raju Pal

Department of Computer Science Engineerng
Jaypee Institute of Information Technology, Noida,

Acknowledgements

I would like to place on record my deep sense of gratitude to **Mr. Raju Pal**, Professor, Jaypee Institute of Information Technology, India for his generous guidance, help and useful suggestions as our mentor for the minor project.

Tripti Shukla (9917103253)
Prashant Dixit (9917103209)
Aditya Pandey (9917103039)

Abstract

The idea behind this project is to provide an easily accessible, user-friendly and highly efficient cross-platform mobile application for providing form-related information and form-filling functionality which can be used by people across India. There have been so many instances where we find uneducated people seeking help in banks, RTO offices and other government offices to fill up forms that they do not understand due to lack of knowledge. Going by the statistics, India alone has 300 million active smartphone users (report 2017) and literacy rate sums up to be around 74 percent (in 2017) which hints that there is still a large bracket of uneducated smartphone user in India and that right there is our primary target audience [3]. The methodologies used in this machine learning based android application are contour detection, four-point perspective transform, Hough transform, morphological techniques, PyOCR, text-to-speech conversion, etc.

ReFormIt is built with the aim of providing complete assistance in filling a form written in English with no proper guidelines and hints, a problem mostly encountered by the older generation and less educated class of our country.

The project is divided into 5 parts, firstly a user authentication service that enables a user to login to his/her own dashboard wherein he/she can maintain a collection of all the previously filled forms, secondly the functionality to capture a form image and pre-process it if it is a form or not, thirdly to extract text from the captured image and check for the form in the database, then creating bounding boxes around the regions of probabilistic input fields and last but not the least providing the functionality of filling the form using text-to-speech and speech-to-text API.

Firebase real-time database [2] has been used for managing the dataset and backend of the Kotlin [1] based android application, text-to-speech android library for text-speech conversion and Chaquopy for integrating machine learning models in the android application [4].

Contents

List of Figures

Abbreviations

- OCR: Optical Character Recognition
- ML: Machine Learning
- ML: Machine Learning
- TF: TensorFlow
- SDK: Software Development Kit
- TFLite: TensorFlow Lite

Chapter 1

INTRODUCTION

India being a developing nation witnesses a literacy rate ranging between 63 percent to 93 percent but still the population who can read properly is less than 50 percent[3]. There are a large number of people who feel the necessity of assistance while dealing with any task that involves reading and writing. One major area of concern is the problem of dealing with form filling by the local elite in various sectors.

With the advancement of education in India, the banking sector, government sector, etc. have also seen a rapid advancement in the functioning as well as working process. English language has become a vital part of the office proceedings and structure. But a majority of Indians most prominently the lower poorer class of the Indian society is still not very well verse with the English language and thus arises the problem of language barrier in the official system. People are generally found confused with how to proceed and try to seek help from others. The problem of filling a form written in English with no proper guidelines and hints to proceed is mostly found difficult by the lower class of India who are not fluent in the English language.

To provide the public with an Android application that makes the process of filling various forms used in different sectors of India an independent and easy task. It acts as a boon for the older generation and the less educated class who find form filling a difficult task and cannot accomplish it independently. The app enables the user to click an image and then perform its task of detecting a form, extracting text and filling the form with a very small amount of user input. The project will use technologies such as image-text extraction, speech-to-text conversion, pdf editing and maintaining a dataset of various forms used in different office sectors in India.

Now the main issue and the driving factor behind our idea was that the designers of these forms are people who are highly educated and cannot relate with the local folks who are the actual users of these forms. This disbalance in the level of understanding leads to a high degree of discomfort for the actual users.

The actual motivation to build something like this comes from the scenario where we ran into an old lady in a bank which was overcrowded and this lady could not figure out the details that she was supposed to fill in a form meant for opening an account. She asked us if we can help her out. This gave me a sense of dependence these people hold on others for errands as easy as these. Let us try to understand the difficulties one faces going through any such situation in his/her daily life: People generally have to go Google through thousands of pages over and over just to get a single piece of information about forms. Physical presence while filling the form (getting the form, filling it and then submitting it). If once gone wrong, not being able to revert the filled information. A sense of helplessness and decrement in the self-respect of oneself.

The project directly serves to the needs of the common folks who are not fluent in English language and are not aware of the procedure of form filling within different sectors in India. By using technologies mentioned above, we intend to make the process of filling and submitting a form an easy endeavour for every person. The app aims to empower the less educated class and the older generation and make them independent.

Chapter 2

Background Study

In India, every sector be it public or private has dozens of forms for every single activity that takes place within these sectors. The major difficulties faced by the common people due to the unstructured and poorly documented form system include, surfing through thousands of pages over and over just to get a single piece of information about forms, physical presence while filling the form and many more. We observed that the majority of people are highly dependent on others for filling up forms. We tried to cater this problem by implementing an Android app which can be used on a large scale by Indian people and have paved the way for a much more digital and smart India, where people can find their every day necessities within the grasp of their pockets. Following are some of the research papers that helped us in adopting a certain course of action into our project:

Paper 1: TEXT EXTRACTION FROM BILLS AND INVOICES

This approach filters out the unnecessary noise from image by using Contour Detection Algorithm and in order to improve the quality of the image, a four-point perspective transform is applied in the image . The refined image is now passed in the Tesseract OCR engine where segmentation process is applied here. But this approach fails in capturing the texts from the bills or invoices that are handwritten as tesseract uses segmentation technique that recognizes first character in the line and then according to the position of the first character tries to read the whole line[6]

Paper 2: USAGE OF HOUGH TRANSFORM FOR EXPIRY DATE EXTRACTION VIA OPTICAL CHARACTER RECOGNITION

This paper demonstrates a typical approach to use Hough Transforms in image pre-processing by limiting it just to rotation of images and a sliding window approach where a variable size window is passed along the image and iterated over and over until the algorithm returns only the highest value of date to minimize the chance of false positives [7]. Furthermore, the technique used in this paper can be improved with focus on the optimization of the algorithm for the detection of expiry dates and addressing the limitations caused by small labelled datasets currently available.[5]

Paper 3: APPLYING TESSERACT-OCR TO DETECTION OF IMAGE SPAM MAILS

This paper generates training images from real spam mail images and examines the minimum numbers of training images and enlarges images that are required for efficient training of TesseractOCR. If one of TesseractOCR for each word prototype recognizes spam words in the

image, the mail is recognized as a spam mail. This paper focuses more on the OCR techniques, counter measuring false positives and there is still a lot of room for improvement by using various image pre-processing techniques which this paper lacks of. Therefore, new techniques such as pyOCR can be used to boost the accuracy of the existing OCR model [7].

