PSG College of Technology

15Z412 SOFTWARE PACKAGE DEVELOPMENT

PLETHORA

Done by

Deepthishree G S 18Z312

Harshini S 18Z320

Iswaryaa G P 18Z323

Janani R 18Z324

Swetha M 18Z360

COMPUTER SCIENCE & ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

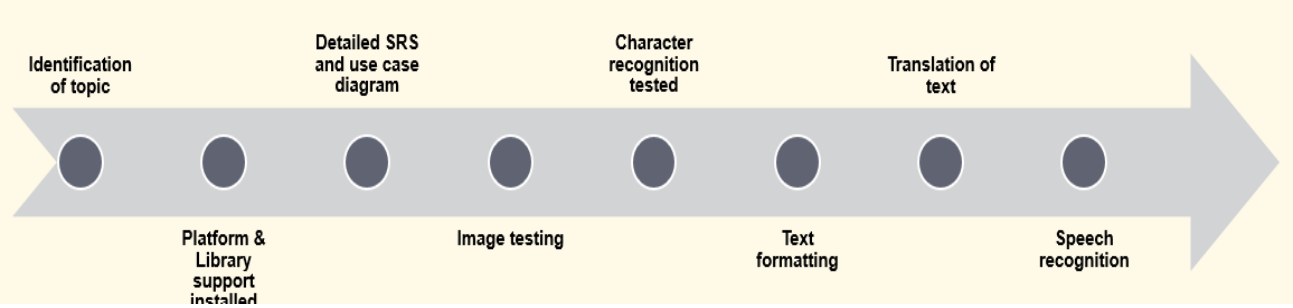
ABSTRACT

In our day-to-day lives, mobile phones have taken over computers and laptops. The smaller is the new trend. We are not so comfortable with word processing through applications in our phone. Not only that, we don't require as much so sophistications too. We decided to take up an application to connect the various forms of small data – that being image, text, languages, speech etc. This is a simple application with further scope of improvement. It is a cross-platform application designed on Windows and primarily meant to be in use for Android. The coding is primarily done in Python with Kivy libraries along APIs of Google for translation and Tesseract for image recognition.

ACKNOWLEGMENT

We sincerely acknowledge our college for giving us hands-on experience on not only working on a project but also as a team. It brought us new experiences. We appreciate the support of our Computer Science department of our college in making the project a success. We also feel great pride in giving credit to the staff who guided us and motivated us, not to go for a mediocre project already done by many students before and to walk an untrodden path and explore new horizons. This project would have been impossible without their kind mentorship. We will also acknowledge the support of the internet and online tools which makes learning new things so easy.

TIMELINE



INDIVIDUAL CONTRIBUTION

Deepthishree GS: She was learning to implement conversion of text and various options like replace. She worked on Login and authentication. She also learned to implement Kivy layouts and design. She initially worked in the conversion of image to text testing. She played a primary role in testing and handling exceptions.

Harshini S: She was instrumental in first making the image to text work as a sample code and improved its efficiency. She helped in sound module. Also, she was good at bringing out Kivy layouts and initial designs. She also showed great interest in making this project come out well and also in meeting the deadline.

Iswaryaa GP: She was amazed and interested in manipulation of images especially, mirror image, grayscale and others. She did a lot for the CV integration and tried image manipulation.

Janani R: She worked in the implementation of Speech recognition modules. She was engaged in bringing out some of the features required for text to text conversion. She also concentrated in the designing of layouts.

Swetha M: She learnt Kivy through tutorials to implement design. She made translations between languages work. She explored layout and implementations for text to text formatting and analysis. She finished the image module She integrated modules through Navigation Drawer. She helped fix bugs and finish documentation.

LIST OF FIGURES

1. USE CASE DIAGRAM..... 6
2. CLASS DIAGRAM..... 8
3. DFD..... 8

LIST OF TABLES

1. LOGIN MODULE TEST 51
2. TEXT RECOGNITION MODULE TEST 51
3. IMAGE MODULE TEST..... 52
4. TEXT FORMATTING MODULE TEST… 52
5. SPEECH MODULE TEST..... 53

TABLE OF CONTENTS

ABSTRACT ii

ACKNOWLEDGMENT ii

TIMELINE ii

INDIVIDUAL CONTRIBUTION iii

LIST OF FIGURES iii

LIST OF TABLES iii

CHAPTER **PAGE NO.**

1.INTRODUCTION 1

1.1. LITERATURE SURVEY 2

2.SOFTWARE REQUIREMENT SPECIFICATION 3

2.1. INTRODUCTION 3

2.1.1. PURPOSE 3

2.1.2. CONVENTION 3

2.1.3. INTENDED AUDIENCE 3

2.1.4. PRODUCT SCOPE 3

2.2. OVERALL DESCRIPTION 4

2.2.1. PRODUCT FUNCTIONS 4

2.2.2. USER CLASSES AND CHARACTERISITICS 4

2.2.3. OPERATING ENVIRONMENT 4

2.2.4. DESIGN AND IMPLEMENTATION CONSTRAINT 4

2.3. EXTERNAL INTERFACE REQUIREMENTS 5

2.3.1. USER INTERFACES 5

2.3.2. HARDWARE INTERFACES 5

2.3.3. SOFTWARE INTERFACES 5

2.4. SYSTEM FEATURES 5

2.4.1. DESCRIPTION AND PRIORITY 5

2.5. NON FUNCTIONAL REQUIREMENTS 6

2.5.1. USE CASE DIAGRAM 6

2.5.3. SAFETY AND SECURITY REQUIREMENTS 7

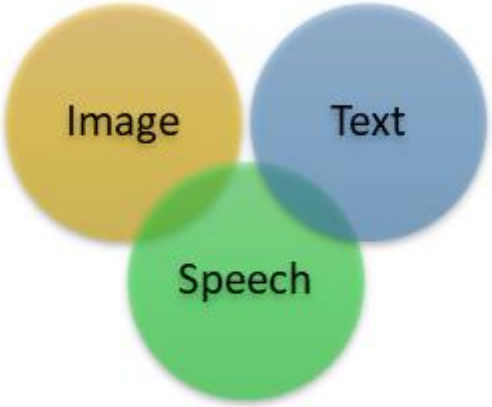
2.5.4. SOFTWARE QUALITY ATTRIBUTES 7

3. SYSTEM DESIGN 8
3.1. CLASS DIAGRAM 8
3.2. DATA FLOW DIAGRAM 8
i. LEVEL 0 DFD 8
ii. LEVEL 1 DFD 9
iii. LEVEL 2 DFD 10
4.IMPLEMENTATION AND RESULTS 12
4.1. Module 1: login page 12
Frontpage.py 12
Database.py 15
LoginLayout.kv 16
4.2. Module 2: Homepage and Navigation Drawer 19
MainNav.py 19
Main.kv 20
Homepage.py 24
Homepage.kv 25
4.3. Module 3: Image module 27
ImageEdit.py 27
ImageEdit.kv 30
4.4. Module 4: OCR page 35
ImageText.py 35
ImageText.kv 37
4.5. Module 5: Text module 38
TextToText.py 38
myTextToText.kv 40
Transkivy.py 42
myTrans.kv 44
4.6. Module 6:Speech 47
audio1.py 47
speech.kv 47
5.TESTING 51
6.CONCLUSION AND FUTURE ENHANCEMENT 54
7.BIBLIOGRAPHY 54

1 INTRODUCTION

We all have been using Android and mobile phones so much for the past few years that it has become an inevitable part of our lives. Our objective was to blend in the various aspects of text, image, language, speech and recognition to bring out the application which could be useful. This application aims at providing convenience for people who want to quickly edit images, text or dictate notes. It could be very useful to recognise letters in images.

This app built with Kivy and Python serves to help small daily applications of user and aid in the efficiency of their work. It has the features of image editing with colour filters and flipping and rotating. It also includes recognising text from images and listening to the converted text loud. It serves as a basic formatter like capitaliser and find and replacer. It analyses the word, character and sentence count and finds keywords. It translates between languages with Internet connectivity. It interconverts speech and text. It is available free with just registering once and creating an account.



.

1.2 LITERATURE SURVEY

There are now a countless number of images to text conversion apps, but mostly they cannot be used on a larger scale because they are not open-sourced. The text obtained are also mostly gibberish and not applicable to unclear pictures. The further processing of text is mostly not a part of the existing applications. They are very sophisticated or complicated for a simple application user. They don't interconnect the various media like image, speech, text, formatting and translation. Formatting in a phone is complicated and word processing applications like office in phone take quite some time for loading and it imitates desktop in performance. Most of us would like to capitalise chunks of text to gain immediate attention. We may also need to change quickly, like replace and know about the numbers like word or sentence count in a passage. It just started as an aid to the passage obtained from OCR of Image.

But since a lot of APIs are available open source, we can expect more such translation and OCR apps in future, without need to pay for it. Also, we have enhanced speech-text modules and text-speech conversion. With Material design and UI features enhanced, these apps could easily become very much aesthetic and user-friendly. These aesthetic and appealing apps are the only existing ones which the user prefers to use.

2 SOFTWARE REQUIREMENT SPECIFICATION

This chapter gives a SRS for the system. It describes all the requirements that are needed to develop this software.

2.1. INTRODUCTION:

2.1.1. PURPOSE:

This document serves to help the readers and users as to how this application can be accessed. The document shall help the reader to understand the intricacies of the application and the effort and base for materialising it. Readers need to go through this for basic understanding of the system

2.1.2. CONVENTION:

As the document shall deal with certain jargons and concepts which shall be duly highlighted in italics. Technical details pertaining to the hardware and software specifications shall be highlighted too.

2.1.3. INTENDED AUDIENCE:

This app can be accessed and is useful to all people. There is no specific domain for this app. Anyone who wants to dictate huge passages, or wants to read text out loud, or those who have to immediately send an Email where images could not be appropriate, can just convert them into text, edit it easily and send it. It is also an immediate tool for grey scaling, or changing color of pictures and saving them quickly.

2.1.4. PRODUCT SCOPE:

The scope of this project is to provide an efficient and enhanced software tool for the users to perform all kinds of Text Image Analysis. The major scopes of this project are

- Converting Images to Text
- Converting Text to Speech and vice versa
- To Find and Replace Words
- Finding Keywords and analysis of text
- Capitalisation of Specific Words
- Finding Synonyms and antonyms

2.2. OVERALL DESCRIPTION:

2.2.1. PRODUCT FUNCTIONS:

Functional requirements are given in the form of user stories. In software development and product management, a user story is an informal, natural language description of one or more features of a software system. User stories are often written from the perspective of an end user or user of a system. They are often recorded on index cards, on Post-it notes, or digitally in project management software. Depending on the project, user stories may be written by various stakeholders including clients, users, managers, or development team members.

1. Picture to text
2. Text to speech
3. Speech to text
4. Translator
5. Error correction with synonyms
6. Keywords finding
7. Summarization

2.2.2.USER CLASSES AND CHARACTERISTICS:

The primary users are students, office going people and all others who could make use of the functionalities.

2.2.3. OPERATING ENVIRONMENT:

Operating environment for the case management system is as listed below.