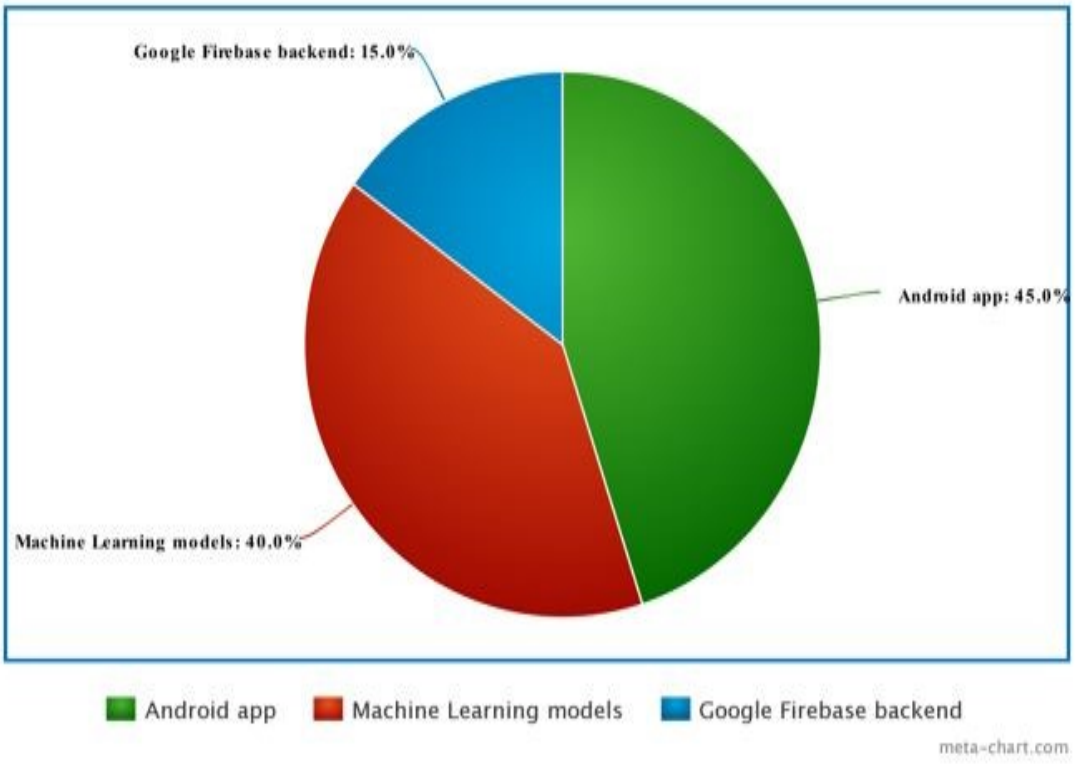


Figure 2.1: Project Division over Tech-Stack

Chapter 3

Requirement Analysis

3.1 User Interface

User Interface for the project is provided through an Android application. User can click the image of the form or select an image from mobile phone's gallery as input. In turn he/she will be given all the possible information regarding the form and later will be provided with an option to auto-fill or manually fill the form from the app.

To provide a communication interface between the Android app and the Machine Learning model we used Chaquopy which is a plugin for the android gradle, which is a lightweight solution for mobile and embedded devices. Our ML models work through python scripts in the backend of the application and are used as assets in our Android app by adding Chaquopy dependencies in our Android Studio SDK.

3.2 Hardware Interface

- Any Android mobile device with an active internet connection
- 4 GB RAM or Higher
- Atleast 8 GB storage on device with SSD support (for testing on system devices)
- GPU (for testing on system devices)
- Atleast 2 GB minimal storage on android device

3.3 Software Interfaces

- Android Studio
- Jupyter Notebook
- Google Firebase
- Python
- Chaquopy 6.3

Chapter 4

Detailed Design

The workflow has been implemented with the help of a Kotlin based android application that uses Firebase real-time database for hosting the dataset and Firebase services for handling some backend functionalities like authentication etc that are easier to implement using Firebase rather than writing the scripts in some other language.

The machine learning scripts for pre-processing and text-extraction have been written in python that have then been integrated with the android app using Chaquopy which is a plugin for the android gradle. The scripts have been written in python because python provides enormous image processing capabilities due to its built-in libraries like Opencv-Python, Pillow, Matplotlib, Pandas, Numpy, etc. Chaquopy embeds a python interpreter inside the android gradle that helps in executing python scripts within the application. Further we have used android's library for text-to-speech conversion as it was easier to integrate an inbuilt library than some external API.

The main reason behind using Kotlin for application development was that being entirely interoperable with Java, you get to use Kotlin code from Java and the other way around. It's easy to compile to Native or JavaScript for developing code that can run on iOS, as well.

We have used Android's API for text extraction that provides efficient output as compared to other softwares.

Following is the design of the firebase database integrated in the backend of the project:

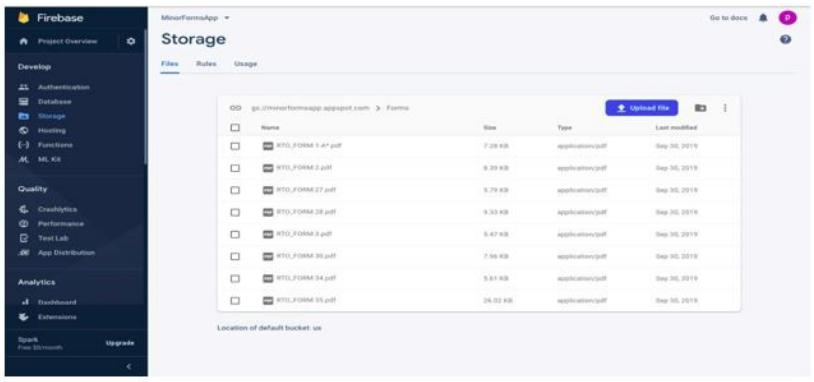


Figure 4.1: Firebase database with the form data set having a specific name format

The following diagram depicts an overview for working of the project:

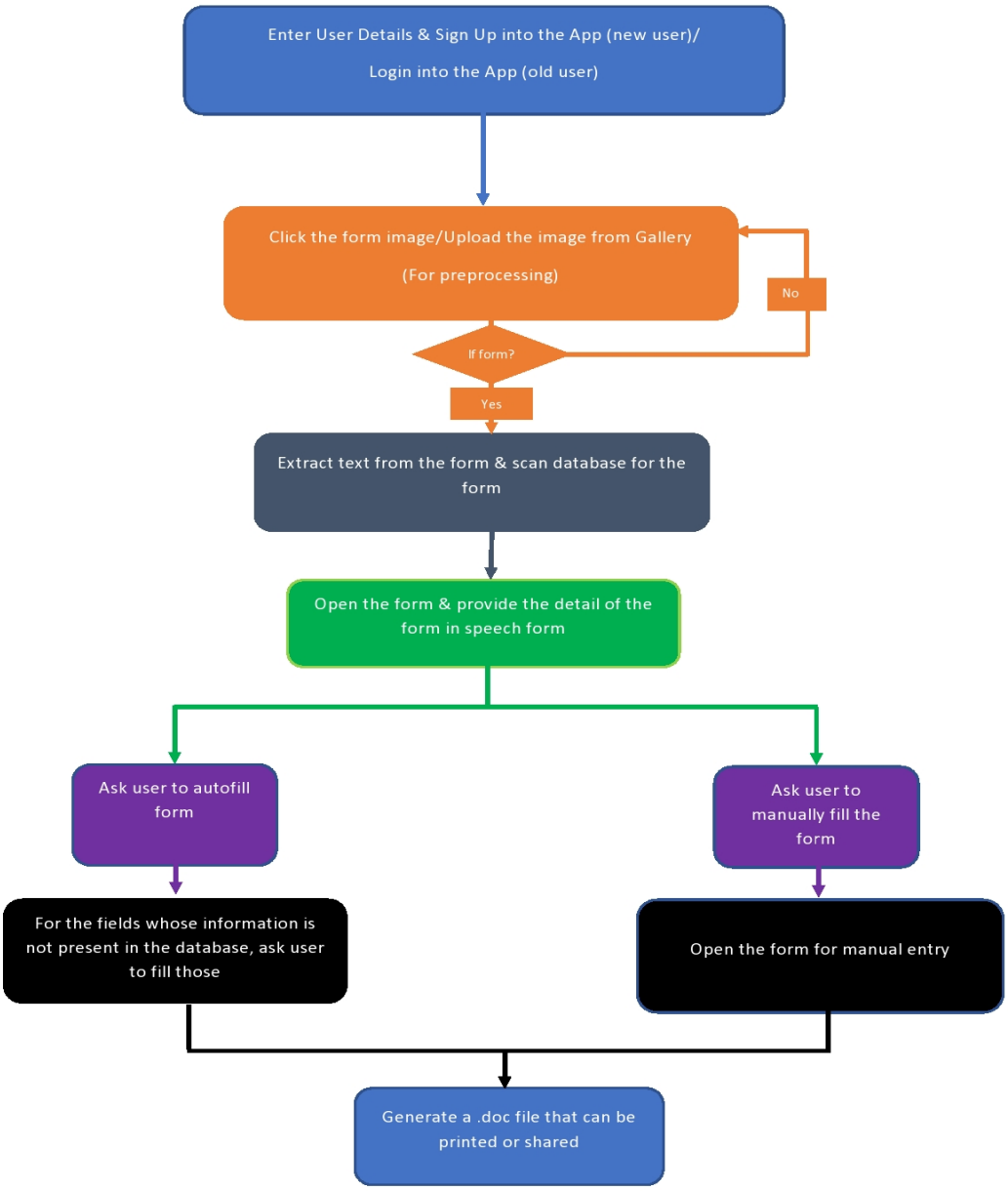


Figure 4.2: Flowchart showing the app workflow

Chapter 5

Implementation

The project mainly comprises of 5 parts:

- Creating user authentication using Firebase Auth Service
- Pre-processing the captured image for form-only app functionality
- Text extraction and searching for the form in the database
- Generating bounding boxes around probabilistic input fields
- Creating Speech-to-text library functionality for automatic form filling

Creating user authentication using Firebase Auth Service

We created the login and signup modules of the app in Kotlin and used the Firebase Auth Service for creating a profile for each user in the app. Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to the app. We used email authentication from this service. This was done to make sure that every user has a personalized experience using the app. and can access the collection of forms he/she has filled in the past.

After the user has been logged in he/she can either access his/her previous forms or capture an image of a new form.