- Operating system: Windows.
- Database: Only file storage required for login authentication
- Platform: Kivy with python

2.2.4.DESIGN AND IMPLEMENTATION CONSTRAINTS:

- Images with handwritten scanned data or very unclear data or complex fonts need attention while recognition of text.
- Speech should be clear for recognition and audio produced from text should not be monotonous and give pauses at right places.
- Text formatting with capitalisation has to get input with proper spacing with respect to full stop for getting recognised as sentence and for sentence capitalisation

- Translation API needs Internet connection for functioning and certain Unicode symbols especially in Tamil and other native languages should be rendered properly.
- Cropping should raise proper exceptions if lengthwise and height wise cropped image size becomes negative.

2.3. EXTERNAL INTERFACE REQUIREMENTS:

The external interface required for this case management system are as listed below:

2.3.1. USER INTERFACES:

The user interfaces used for this system is

- Front-end software: Kivy and KivyMD (Material Design Standards as per Google)
- Back-end software: Python

2.3.2. HARDWARE INTERFACES:

The hardware interfaces required for this project are:

- Windows and Android for final deployment
- Cross Platform deployment possible in MacOS, iOS, Linux etc. also

2.3.3. SOFTWARE INTERFACES:

Following are the software used for the cases library maintenance:

Software used

1. Operating system - Windows OS for its best support and user-friendliness. But it supports all platforms for operation.
2. Python for basic coding with wrappers and modules like Tesseract for OCR, Pillow and OpenCV for image processing, SpeechRecognition and Text to Speech (TTS) library ie. pyttsx3 for speech module

2.4. SYSTEM FEATURES:

2.4.1. DESCRIPTION AND PRIORITY:

Recognising huge texts from images was difficult especially when it came to sending the text through mail or for converting them into presentations or documents. Changing

image colours and basic small functionalities like flip and rotate do not require complex heavyweight applications after this app was built. Doing replacement, capitalisation and analysis along with text recognition manually is very difficult and it also consumes a lot of time and manpower. To overcome this the application has been developed in such a way that it enhances functionality and produces optimized results in minimal time.

2.5.NON FUNCTIONAL REQUIREMENTS:

2.5.1. USE CASE DIAGRAM:

A use case diagram is a dynamic or behaviour diagram. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform.

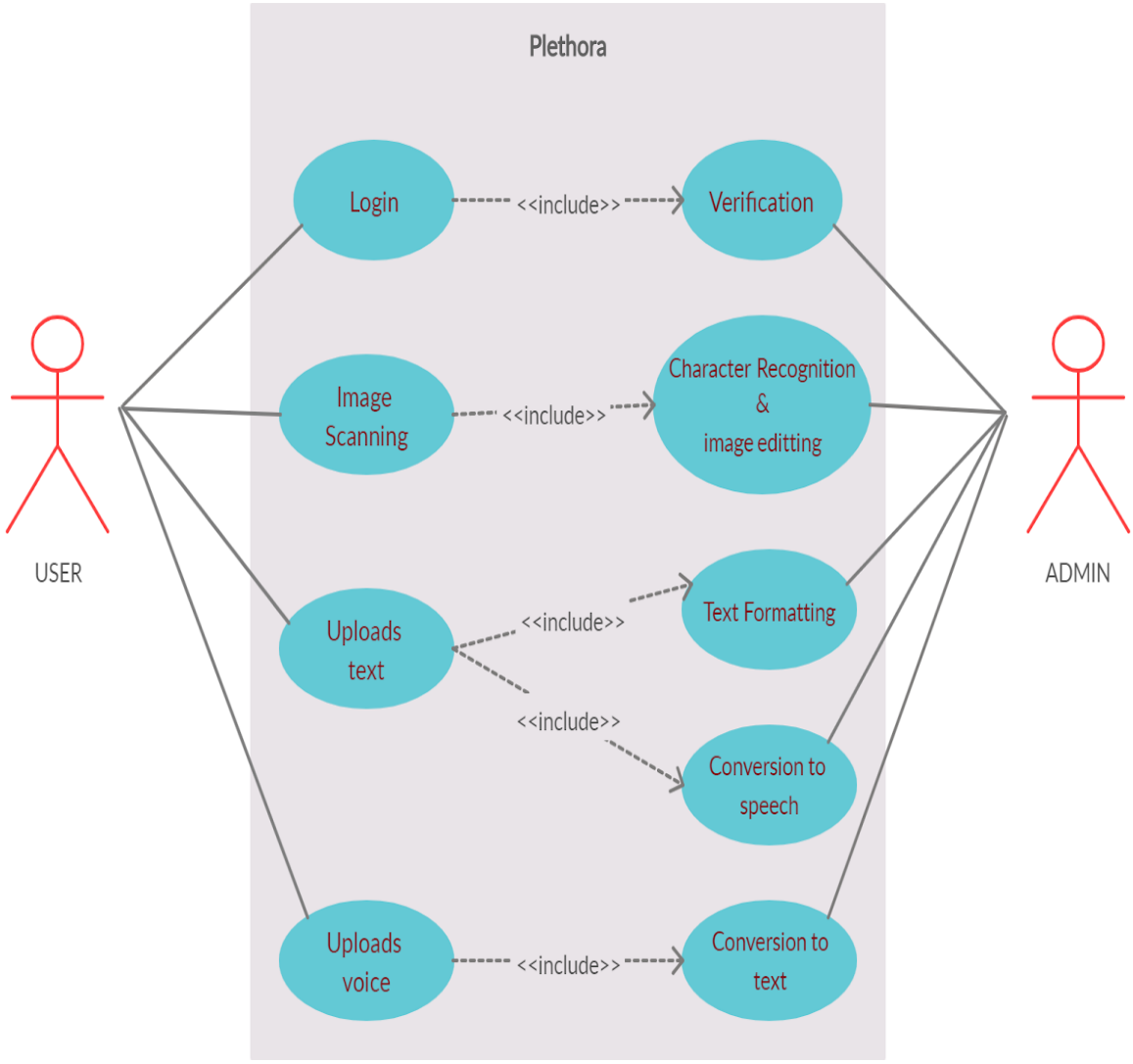


FIGURE 1

2.5.2. SAFETY AND SECURITY REQUIREMENTS:

1. It must prevent illegal access to camera, microphone and other hardware when app is not used
2. Images must be safe and inaccessible for other users.
3. It should be safe from hacking and illegal access of contents of the phone

2.5.3. SOFTWARE QUALITY ATTRIBUTES:

The system should have the following quality attributes, which are as stated below:

- AVAILABILITY: Once the user registers in the software it can be accessed 24/7. This availability is the key quality assured by this software.
- CORRECTNESS: The features provided by the software are accurate and reliable. The erroneous workings identified by the feedback obtained is corrected.
- MAINTAINABILITY: The app is maintained and regular update versions are provided.
- USABILITY: This app is flexible and can be accessed by everyone.

3 SYSTEM DESIGN

3.1.CLASS DIAGRAM:

A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attribute, operations or methods and the relationship among objects.

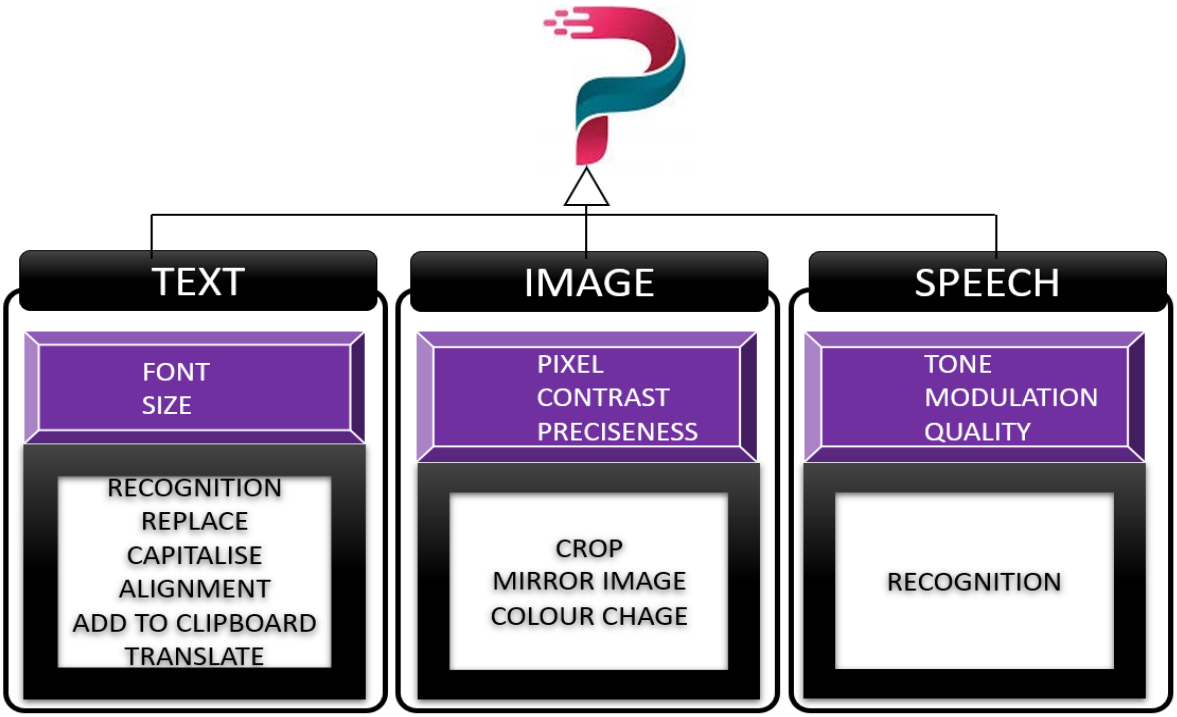


FIGURE 2

3.2. DESIGNING USING DFD:

A data-flow diagram (DFD) is a way of representing the flow of a data of a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

STEP 1: Draw the context diagram or level 0 DFD.