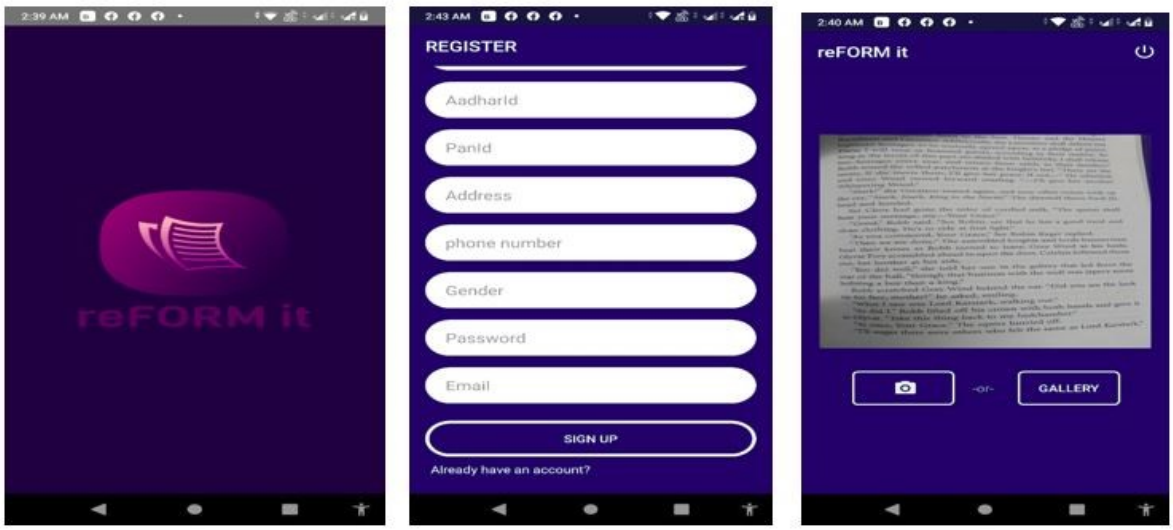


Figure 5.1: Snapshots of the UI of the application

Pre-processing the captured image for form-only functionality (noble method)

After the image is captured, it is processed through our pre-processing model that checks if the image is that of a form or not. The image is processed through various stages and finally passed through through morphological operations. Following are the steps followed for this methodology:

Removing noise from image and converting it into binary form

It is the first step towards contour detection and adjusting the perspective of the image. Noise is removed by converting the image into grayscale, applying Gaussian blur on it and then finding the edges. Edges correspond to a change of pixels' intensity. To detect it, the easiest way is to apply filters that highlight this intensity change in both directions: horizontal (x) and vertical (y).

Figure 5.2: Snapshots of pre processing of the clicked image

Finding the largest contour and adjusting the perspective

After finding the edge map, we find all the contours from it and sort them according to their size in decreasing order. Now the largest contour is approximated and if it has four-points then it is the required image of the form. Then a four point perspective transform is applied to both the original image and grayscale image to obtain a top-down birds eye view of the form.

Implementing morphology operations and finding horizontal line segments

Morphology operations are applied after adaptive thresholding of the image. Morphology is a set of image processing operations that process images based on predefined structuring elements known also as kernels. Two of the most basic morphological operations are dilation and erosion. Dilation adds pixels to the boundaries of the object in an image, while erosion does exactly the opposite. After the horizontal lines are found in the image, Hough Transform is applied to get the x and y coordinates of the line segments. Now if more than a threshold value of lines are approximately of the same size then it is estimated that the image is of a form.

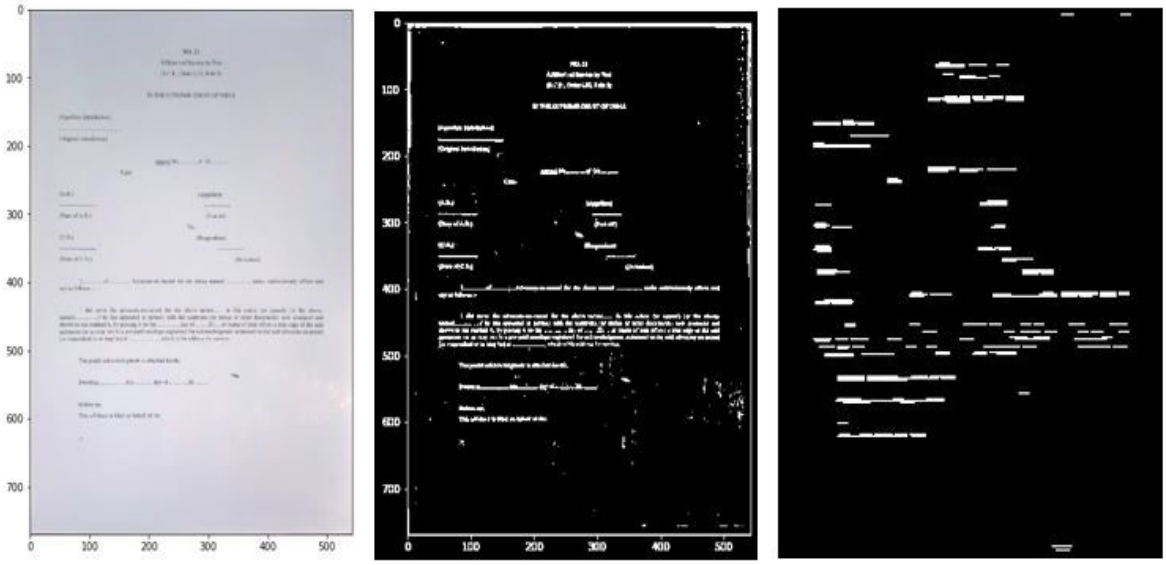


Figure 5.3: Snapshots of Processing the data using ML techniques

Preprocessing integration with android application

The preprocessing python script is integrated with the android application using Chaquopy which is a plugin for the android gradle. After making necessary configuration in the build.gradle file almost 90 percent of the python libraries can be used in an android application while running a python script in the backend.

Text extraction and searching for the form in the database

After it has been found that the image is of a form our next step is to extract text from the form. For this we have applied Android's text extraction API that provides highly efficient results as compared to the later. For this we made sure to create a collection of forms to our realtime Firebase database which provides easy integration with android applications and fast processing in realtime.