LEVEL 0 DFD:

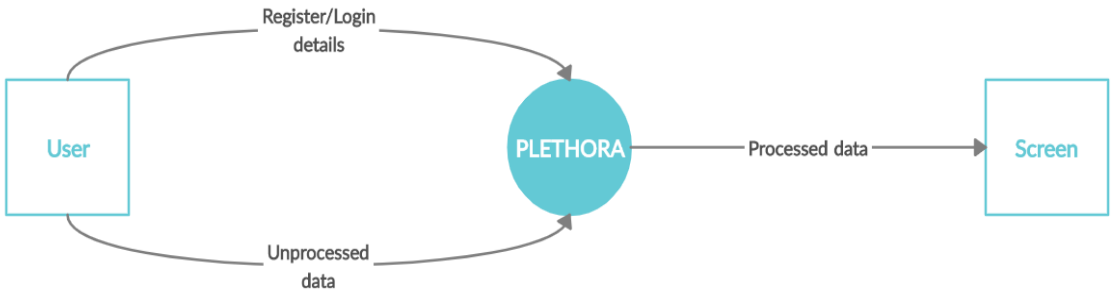


FIGURE 3a

STEP 2: Expand ‘PLETHORA’ process into the following modules,

1. Register
2. Login
3. Categorise

    - Image module
    - Text module
    - Speech module

LEVEL 1 DFD:

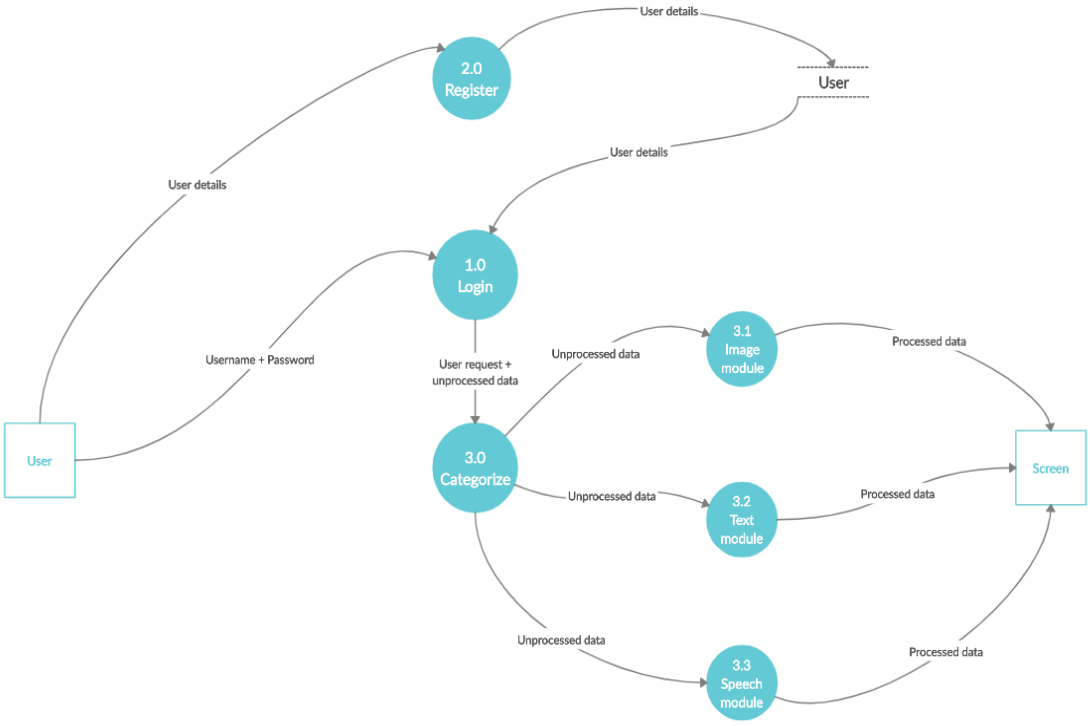


FIGURE 3b

STEP 3: Draw DFD level 2 where the process 3.1, Image module is divided into

- Colour change
- Greyscale
- Flip
- Crop
- Rotate clockwise
- Negative

LEVEL 2 DFD:

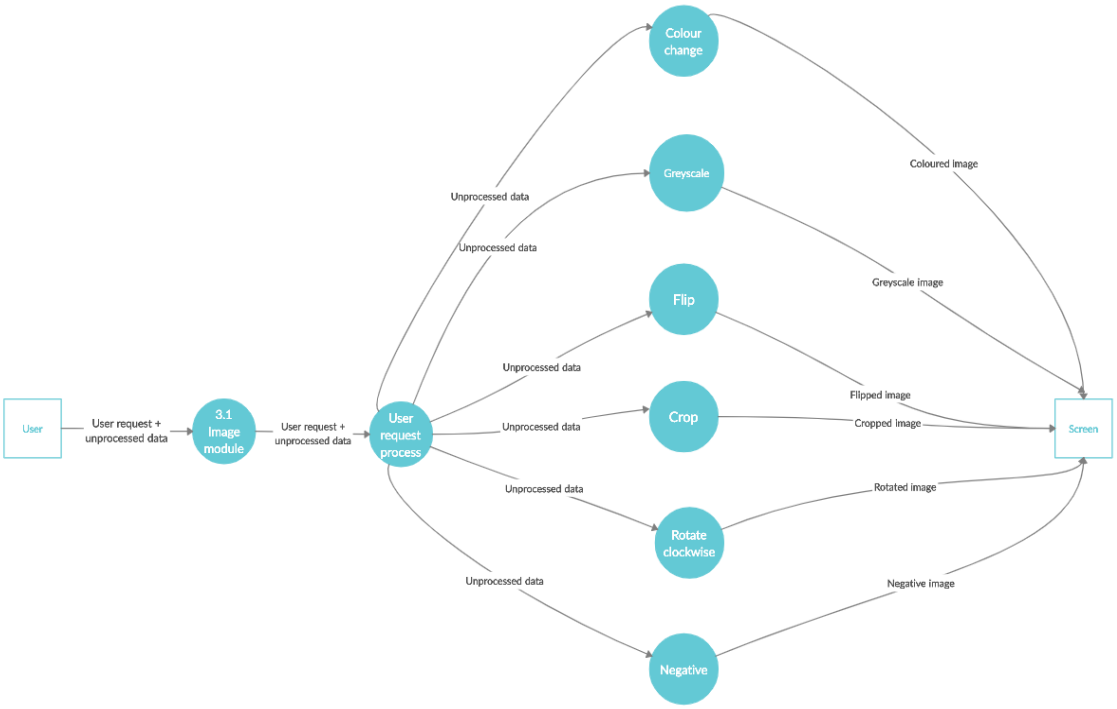


FIGURE 3c

MODULES DESCRIPTION:

A module is a software component or part of a program that contains one or more routines. One or more independently developed modules make up a program. Modules make a programmer's job easy by allowing the programmer to focus on only one area of the functionality of the software application.

Our system has been divided into modules like login, image, text recognition, text formatting, analysing and translation and the speech module. Each module plays a significant role in this project. The logic behind each module is explained below:

1. LOGIN MODULE:

The Login Module is a portal module that allows users to type a username and password to log in that particular system.

2. IMAGE PROCESSING MODULE:

The image processing module allows the user to browse for images and accepts the valid image format for further processing. The functions include :

- Color tints or filters
- Greyscale
- Vertical and horizontal flip
- Crop and Rotate
- Zoom and Brightness

3. OPTICAL CHARACTER RECOGNITION:

This module is used to recognise the various letters present in images. It is especially useful for large volumes of text. It also provides an option to read out the converted text. Fonts and colours are all recognised with utmost perfection when it is a snapshot of a printed text. Handwritten images work with minimum errors.

4. TEXT MODULE:

This is a supplementary module with functions like

- Capitalisation
- Find and Replace
- Analyse the number of words, sentences
- Translation

5. SPEECH MODULE:

This provides the basic functionality of text to speech and speech to text interconversions.

4 IMPLEMENTATION AND RESULTS

4.1 Module 1: login page

Frontpage.py

```
from kivymd.app import MDApp
from kivy.lang import Builder
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.properties import ObjectProperty
from kivy.uix.popup import Popup
from kivy.uix.label import Label
from kivymd.theming import ThemeManager
from database import DataBase
from kivy.config import Config
from kivy.core.window import Window

Config.set('graphics','resizable',0)

Builder.load_file('Main.kv')
import MainNav
Builder.load_file("mytrans.kv")
Builder.load_file("myTextToText.kv")
Builder.load_file("imageEdit.kv")
Builder.load_file("imageText.kv")
Builder.load_file("speech.kv")
Builder.load_file("Homepage.kv")
import TransKivy
import textToText
import ImageEdit
import ImageText
import audio1
import Homepage
db = DataBase("users.txt")


class WindowManager(ScreenManager):
    pass

class NavScreen(Screen):
    def logOut(self):
        self.manager.current='login'
    def on_pre_enter(self):
        Window.size = (800, 600)
    pass


class CreateAccountWindow(Screen):
```

```
    namee = ObjectProperty(None)
    email = ObjectProperty(None)
    password = ObjectProperty(None)

    def on_pre_enter(self):
        Window.size = (400, 400)
    pass

    def submit(self):
        if self.namee.text != "" and self.email.text != "" and self.ema
il.text.count("@") == 1 and self.email.text.count(".") > 0:
            if self.password.text != "":
                db.add_user(self.email.text.strip(), self.password.text
.strip(), self.namee.text.strip())
                self.reset()
                self.manager.current = "login"
            else:
                invalidForm()
        else:
            invalidForm()

    def login(self):
        self.reset()
        self.manager.current = "login"

    def reset(self):
        self.email.text = ""
        self.password.text = ""
        self.namee.text = ""


class LoginWindow(Screen):
    email = ObjectProperty(None)
    password = ObjectProperty(None)

    def on_pre_enter(self):
        Window.size = (400, 400)
    pass

    def loginButton(self):
        if db.validate(self.email.text.strip(), self.password.text.stri
p()):
            #MainWindow.current = self.email.text
            self.reset()
            self.manager.current = "main"
        else:
```

```
            invalidLogin()

    def createButton(self):
        self.reset()
        self.manager.current = "create"

    def reset(self):
        self.email.text = ""
        self.password.text = ""




def invalidLogin():
    pop = Popup(title='Invalid Login',
                  content=Label(text='Invalid username or password.'),
                  size_hint=(None, None), size=(250, 100))
    pop.open()


def invalidForm():
    pop = Popup(title='Invalid Form',
                  content=Label(text='Please fill in all inputs with va
lid information.'),
                  size_hint=(None, None), size=(350, 100))

    pop.open()



class LoginApp(MDApp):

    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.theme_cls.primary_palette = "Pink"
        self.title="Plethora"

    def build(self):

        kv = Builder.load_file("Loginlayout.kv")
        return kv


if __name__ == "__main__":
    LoginApp().run()
```

Database.py

```
import datetime


class DataBase:
    def __init__(self, filename):
        self.filename = filename
        self.users = None
        self.file = None
        self.load()

    def load(self):
        self.file = open(self.filename, "r")
        self.users = {}

        for line in self.file:
            email, password, name, created = line.strip().split(";")
            self.users[email] = (password, name, created)

        self.file.close()

    def get_user(self, email):
        if email in self.users:
            return self.users[email]
        else:
            return -1

    def add_user(self, email, password, name):
        if email.strip() not in self.users:
            self.users[email.strip()] = (password.strip(), name.strip()
, DataBase.get_date())
            self.save()
            return 1
        else:
            print("Email exists already")
            return -1

    def validate(self, email, password):
        if self.get_user(email) != -1:
            return self.users[email][0] == password
        else:
            return False

    def save(self):
        with open(self.filename, "w") as f:
            for user in self.users:
```

```
                f.write(user + ";" + self.users[user][0] + ";" + self.u
sers[user][1] + ";" + self.users[user][2] + "\n")

    @staticmethod
    def get_date():
        return str(datetime.datetime.now()).split(" ")[0]
```

LoginLayout.kv

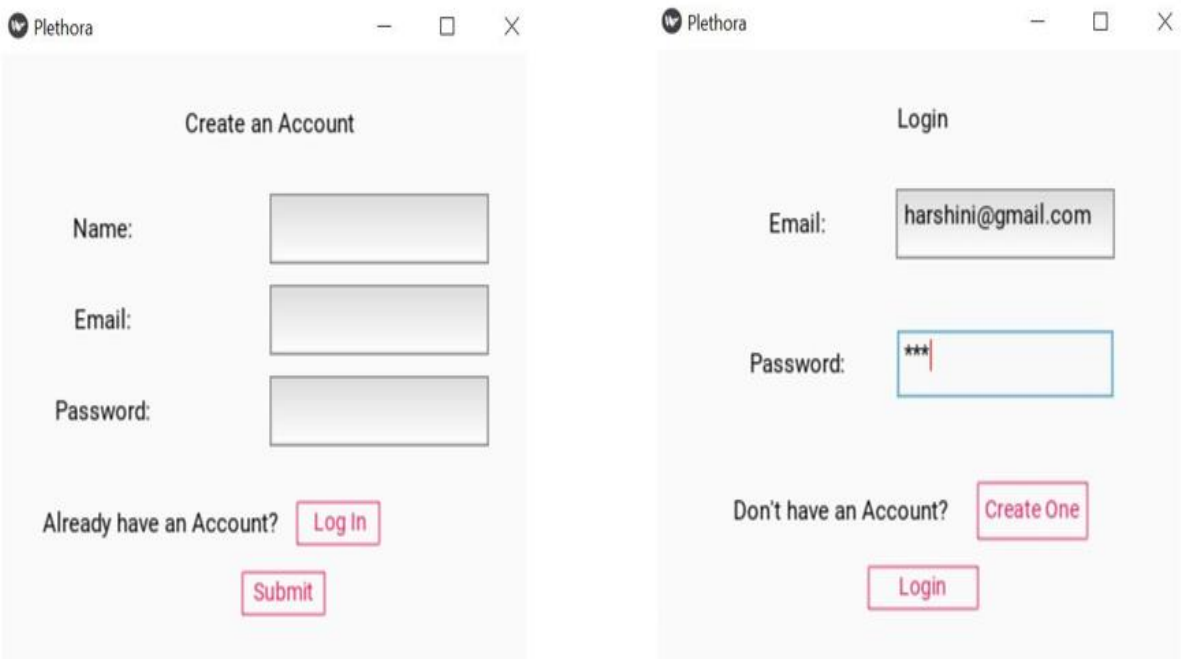
```
WindowManager:
    id:sm
    CreateAccountWindow:
    LoginWindow:
    NavScreen:

<Button>
    background_normal:''
    background_color:(1,1,1,1)
    color:app.theme_cls.primary_color
    background_down:'pink.png'
<CreateAccountWindow>:
    name: "create"

    namee: namee
    email: email
    password: passw

    FloatLayout:
        cols:1

        FloatLayout:
            size: root.width, root.height/2

            Label:
                text: "Create an Account"
                color:0,0,0,1
                size_hint: 0.8, 0.2
                pos_hint: {"x":0.1, "top":1}


            Label:
                size_hint: 0.4,0.1
                color:0,0,0,1
                pos_hint: {"x":0, "top":0.8}
                text: "Name: "
```

```
    TextInput:
        pos_hint: {"x":0.5, "top":0.8}
        size_hint: 0.4, 0.1
        id: namee
        multiline: False



    Label:
        size_hint: 0.4,0.1
        color:0,0,0,1
        pos_hint: {"x":0, "top":0.67}
        text: "Email: "


    TextInput:
        pos_hint: {"x":0.5, "top":0.67}
        size_hint: 0.4, 0.1
        id: email
        multiline: False


    Label:
        size_hint: 0.4,0.1
        color:0,0,0,1
        pos_hint: {"x":0, "top":0.54}
        text: "Password: "


    TextInput:
        pos_hint: {"x":0.5, "top":0.54}
        size_hint: 0.4, 0.1
        id: passw
        multiline: False
        password: True
Label:
    text: "Already have an Account?"
    pos_hint:{"x":0.15,"y":0.3}
    color:0,0,0,1
    size_hint:0.3,0.06

MDRectangleFlatButton:
    pos_hint:{"x":0.55,"y":0.3}
    size_hint: 0.15, 0.06
```

```
            text:" Log In"
            on_release:
                root.manager.transition.direction = "left"
                root.login()

        MDRectangleFlatButton:
            pos_hint:{"x":0.45,"y":0.2}
            size_hint: 0.15, 0.06
            text: "Submit"
            on_release:
                root.manager.transition.direction = "left"
                root.submit()


<LoginWindow>:
    name: "login"

    email: email
    password: password

    FloatLayout:

        Label:
            text:"Login"
            color:0,0,0,1
            size_hint:0.8,0.2
            pos_hint:{"x":0.1,"top":1}

        Label:
            text:"Email: "
            color:0,0,0,1
            pos_hint: {"x":0.1, "top":0.8}
            size_hint: 0.35, 0.1

        TextInput:
            id: email

            multiline: False
            pos_hint: {"x": 0.45 , "top":0.8}
            size_hint: 0.4, 0.1

        Label:
            text:"Password: "
            color:0,0,0,1
            pos_hint: {"x":0.1, "top":0.6}
            size_hint: 0.35, 0.1
```

```
        TextInput:
            id: password
            multiline: False
            password: True
            pos_hint: {"x": 0.45, "top":0.6}
            size_hint: 0.4, 0.1

        MDRectangleFlatButton:
            pos_hint:{"x":0.4,"y":0.2}
            size_hint: 0.2, 0.06

            text: "Login"
            on_release:
                root.manager.transition.direction = "up"
                root.loginButton()

        MDRectangleFlatButton:
            pos_hint:{"x":0.6,"y":0.3}
            size_hint: 0.2, 0.08

            text: "Create One"
            on_release:
                root.manager.transition.direction = "right"
                root.createButton()

        Label:
            pos_hint:{"x":0.1,"y":0.3}
            size_hint:0.5,0.08
            color:0,0,0,1
            text:"Don't have an Account?"

<NavScreen>
    name:'main'
    NavigationDrawer:
```

Screenshots:



4.2 Module 2: Homepage and Navigation Drawer

MainNav.py

```
from kivymd.app import MDApp
from kivymd.uix.navigationdrawer import NavigationLayout
from kivy.uix.boxlayout import BoxLayout
from kivy.lang import Builder
from kivy.uix.screenmanager import ScreenManager, Screen


class NavigationDrawer(NavigationLayout):
    def back_to_home_screen(self):
        self.ids.navsm.current='home'

class ContentNavigationDrawer(BoxLayout):
    pass
```

Main.kv

```
#: import SlideTransition kivy.uix.screenmanager.SlideTransition
<NavigationDrawer>:

    ScreenManager:
        id:navsm
        transition:SlideTransition(direction='left')
```

```
        Screen:
            name:'home'
            BoxLayout:
                orientation: 'vertical'
                MDToolbar:
                    title: "Plethora"
                    md_bg_color: app.theme_cls.primary_color
                    elevation: 10
                    left_action_items: [['menu', lambda x: nav_drawer.s
et_state("toggle")]]
                    right_action_items: [['home-
outline', lambda x: root.back_to_home_screen()]]
                HomePage:
        Screen:
            name:'img'
            BoxLayout:
                orientation: 'vertical'
                MDToolbar:
                    title: " Edit Image"
                    md_bg_color: app.theme_cls.primary_color
                    elevation: 10
                    left_action_items: [['menu', lambda x: nav_drawer.s
et_state("toggle")]]
                    right_action_items: [['home-
outline', lambda x: root.back_to_home_screen()]]
                ImageLayout1
        Screen:
            name:'ocr'
            BoxLayout:
                orientation: 'vertical'
                MDToolbar:
                    title: "Text Recognise"
                    md_bg_color: app.theme_cls.primary_color
                    elevation: 10
                    left_action_items: [['menu', lambda x: nav_drawer.s
et_state("toggle")]]
                    right_action_items: [['home-
outline', lambda x: root.back_to_home_screen()]]
                ImageToText:

        Screen:
            name:'translate'
            BoxLayout:
                orientation: 'vertical'
                MDToolbar:
                    title: "Translate"
```

```
                    md_bg_color: app.theme_cls.primary_color
                    elevation: 10
                    left_action_items: [['menu', lambda x: nav_drawer.s
et_state("toggle")]]
                    right_action_items: [['home-
outline', lambda x: root.back_to_home_screen()]]
                TransLayout:


        Screen:
            name:'Textformat'
            BoxLayout:
                orientation: 'vertical'
                MDToolbar:
                    title: "Plethora"
                    md_bg_color: app.theme_cls.primary_color
                    elevation: 10
                    left_action_items: [['menu', lambda x: nav_drawer.s
et_state("toggle")]]
                    right_action_items: [['home-
outline', lambda x: root.back_to_home_screen()]]
                TEXTFormatLayout:

        Screen:
            name:'speech'
            BoxLayout:
                orientation: 'vertical'
                MDToolbar:
                    title: "Speech"
                    md_bg_color: app.theme_cls.primary_color
                    elevation: 10
                    left_action_items: [['menu', lambda x: nav_drawer.s
et_state("toggle")]]
                    right_action_items: [['home-
outline', lambda x: root.back_to_home_screen()]]
                SpeechLayout:


    MDNavigationDrawer:
        id: nav_drawer
        ContentNavigationDrawer:
            id: content_drawer
            navsm:navsm


<ContentNavigationDrawer>
```

```
BoxLayout:
    orientation: 'vertical'
    FloatLayout:
        size_hint_y: None
        height: "200dp"
        canvas:
            Color:
                rgba: app.theme_cls.primary_color
            Rectangle:
                pos: self.pos
                size: self.size
        BoxLayout:
            id: top_box
            size_hint_y: None
            height: "200dp"
            x: root.parent.x
            pos_hint: {"top": 1}
            FitImage:
                source: "logo.jfif"
        MDIconButton:
            icon: "close"
            x: root.parent.x + dp(10)
            pos_hint: {"top": 1}
            on_press: root.parent.set_state()



    ScrollView:
        pos_hint: {"top": 1}
        MDGridLayout:
            id: box_item
            cols: 1
            adaptive_height: True
            TwoLineListItem:
                text: "Plethora "
                secondary_text: "Version 1.0"
            OneLineIconListItem:
                text:"HOME"
                on_press:
                    root.parent.set_state("close")
                    root.navsm.current='home'
            OneLineIconListItem:
                text:"Image Edit"
                on_press:
```

```
                root.parent.set_state("close")
                root.navsm.current='img'

        OneLineIconListItem:
            text:"Recognise Text"
            on_press:
                root.parent.set_state("close")
                root.navsm.current='ocr'
        OneLineIconListItem:
            text:"Translate"
            on_press:
                root.parent.set_state("close")
                root.navsm.current='translate'

        OneLineIconListItem:
            text:"Text Format and Analyse"
            on_press:
                root.parent.set_state("close")
                root.navsm.current='Textformat'

        OneLineIconListItem:
            text:"Speech Tool"
            on_press:
                root.parent.set_state("close")
                root.navsm.current='speech'
        OneLineIconListItem:
            text:"Feedback"
            on_press:
                root.parent.set_state("close")
                root.navsm.current='home'

        OneLineIconListItem:
            text:"Sign out"
            on_press:
                root.parent.set_state("close")
                root.parent.parent.parent.logOut()
```