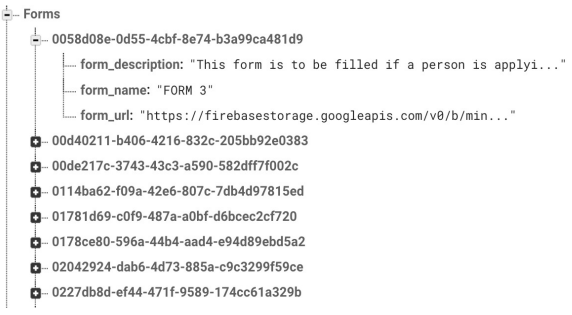


Figure 5.4: Database schema

Now the text is extracted from the captured image and the first string of the extracted text that contains the name of the form is appended with the department name of the form and then the database is searched for the required form.

After the form has been detected from the database, the description of the form will be read out to the user for his/her assistance. The next step is to create bounding boxes around the

probabilistic input fields and then enable the user to fill the form fields using the text-to-speech android library and also allow manual form filling.

Class Diagram showing all the implementation functions and attributes of each component:

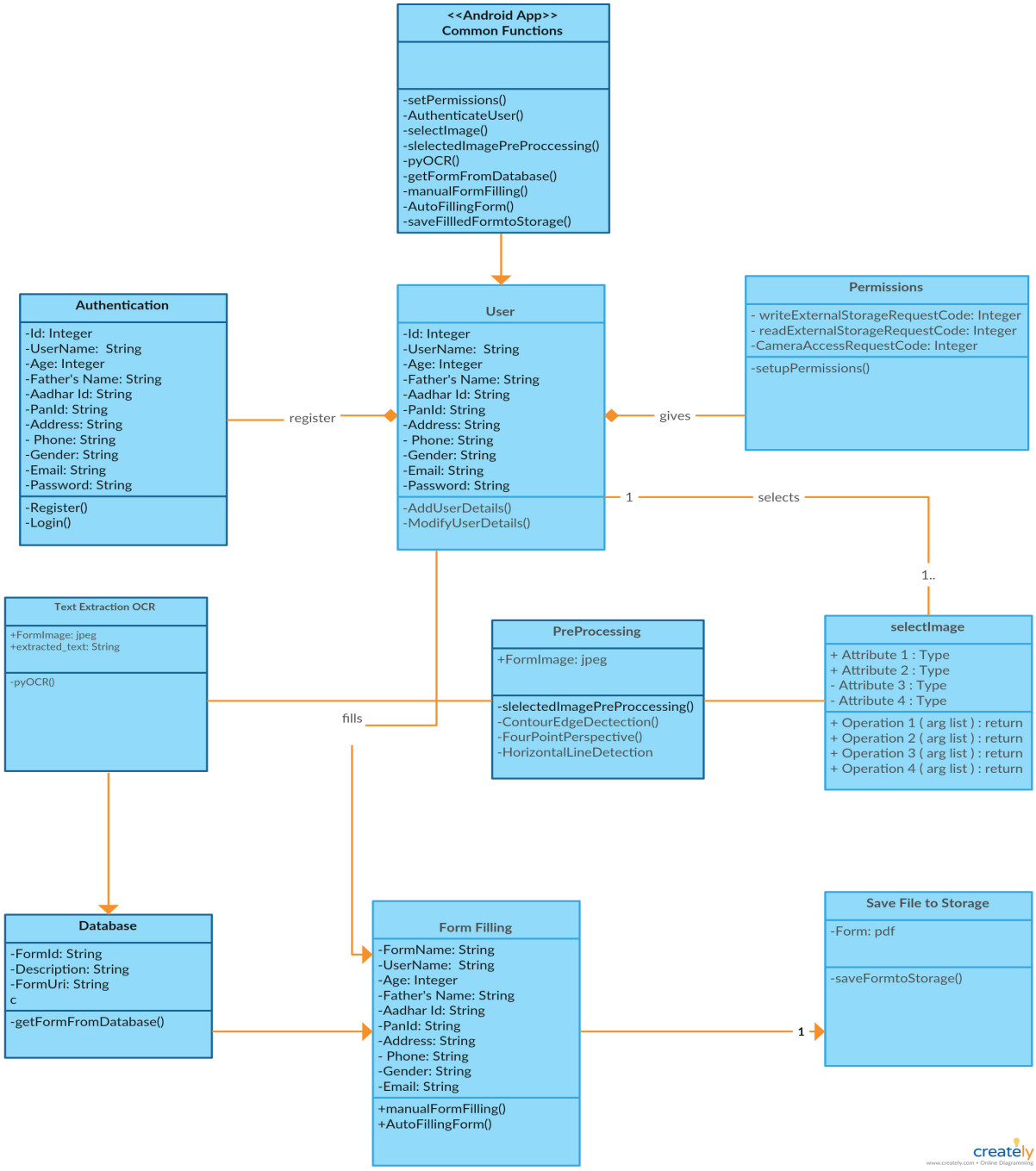


Figure 5.5: Class Diagram

Chapter 6

Experimental Results and Analysis

The designed application lead to the development of a method to check if the clicked image is of a form or not and then enabled the user to fill the form with ease along with a full description of it in audio format so as to provide complete information about it.

Following are the output screens from the final application:

If the clicked image is processed as a non-form document then a toast is generated saying "Form Not Detected" and no action takes place further.

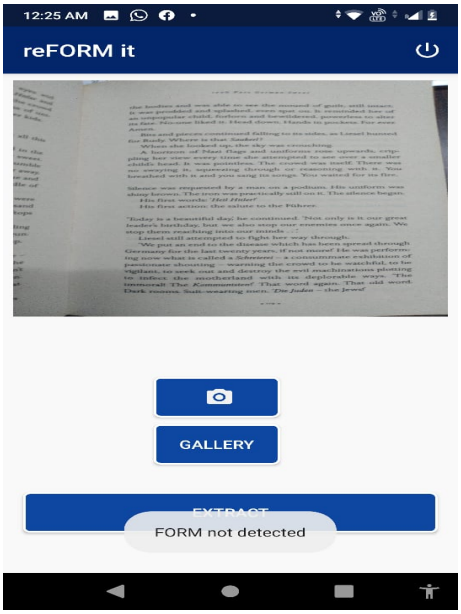


Figure 6.1: Application detecting the image is not of a form

After an image is clicked and processed through the ML model and once detected that the image is of a form, text is extracted from the form and the name is searched through the database.