Homepage.py

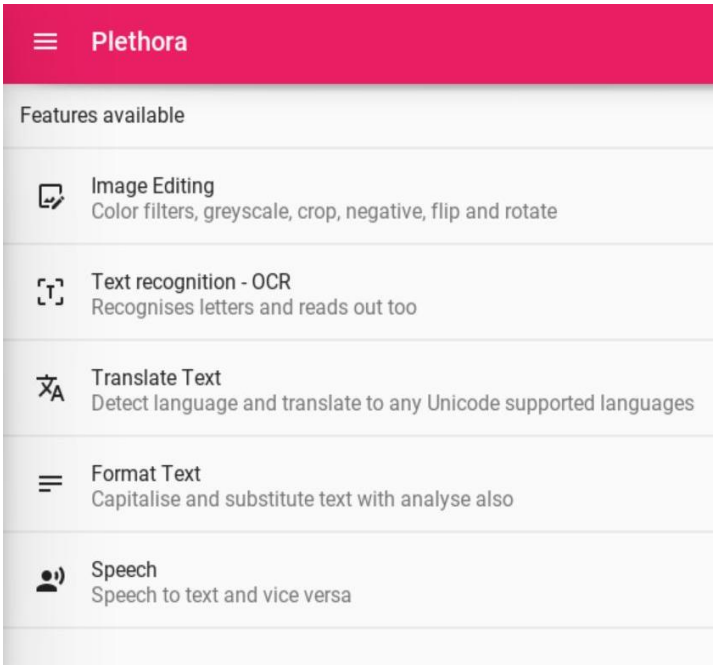
```
from kivy.uix.widget import Widget

class HomePage(Widget):
    pass
```

Homepage.kv

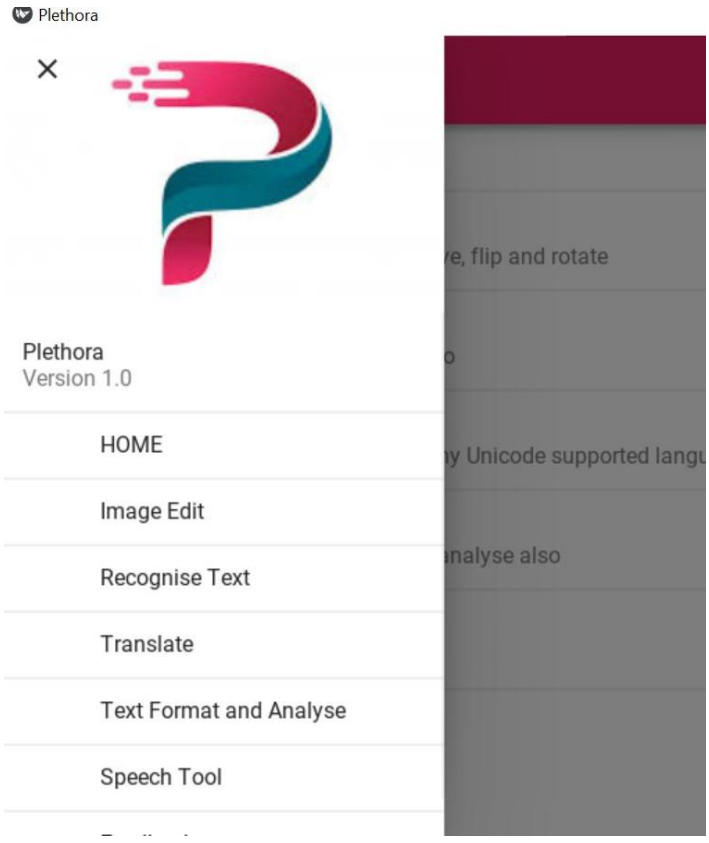
```
<HomePage>


    MDGridLayout:
        cols:1
        adaptive_size:True
        pos_hint:{"x":0.3,"top":0.85 }
        size:root.width,root.height
        OneLineListItem:
            text:"Features available"
            color:app.theme_cls.primary_color


        TwoLineIconListItem:
            text: "Image Editing"
            secondary_text: "Color filters, greyscale, crop, negative,
flip and rotate"

            IconLeftWidget:
                icon: "image-edit-outline"
        TwoLineIconListItem:
            text: "Text recognition - OCR"
            secondary_text: "Recognises letters and reads out too"

            IconLeftWidget:
                icon: "text-recognition"
        TwoLineIconListItem:
            text: "Translate Text"
            secondary_text: "Detect language and translate to any Unico
de supported languages"

            IconLeftWidget:
                icon: "translate"
        TwoLineIconListItem:
            text: "Format Text"
            secondary_text: "Capitalise and substitute text with analys
e also"

            IconLeftWidget:
                icon: "text"
        TwoLineIconListItem:
            text: "Speech "
            secondary_text: "Speech to text and vice versa"
```

```
IconLeftWidget:
    icon: "voice"
```

Screenshots



Homepage



Navigation drawer

4.3 Module 3: Image Module

ImageEdit.py

```
from kivymd.app import MDApp
from kivy.uix.widget import Widget
import os
from kivy.uix.label import Label
from kivy.uix.popup import Popup
import tkinter

from kivy.properties import ObjectProperty,StringProperty
from PIL import Image,ImageOps
from kivy.uix.popup import Popup
from kivy.uix.button import Button
from kivymd.toast import toast
from kivy.uix.gridlayout import GridLayout



class ImageLayout1(Widget):
    im1=ObjectProperty(None)
    im2=ObjectProperty(None)
    global filename
    filename="1.jpg"
    def SaveFile(self):
        toast('Saved successfully in gallery as out.jpg')




    def tint(self):
        try:

            if os.path.exists("out.jpg"):
                os.remove("out.jpg")
            img=Image.open(self.filename)
            imgG=img.convert("L")
            img1=ImageOps.colorize(imgG,black=self.ids.spin.text,white=
"white")
            img2=img1.save('out.jpg')

        except:
            pop=Popup(title='Some error occurred',content=Label(text='T
ry again'),size_hint=(None,None),size=(350,100))
            pop.open()
```

```
    def grey(self):
        try:

            if os.path.exists("out.jpg"):
                os.remove("out.jpg")
            img=Image.open(self.filename)
            imgG=img.convert("L")
            img2=imgG.save('out.jpg')

        except:
            pop=Popup(title='Some error occurred',content=Label(text='T
ry again'),size_hint=(None,None),size=(350,100))
            pop.open()


    def flip(self,dir):
        try:

            if os.path.exists("out.jpg"):
                os.remove("out.jpg")
            img=Image.open(self.filename)
            img=img.convert('RGB')
            if dir=='v' :
                img1=ImageOps.flip(img)
            elif dir=='h':
                img1=ImageOps.mirror(img)
            elif dir=='neg':
                img1=ImageOps.invert(img)
            img2=img1.save('out.jpg')



        except:
            pop=Popup(title='Some error occurred',content=Label(text='T
ry again'),size_hint=(None,None),size=(350,100))
            pop.open()


    def Convert(self):
        try:

            self.im2.source='old.jpg'
            for i in range(0,5):
```

```
                pass
            self.im2.source='out.jpg'
        except:
            pop=Popup(title='Some error occurred',content=Label(text='T
ry again'),size_hint=(None,None),size=(350,100))
            pop.open()

    def workOn(self):
        try:

            image=Image.open('out.jpg')
            image.save('workagain.jpg')
            self.filename='workagain.jpg'
            self.im1.source='old.jpg'
            print('test')
            self.im1.source=self.filename
        except:
            pop=Popup(title='Some error occurred',content=Label(text='T
ry again'),size_hint=(None,None),size=(350,100))
            pop.open()


    def browse(self):
        try:

            global filename
            from tkinter import ttk
            from tkinter import filedialog
            self.filename= filedialog.askopenfilename(initialdir="/",ti
tle="Select A file",filetype=(("jpeg files","*.jpg"),("all files","*.*"
)))
            self.im1.source=self.filename
        except:
            pop=Popup(title='Some error occurred',content=Label(text='T
ry again'),size_hint=(None,None),size=(350,100))
            pop.open()


    def resetSlider(self):
        self.ids.slid1.value=1

    def rot90(self):
        try:

            if os.path.exists("out.jpg"):
                os.remove("out.jpg")
```

```
            img=Image.open(self.filename)
            imgG=img.rotate(90)
            for i in range(0,5):
                pass
            img2=imgG.save('out.jpg')

        except:
            pop=Popup(title='Some error occurred',content=Label(text='T
ry again'),size_hint=(None,None),size=(350,100))
            pop.open()


    def crop(self):
        try:

            if os.path.exists("out.jpg"):
                os.remove("out.jpg")
            img=Image.open(self.filename)
            SZ=img.size
            l=SZ[0]*self.ids.left.value
            r=SZ[0]*self.ids.rgt.value
            t=SZ[1]*self.ids.top.value
            b=SZ[1]*self.ids.btm.value
            box=(l,t,r,b)
            if l>r or t>b:
                pop=Popup(title='Some error occurred',content=Label(tex
t='Try again'),size_hint=(None,None),size=(350,100))
                pop.open()

            else:
                img1=img.crop(box)
                img1.save('out.jpg')
        except:
            pop=Popup(title='Some error occurred',content=Label(text='T
ry again'),size_hint=(None,None),size=(350,100))
            pop.open()
```

ImageEdit.kv

```
<Label>
    color:0,0,0,1
```

```
<ImageLayout1>
    im1:QuesImag
    im2:Editted
    FloatLayout:
        id:fl
        size:root.width,root.height
        Image:
            id:QuesImag
            source:'old.jpg'
            nocache:True
            size_hint:0.4,0.4
            pos_hint:{"x":0.1,"top":0.95}

        Image:
            id:Editted
            source:'new.jpg'
            nocache:True
            opacity:slid2.value
            #size_hint:0.8,0.4
            size_hint:0.4*slid1.value,0.4*slid1.value
            pos_hint:{"center_x":0.3,"center_y":0.3}
            border:[30,30,30,30]
        Label:
            text:"Zoom"
            size_hint:0.05,0.05
            pos_hint:{"x":0.6,"top":0.55}
        MDSlider:
            id:slid1
            min:0.3
            max:1.5
            value:1
            hint:False
            pos_hint:{"x":0.64,"top":0.55}
            size_hint:0.3,0.05
        MDRectangleFlatButton:
            text:"Reset"
            pos_hint:{"x":0.93,"top":0.55}
            size_hint:0.05,0.05
            on_press:
                root.resetSlider()

        Label:
            text:"Brightness"
            size_hint:0.05,0.05
            pos_hint:{"x":0.58,"top":0.49}
```

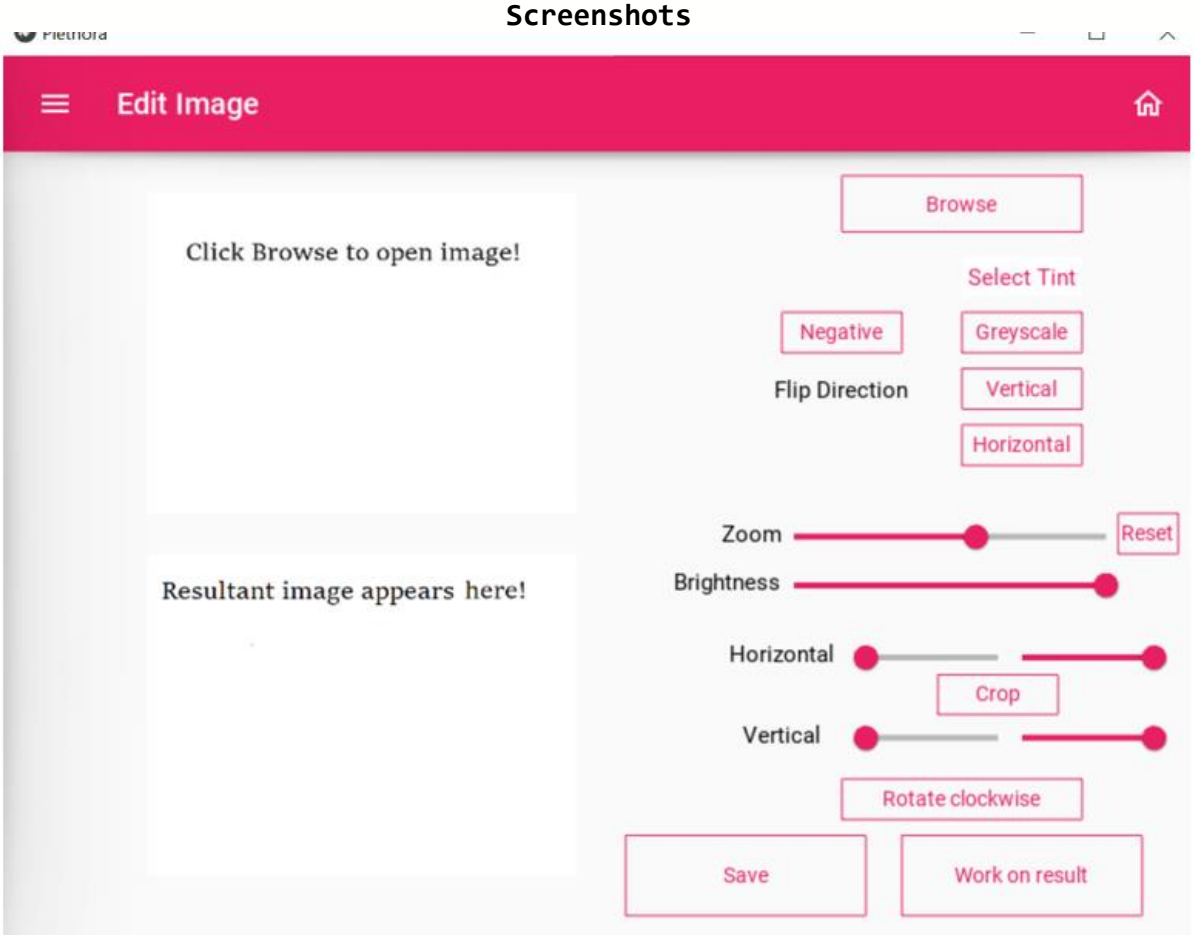
```
MDSlider:
    id:slid2
    min:0.3
    max:1
    value:1
    pos_hint:{"x":0.64,"top":0.49}
    size_hint:0.3,0.05
    hint:False



MDRectangleFlatButton:
    text:"Browse"
    pos_hint:{"x":0.7,"top":0.97}
    size_hint:0.2,0.07
    on_release:
        root.browse()


Spinner:
    id:spin
    text:"Select Tint"
    values:['Red','Blue','Green','Purple','Pink','Orange']
    pos_hint:{"x":0.8,"top":0.87}
    size_hint:0.1,0.05
    on_text:
        root.tint()
        root.Convert()


MDRectangleFlatButton:
    text:"Greyscale"
    pos_hint:{"x":0.8,"top":0.8}
    size_hint:0.1,0.05
    on_press:
        root.grey()
        root.Convert()
Label:
    text:"Flip Direction"
    pos_hint:{"x":0.65,"top":0.73}
    size_hint:0.1,0.05


MDRectangleFlatButton:
    text:"Vertical"
```

```
        pos_hint:{"x":0.8,"top":0.73}
        size_hint:0.1,0.05
        on_press:
            root.flip("v")
            root.Convert()

    MDRectangleFlatButton:
        text:"Horizontal"
        pos_hint:{"x":0.8,"top":0.66}
        size_hint:0.1,0.05
        on_press:
            root.flip("h")
            root.Convert()

    MDRectangleFlatButton:
        text:"Negative"
        pos_hint:{"x":0.65,"top":0.8}
        size_hint:0.1,0.05
        on_press:
            root.flip("neg")
            root.Convert()

    MDRectangleFlatButton:
        text:"Rotate clockwise"
        pos_hint:{"x":0.7,"top":0.22}
        size_hint:0.2,0.05
        on_press:
            root.rot90()
            root.Convert()

    MDRectangleFlatButton:
        text:"Crop"
        pos_hint:{"x":0.78,"top":0.35}
        size_hint:0.1,0.05
        on_press:
            root.crop()
            root.Convert()


    MDRectangleFlatButton:
        text:"Work on result"
        pos_hint:{"x":0.75,"top":0.15}
        size_hint:0.2,0.1
        on_press:
            root.workOn()
```

```
MDRectangleFlatButton:
    text:"Save"
    pos_hint:{"x":0.52,"top":0.15}
    size_hint:0.2,0.1
    on_press:
        root.SaveFile()


MDSlider:
    id:left
    min:0
    max:1
    value:0
    pos_hint:{"x":0.7,"top":0.4}
    size_hint:0.15,0.05
    hint:False

MDSlider:
    id:top
    min:0
    max:1
    value:0
    pos_hint:{"x":0.7,"top":0.3}
    size_hint:0.15,0.05
    hint:False

Label:
    text:"Horizontal"
    pos_hint:{"x":0.6,"top":0.4}
    size_hint:0.1,0.05

Label:
    text:"Vertical"
    pos_hint:{"x":0.6,"top":0.3}
    size_hint:0.1,0.05


MDSlider:
    id:rgt
    min:0
    max:1
    value:1
    pos_hint:{"x":0.83,"top":0.4}
    size_hint:0.15,0.05
    hint:False
```

```
MDSlider:
    id:btm
    min:0
    max:1
    value:1
    pos_hint:{"x":0.83,"top":0.3}
    size_hint:0.15,0.05
    hint:False
```

Screenshots



4.4 Module 4: OCR page

ImageText.py

```
from kivymd.app import MDApp
import pytesseract
from pytesseract import image_to_string
from tkinter import *
from tkinter import ttk
from tkinter import filedialog
```

```
import pyttsx3
from kivy.uix.popup import Popup
from kivy.uix.widget import Widget
from kivy.uix.label import Label

class ImageToText(Widget):
    global text
    text=" "
    global filaname
    filename=" "
    def TextConversion(self):
        from PIL import Image
        global text
        pytesseract.pytesseract.tesseract_cmd=r'C:\Program Files\Tesser
act-OCR\tesseract.exe'
        try:
            image = Image.open(filename)
            text = pytesseract.image_to_string(image)
            text=text.replace('\n',' ')
            self.textLbl.text=text
        except:
            pop=Popup(title='Some error occurred',content=Label(text='T
ry again'),size_hint=(None,None),size=(350,100))
            pop.open()

    def SpeechConversion(self):
        if text==" ":
            pop=Popup(title='Some error occurred',content=Label(text='T
ry again',color=(1,1,1,1)),size_hint=(None,None),size=(350,100))
            pop.open()
        else:
            engine = pyttsx3.init()
            engine.say(text)
            engine.setProperty('rate',120)
            engine.setProperty('volume', 0.9)
            engine.runAndWait()
    def BrowseImage(self):
        global filename
        filename= filedialog.askopenfilename(initialdir="/",title="Sele
ct A file",filetype=(("jpeg files","*.jpg"),("all files","*.*")))
        #self.label.configure(text=self.filename)
        self.label_wid1.source=filename
```

ImageText.kv

```
<MDRectangleFlatIconButton>
    custom_color:app.theme_cls.primary_color

<ImageToText>:
    textLbl:textLbl
    label_wid1: my_image1
    FloatLayout:
        size:root.width,root.height
        Image:
            id: my_image1
            source:'old.jpg'
            size_hint:0.45,0.45
            pos_hint:{"x":0.1,"top":0.98}
        MDRectangleFlatIconButton:
            icon:'image'
            text:'Browse'
            pos_hint:{"x":0.7,"top":0.9}
            size_hint:0.15,0.05
            on_press: root.BrowseImage()
        MDRectangleFlatIconButton:
            icon:'text-recognition'
            text:'Convert'
            on_press:root.TextConversion()
            pos_hint:{"x":0.7,"top":0.8}
            size_hint:0.15,0.05
        TextInput:
            id:textLbl
            text:"here comes the result"
            size_hint:0.45,0.45
            pos_hint:{"x":0.1,"top":0.48}
        MDRectangleFlatIconButton:
            icon:'text-to-speech'
            text:'Listen'
            on_release: root.SpeechConversion()
            pos_hint:{"x":0.7,"top":0.7}
            size_hint:0.15,0.05
```

Screenshots

4.5 Module 5: Text module

TextToText.py

```
from kivymd.app import MDApp
from kivy.uix.widget import Widget
from kivy.uix.button import Button
from kivy.uix.dropdown import DropDown
from kivymd.theming import ThemeManager
from kivy.uix.popup import Popup
from kivy.uix.label import Label
import re
from kivy.uix.floatlayout import FloatLayout
import pyperclip

class TEXTFormatLayout(Widget):
    def analyser(self):
        s=self.ids.ques.text
        wrdcnt=len(s.split(' '))
        charcnt=len(s)
        charWithoutSpace=len(s.replace(' ',''))
        senct=len(s.split('.'))-1
        res='Char count ='+str(charcnt)+'\nChar count without space ='+
str(charWithoutSpace)+'\nWord Count ='+str(wrdcnt)+'\nSentence count ='
+str(senct)
        self.ids.res.text=res
```

```
def copyClip(self):
    #Clipboard.copy(self.ids.res.tcext)
    pyperclip.copy(self.ids.res.text)
def CpToQ(self):
    self.ids.ques.text=self.ids.res.text
def upperCap(self):
    self.ids.res.text=self.ids.ques.text.upper()
def lowerCap(self):
    self.ids.res.text=self.ids.ques.text.lower()
def SenCap(self):
    s=self.ids.ques.text
    s='. '.join(x.capitalize() for x in s.split('. '))
    self.ids.res.text=s
def InitCap(self):
    s=self.ids.ques.text
    self.ids.res.text=s.title()
def replacer(self):
    try:

        s=self.ids.ques.text
        res=''
        find=self.ids.finder.text.strip()
        rep=self.ids.replace.text.strip()
        if self.ids.whole.active:
            if self.ids.case.active:
                #print('both wholewords and ignore case')
                x=s.split( )
                for i in range(len(x)):
                    if x[i].lower()==find.lower():
                        x[i]=rep
                res=' '.join(x)


            else:
                #print('just whole words no ignore case')
                x=s.split( )
                for i in range(len(x)):
                    if x[i]==find:
                        x[i]=rep
                    #print(x[i])
                res=' '.join(x)
        else:
            if self.ids.case.active:
               # print('just ignore case')
                res=re.sub(find,rep,s,flags=re.IGNORECASE)
            else:
```

```
                    # print('just normal')
                     res=s.replace(find,rep)
            self.ids.res.text=res
        except:
            pop=Popup(title='Some error occurred',content=Label(text='T
ry again',color=(1,1,1,1)),size_hint=(None,None),size=(350,100))
            pop.open()
```