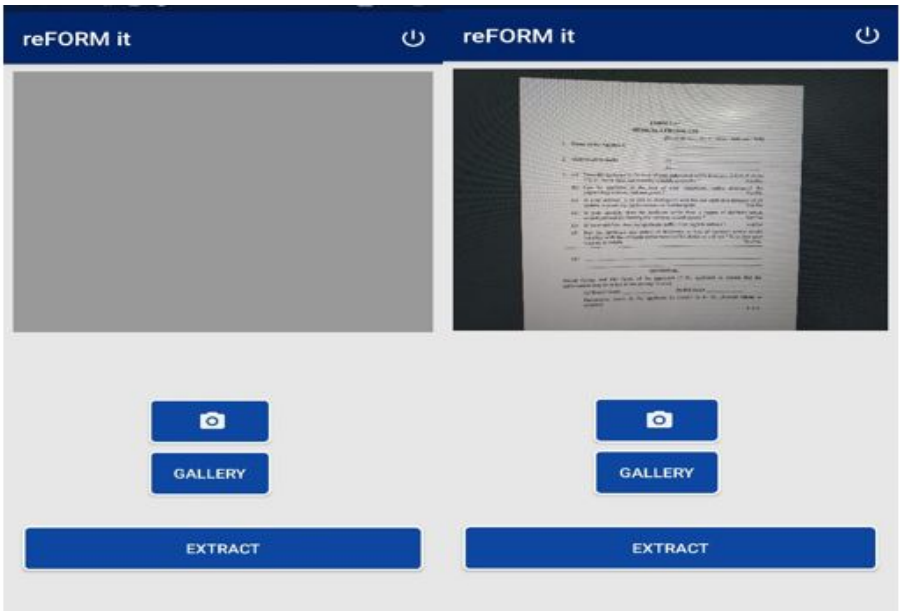


Figure 6.2: Clicked image being displayed while pre-processing in the backend

Once we have the information about the form from the database the information is provided to the user in speech and text form and an editable form area is displayed with the form text which can be filled by the user with manual filling in the current verson.

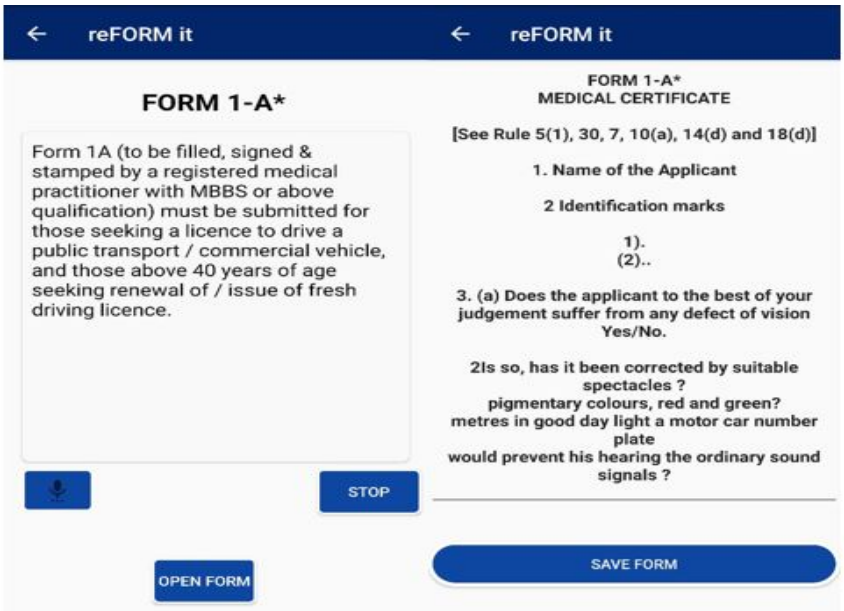


Figure 6.3: Form description as provided by the database and an editable form displayed

6.1 Testing

We carried out various tests at different stages of the application development. Following are the major ones listed below:

Testing Report

Action	Objective	Expected Output	Current Output
Preprocessing and Text extraction test	To test if preprocessing & text extraction scripts compiles successfully	All the scripts work in unison to give one output	Individual attainment of the outputs from each script
Integration Test	To test through TFLite if all the gathered python scripts merge successfully	All the python scripts merged perfectly and give output on the android platform	All the python scripts work on virtual environment
Android App working	To test if android app works fine and integrates with Firebase Backend	Android app Responding to users queries by communicating with scripts and Firebase backend.	Android App frontend is ready. Work in progress for rest of the API and integration.
Android App Security	To ensure security of users data by harnessing the power of Firebase Authentication Services.	As user enters the personal details it is shielded by Firebase Authentication Services.	User inputs are currently stored in the database as expected.

Figure 6.4: Testing Report with Results

Chapter 7

Conclusion and Future Scope

The main impact this application will cast is with providing information and filling of forms, with this functionality on our fingertips Re-Form-It will benefit a large bracket of people. This app by far can pre-process the clicked images i.e. Once you have taken a picture of the form, our app can filter out as much noise as possible from the image and selecting only the required image of form. Then it will extract text out of the form to later search it in the database and do the rest of the function.

We have successfully implemented the first version of the app with successful pre-processing, text extraction and text-to-speech conversion functionalities.

Future improvements which can be seen on Re-Form-It can be like, link of a form (some google form or the ones present on some website) can be directly pasted on the dashboard search bar redirecting it to the actual form. Future scope involves text extraction and text to speech conversion the form can be auto-filled by asking out the necessary details of the user in speech format (speech to text conversion). The USP of the application solely depend on database and if a strong database is built the app can be used on a much larger scale. Algorithms can be refined for better response from app.

Bibliography

[1] Android official documentation. https://developer.android.com.

[2] Firebase. Google Firebase Official documentation.

[3] Literacy in india. https://en.m.wikipedia.org/wiki/Literacy$_in_India$.

[4] Tensorflowlite official documentation. https://www.tensorflow.org/lite/guide/get$_started$.

[5] D. Scazzoli, G. Bartezzaghi, D. Uysal, M. Magarini, M. Melacini, and M. Marcon. Usage of hough transform for expiry date extraction via optical character recognition. In *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, pages 1–6, March 2019.

[6] H. Sidhwa, S. Kulshrestha, S. Malhotra, and S. Virmani. Text extraction from bills and invoices. In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pages 564–568, Oct 2018.

[7] D. Yamakawa and N. Yoshiura. Applying tesseract-ocr to detection of image spam mails. In *2012 14th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–4, Sep. 2012.

Problem Statement

- To create an application such that no one has to look forward to google down thousands of pages to get information about a single form reducing wastage of time and energy.
- To provide assistance to the less educated class of the society in filling up forms independently.
- There are no such products available in the market which can specifically help us in not only knowing the details of the form but also filling it up ,automatically or manually.

STATE OF THE ART

- There have been a large amount of work in the field of line detection and text extraction using optical character recognition. Many research papers have been already implemented that implement one or the other methodologies.
- "Text Extraction from Bills and Invoices" mainly deals with contour detection in preprocessing and text extraction which helped us in defining the methodology for our implementation.
- "Usage of Hough Transform for Expiry Date Extraction via Optical Character Recognition" presents a typical implementation of Hough transform that has been harnessed by us in a different way so as to provide nobility to our implementation.
- "Applying Tesseract-OCR to detection of image spam mails" applies a methodology to detect certain specific words from the input image, a technique we may work upon.

LIMITATIONS

- Houghline Transform does not provide high accuracy over unprocessed images.
- Contour detection cannot be used over document images unless the perspective has been adjusted.
- A text extraction implementation designed for specific words possesses a limitation over detection.
- Tesseract OCR is an older methodology that provides less accuracy as compared to other on top implementations and neural networks.

OBJECTIVE & WORK DISTRIBUTION

OBJECTIVE

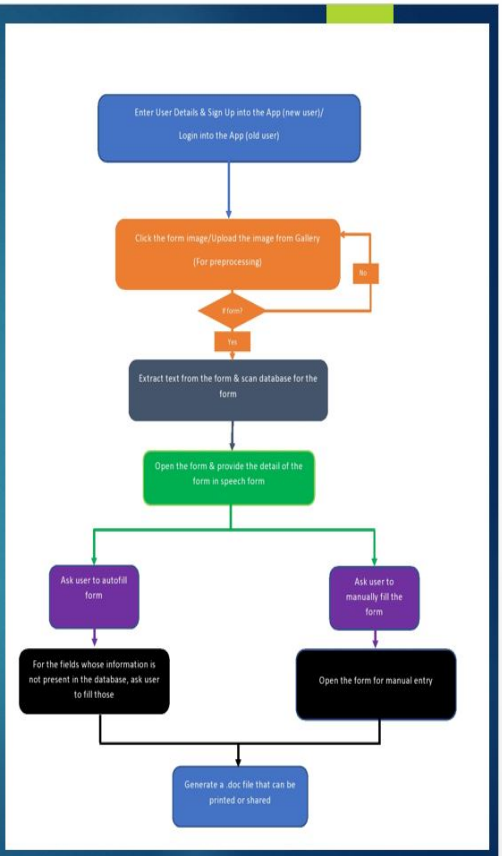
- Going by the statistics, India alone has 300 million active smartphone users (report 2017) and literacy rate sums up to be around 74% (in 2017) which hints that there is still a large bracket of uneducated smartphone users in India leading to the requirement of applications to provide assistance.
- To create a noble methodology based application that help us get information about any government form. Not only it provides every information about the form but it also helps us fill the form automatically or manually as per the user desires. Our goal is to fill the form in a detailed manner from just a clicked image without any external assistance.

WORK DISTRIBUTION

- Initially we read roughly around 20 research papers and journals and picked up a strong base for the implementation.
- We then applied multiple preprocessing techniques to detect the presence of form in an image and then performed text extraction.
- Then an initial android app was developed along with the integration of the database.

PROPOSED DESIGN

- Kotlin-based android application with an integrated Firebase real-time database providing wide capabilities to design a highly professional application.
- User is initially directed to the Register/Login page wherein a user can create an account filling the mandatory information.
- User authentication is performed using sign in with email & password functionality of Firebase Auth.
- After successful login, a dashboard appears to user wherein he can select an image from gallery or click a picture from camera.

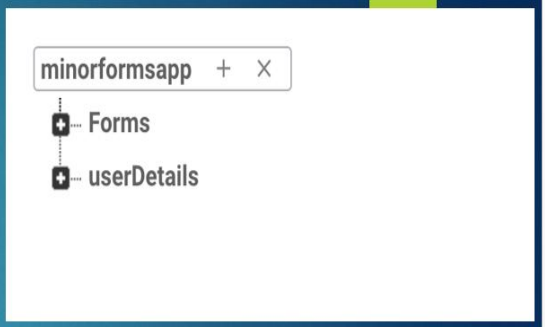


IMPLEMENTATION

- The user selected image is then processed to detect whether the image is a form or not.
- A noble approach of form detection is applied using a combination of contour detection, four point perspective, morphological methodologies and hough transform that provide a high end assurity of an image being of a form or not.
- In order to implement the python code mentioned above in Android Studio we used a tool named chaquopy - A "Python SDK for Android".It's distributed as a plugin for the standard Android build system.
- If the form is not found, then appropriate error Toast messages are given to the user.

IMPLEMENTATION

- If the form is detected successfully in the image, then a search query containing form name is sent to our forms table created on Firebase Real-time database.
- If the search is successful user is taken to Form Description Activity, else appropriate error messages are given to the user.