myTextToText.kv

```
<MDRectangleFlatButton>
    size_hint:0.11,0.06



<TEXTFormatLayout>
    FloatLayout:
        size:root.width,root.height
        TextInput:
            id:ques
            size_hint:0.7,0.3
            pos_hint:{"x":0.05,"top":0.97}
            text: "Type the text to be formatted"
        TextInput:
            id:res
            size_hint:0.7,0.3
            pos_hint:{"x":0.05,"top":0.35}
            text:"Here comes the result"
        MDRectangleFlatButton:
            text:"Copy to clipboard"
            color:1,1,1,1
            size_hint:0.16,0.06
            pos_hint:{"x":0.1,"top":0.65}
            on_press:root.copyClip()
        MDRectangleFlatButton:
            text:"Work on result"
            color:1,1,1,1
            size_hint:0.15,0.06
            on_press:root.CpToQ()
            pos_hint:{"x":0.35,"top":0.65}

        Label:
            text:"Capitalise Options"
            pos_hint:{"x":0.1,"top":0.58}
```

```
        size_hint:0.1,0.06
        color:[0,0,0,1]

    MDRectangleFlatButton:
        pos_hint:{"x":0.25,"top":0.58}
        text:"Upper Case"
        size_hint:0.10,0.06
        color:1,1,1,1
        on_press:root.upperCap()
    MDRectangleFlatButton:
        pos_hint:{"x":0.37,"top":0.58}
        text:"Lower Case"
        size_hint:0.10,0.06
        color:1,1,1,1
        on_press:root.lowerCap()
    MDRectangleFlatButton:
        pos_hint:{"x":0.48,"top":0.58}
        text:"Sentence Case"
        color:1,1,1,1
        size_hint:0.14,0.06
        on_press:root.SenCap()
    MDRectangleFlatButton:
        pos_hint:{"x":0.63,"top":0.58}
        text:"InitCap"
        size_hint:0.11,0.06
        color:1,1,1,1
        on_press:root.InitCap()
    MDRectangleFlatButton:
        pos_hint:{"x":0.1,"top":0.51}
        text:"Replace"
        color:1,1,1,1
        size_hint:0.11,0.06
        on_press:root.replacer()
    Label:
        pos_hint:{"x":0.42,"top":0.51}
        size_hint:0.02,0.06
        text:"With "
        color:[0,0,0,1]
    TextInput:
        pos_hint:{"x":0.25,"top":0.51}
        size_hint:0.15,0.06
        id:finder
    TextInput:
        pos_hint:{"x":0.45,"top":0.51}
        size_hint:0.15,0.06
        id:replace
```

```
        CheckBox:
            id:case
            pos_hint:{"x":0.83,"top":0.51}
            size_hint:0.05,0.06
            color:[0,0,0,1]
        CheckBox:
            id:whole
            pos_hint:{"x":0.6,"top":0.51}
            size_hint:0.05,0.06
            color:[0,0,0,1]
        Label:
            pos_hint:{"x":0.65,"top":0.51}
            size_hint:0.15,0.06
            text:"Whole words only"
            color:[0,0,0,1]
        Label:
            pos_hint:{"x":0.85,"top":0.51}
            size_hint:0.15,0.06
            text:"Ignore Case"
            color:[0,0,0,1]
        MDRectangleFlatButton:
            text:"Analyse"
            color:1,1,1,1
            size_hint:0.11,0.06
            pos_hint:{"x":0.1,"top":0.43}
            on_press:root.analyser()
```

Transkivy.py

```
from kivymd.app import MDApp
from googletrans import Translator
import googletrans
import pyperclip
from kivy.uix.button import Button
from kivy.uix.dropdown import DropDown
from kivy.uix.widget import Widget
from kivymd.theming import ThemeManager
from kivy.uix.popup import Popup
from kivy.uix.label import Label


def get_key(val):
    for key, value in googletrans.LANGUAGES.items():
         if val.lower() == value.lower():
             #print(key)
             return key
```

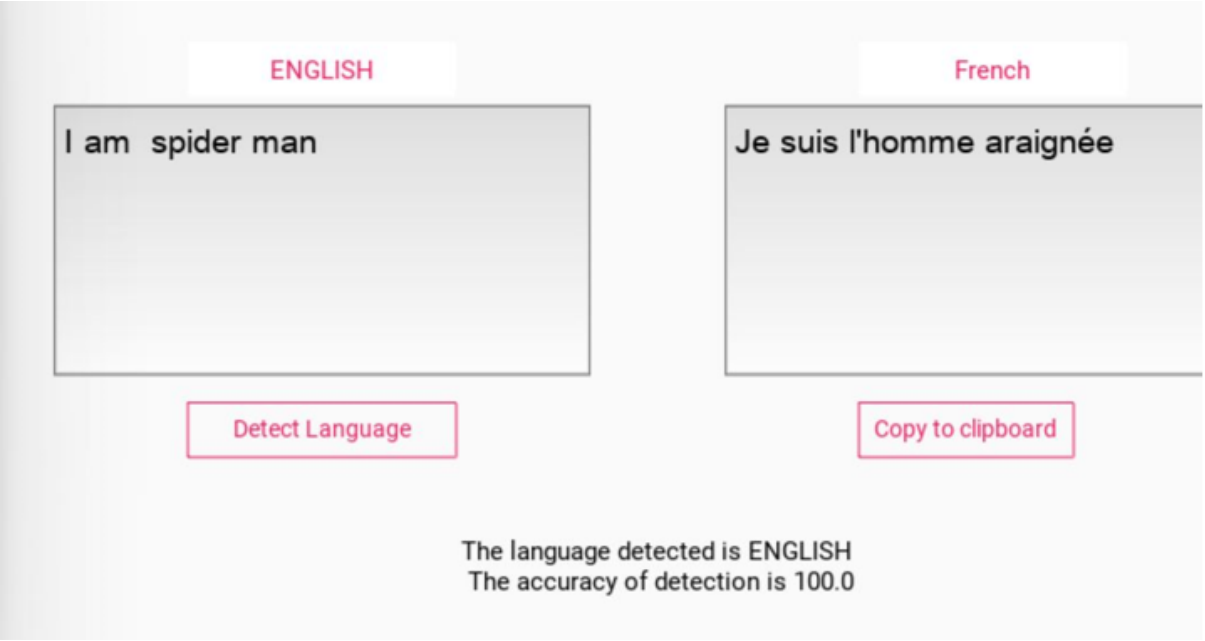
```
    return "error"


class TransLayout(Widget):

    pass
    def copyClip(self):
        #Clipboard.copy(self.ids.res.tcext)
        pyperclip.copy(self.ids.dest.text)
    def translate(self):
        try:

            translator=Translator()
            S=get_key(self.ids.spin2.text)
            if S=='error':
                S='en'
                self.ids.spin2.text='English'
            D=get_key(self.ids.spin1.text)
            if D=='error':
                D='fr'
                self.ids.spin1.text='French'
            res=translator.translate(self.ids.src.text,src=S,dest=D)
            self.ids.dest.text=res.text
        except:
            pop=Popup(title='Some error occurred',content=Label(text='T
ry again',color=(1,1,1,1)),size_hint=(None,None),size=(350,100))
            pop.open()


    def detect(self):
        try:

            #print('the fun works')
            tr=Translator()
            detected=tr.detect(self.ids.src.text)
            L=googletrans.LANGUAGES.get(detected.lang).upper()
            labeltext='The language detected is '+L+'\n The accuracy of
 detection is '+str(detected.confidence*100)
            #print(labeltext)
            self.ids.detect.text=labeltext
            self.ids.spin2.text=L
        except:
            pop=Popup(title='Some error occurred',content=Label(text='T
ry again',color=(1,1,1,1)),size_hint=(None,None),size=(350,100))
            pop.open()
```