IMPLEMENTATION

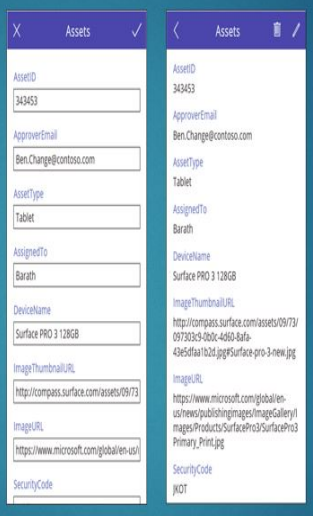
- FormDescription activity provides user with all the necessary information regarding the form fetched from the database
- For user's convenience an option of listening the form description in audio form using Android's inbuilt Text to Speech library is provided as well.
- If the user wishes to fill the form, he/she can click on the Open Form button which creates FormFill Activity.
- In FormFillActivity, user is given the option to manually fill the form extracted from the image using Android EditText.

IMPLEMENTATION

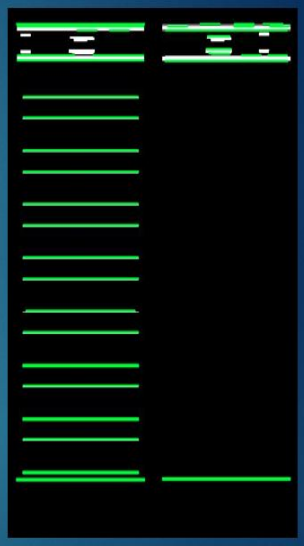
- After manual filling of form by user is done, he/she can save the filled EditText in pdf format by clicking on Save Pdf button.
- For creating a pdf file from edit Text, we have used a Java library named PDFWriter.
- It enables users of **Android** devices that were released under the BSD license to generate simple PDF documents.
- The user filled edit Text is set to Pdf standards using this library and is saved in user's external storage.
- User can now access the filled pdf form stored in his phone's storage which he can upload or submit to intended destination.

RESULT AND ANALYSIS

- Creation of a Kotlin based android application with firebase integration.
- Noble preprocessing methodology leading to a domain specific application (detecting if an image is of a form or not): Returns false for images not of forms and no further action is taken.



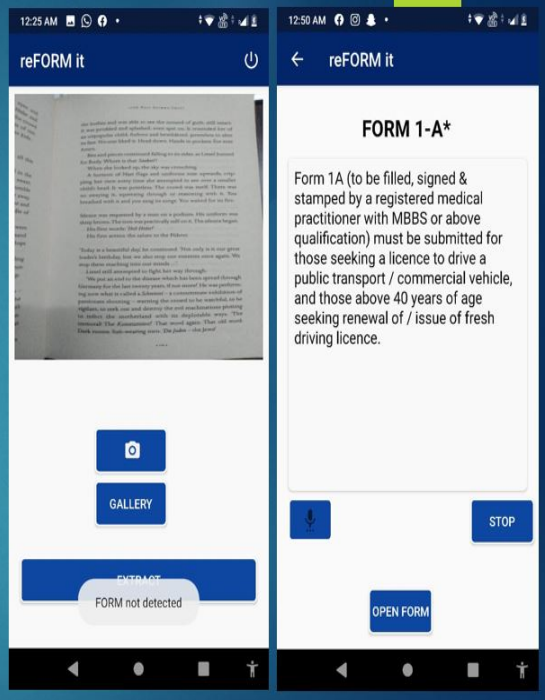
Input Image



Output Image After Pre-processing

RESULT AND ANALYSIS

- After preprocessing if form is not detected then proper error messages are displayed.
- If form detected successfully, user is taken to FormDescription Activity.
- When the user clicks on open form button, he is taken to FormFill Activity.

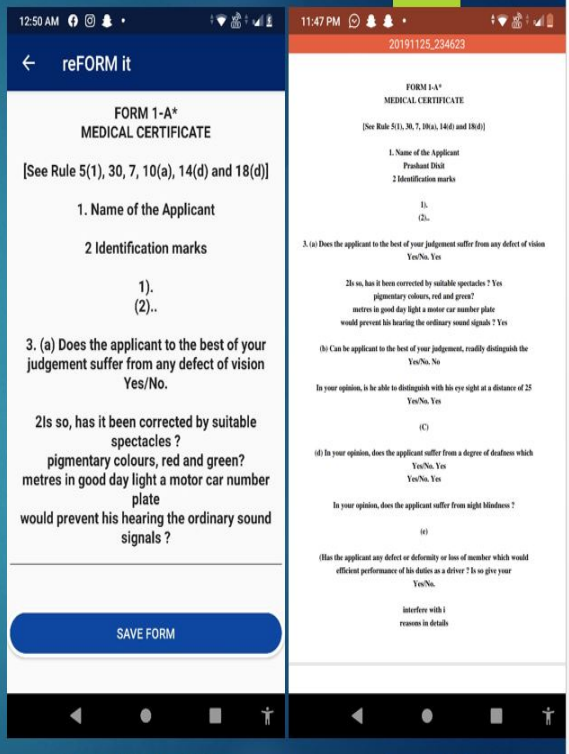


Unsuccessful Detection

Successful Detection

RESULT AND ANALYSIS

- In FormFill Activity, user can fill the form by manually editing the form in the EditText given
- After user has filled the form, he can save the form as pdf.
- The pdf file is stored in user's phone storage.



FormFill Activity

Saved Pdf file

CONCLUSION OF THE REPORT

- This app by far can preprocess the clicked images i.e. Once you have taken a picture of the form, our app can filter out as much noise as possible from the image and selecting only the required image of form.
- With such functionality on our fingertips Re-Form-It will benefit a large bracket of people.
- Then it will extract text out of the form to later search it in the database and do the rest of the function.

FUTURE SCOPE

- Future improvements which can be seen on Re-Form-It can be like, link of a form (some google form or the ones present on some website) can be directly pasted on the dashboard search bar redirecting it to the actual form.
- There with the help of text extraction and text to speech conversion the form can be auto-filled by asking out the necessary details of the user in speech format (speech to text conversion).
- The USP of the application solely depend on database and if a strong database is built the app can be used on a much larger scale.
- Algorithms can further be refined for better response from app.