myTrans.kv

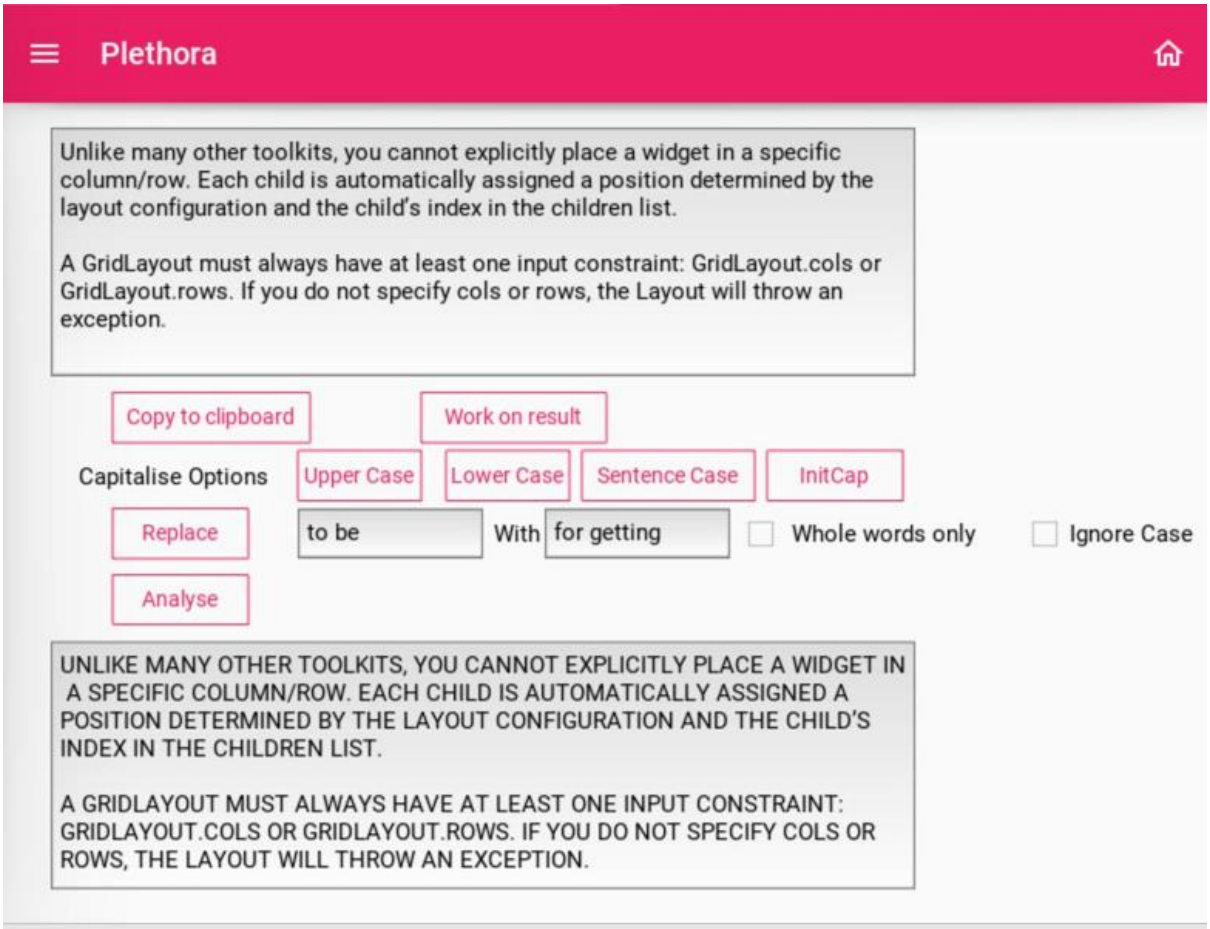
```
#: import googletrans googletrans
<MDRectangleFlatButton>
    background_color: app.theme_cls.primary_color
<SpinnerOption>
    background_normal:''
    color:(0,0,0,1)

<Spinner>
    #background_down:app.theme_cls.primary_dark
    background_normal: ''
    background_color:app.theme_cls.primary_color


<TransLayout>
    FloatLayout:
        size:root.width,root.height

        Spinner:
            id:spin2
            size_hint: 0.2,0.06
            pos_hint: {"x":0.15,"top":0.8}
            text: 'Source Lang'
            values: list(i.capitalize() for i in  googletrans.LANGUAGES
.values())
            #values:['English','Tamil','Hindi','Telugu','Malayalam']


        Spinner:
            id:spin1
            size_hint: 0.2,0.06
            pos_hint: {"x":0.65,"top":0.8}
            text: 'Dest Lang'
            values: list(i.capitalize() for i in  googletrans.LANGUAGES
.values())
            on_text:root.translate()

            #values:['English','Tamil','Hindi','Telugu','Malayalam','Fr
ench']

        MDRectangleFlatButton:
            text:"Copy to clipboard"
            size_hint:0.16,0.06
```

```
        pos_hint:{"top":0.4,"x":0.65}
        on_press:root.copyClip()



    MDRectangleFlatButton:
        text:'Detect Language'
        size_hint:0.2,0.06
        pos_hint:{"top":0.4,"x":0.15}
        on_press:root.detect()

    TextInput:
        id:src
        text:'I am  spider man '
        size_hint:0.4,0.3
        pos_hint:{"top":0.73,"x":0.05}
        font_size:20
        font_name:"arial-unicode-ms.ttf"

    TextInput:
        id:dest
        text:'Here is the result'
        size_hint:0.4,0.3
        pos_hint:{"top":0.73,"x":0.55}
        font_size:20
        font_name:"arial-unicode-ms.ttf"

    Label:
        #size: self.texture_size
        id:detect
        text:'The source language is detected here'
        size_hint:0.4,0.3
        pos_hint:{"top":0.37,"x":0.3}
        color:0,0,0,1
```

Screenshots:



Translate



Text formatter and analyser

4.6 Module 6: Speech

Audio1.py

```
import speech_recognition as sr
import pyttsx3
from kivy.uix.widget import Widget
from kivy.uix.popup import Popup
from kivy.uix.label import Label

class SpeechLayout(Widget):
    def change(self):
        if self.ids.choice.text=='Text to Speech':
            self.ids.sm.current='first'
        else:
            self.ids.sm.current='second'

    def speakOut(self):
        try:
            text=self.ids.tToS.text
            engine = pyttsx3.init()
            engine.say(text)
            engine.setProperty('rate',120)
            engine.setProperty('volume', 0.9)
            engine.runAndWait()
        except:
            pop=Popup(title='Some error occurred',content=Label(text='T
ry again',color=(1,1,1,1)),size_hint=(None,None),size=(350,100))
            pop.open()

    def recog(self):

        r=sr.Recognizer()
        with sr.Microphone() as source:
            audio = r.listen(source)
            try:
                text=r.recognize_google(audio)
                self.ids.result.text=text
            except :
                self.ids.result.text="Sorry Voice Not recognisable"
```

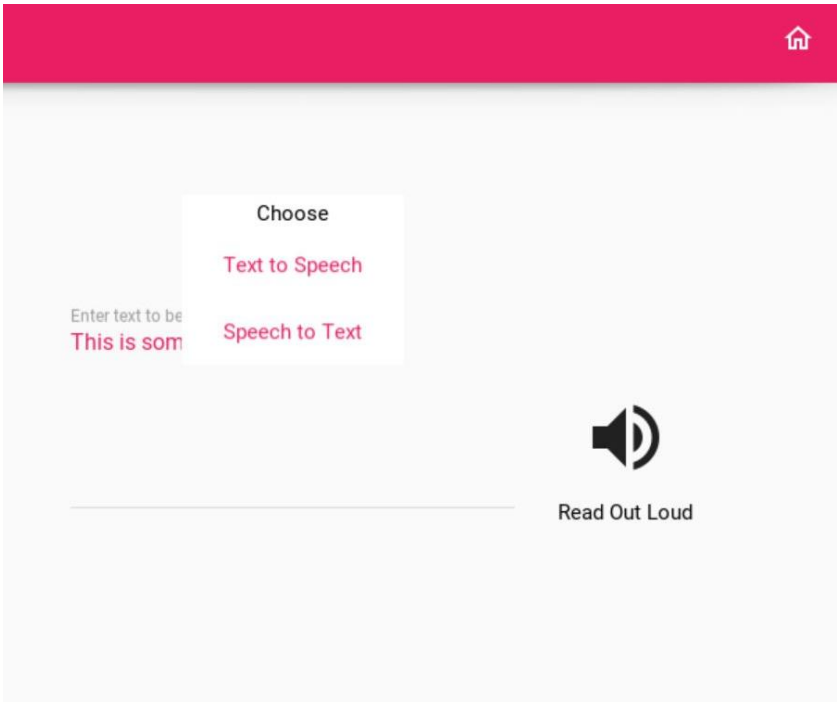
Speech.kv

```
<SpeechLayout>
    FloatLayout:
```

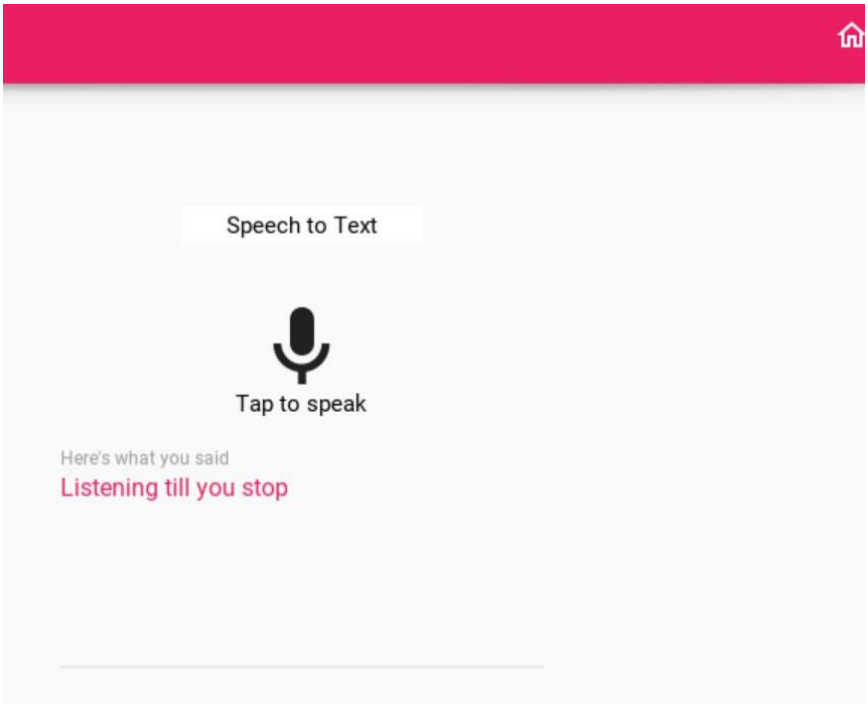
```
size:root.width,root.height
Spinner:
    id:choice
    text:'Choose'
    values:['Text to Speech' , 'Speech to Text']
    pos_hint:{"x":0.4,"top":0.85}
    size_hint:0.2,0.05
    color:0,0,0,1
    on_text:
        root.change()
ScreenManager:
    id:sm
    Screen:
        name:'first'
        MDTextField:
            id:tToS
            size_hint:0.4,0.3
            pos_hint:{"x":0.3,"top":0.7}
            multiline:True
            hint_text:'Enter text to be read'
            text:'This is some sample text'
            required:True
            color_mode:'primary'
            helper_text:'Please Enter TEXT'
            helper_text_mode:'on_error'
        MDIconButton:
            size_hint:0.2,0.05
            pos_hint:{"x":0.7,"top":0.55}
            #text:"Read out loud"
            icon:"volume-high"
            user_font_size:"64sp"
            background_color:app.theme_cls.primary_color
            on_press:root.speakOut()
        Label:
            text:'Read Out Loud'
            pos_hint:{"x":0.7,"top":0.45}
            size_hint:0.2,0.05
            color:0,0,0,1

    Screen:
        name:'second'
        MDIconButton:
            size_hint:0.2,0.05
            pos_hint:{"x":0.4,"top":0.7}
            #text:"Tap to Speak"
            user_font_size:"64sp"
```

```
        icon:'microphone'
        on_release:
            root.recog()

    Label:
        text:'Tap to speak'
        pos_hint:{"x":0.4,"top":0.63}
        size_hint:0.2,0.05


    MDTextField:
        id:result
        size_hint:0.4,0.3
        pos_hint:{"x":0.3,"top":0.55}
        multiline:True
        hint_text:"Here's what you said"
        text:'Listening till you stop'
        color_mode:'primary'
```

Screenshots



Text to speech



Speech to text

5 TESTING OF MODULES

Module testing is defined as a software testing type, which checks individual subprograms, subroutines, classes, or procedures in a program. Instead of testing the whole software program at once, module testing recommends testing the smaller building blocks of the program.

Module testing is largely a white box oriented. The objective of doing Module testing is not to demonstrate proper functioning of the module but to demonstrate the presence of an error in the module.

Module level testing allows to implement parallelism into the testing process by giving the opportunity to test multiple modules simultaneously.

5.1 LOGIN MODULE:

Sno	**Module**	**Test scenario**	**Test case**	**Test Data**	**Expected result**	**Actual result**	**Pass/fail**
1	Login module	Login authentication	Login and Signup	Enter username and password for existing and create new account for others	Accept correct logins and raise error for incorrect ones	Achieved expected result	Pass

Table 1

5.2 OCR module

Sno	**Module**	**Test scenario**	**Test case**	**Test Data**	**Expected result**	**Actual result**	**Pass/fail**
2	Image to text module	Check various images for conversion	Check for the accuracy of conversion	Varieties of images like handwritten, coloured, multiple languages and unclear texts	Right or nearly right text without any gibberish text	Achieved expected result	Pass
		Speech function of converted text	Image converted to text is read out loud	The image with some amount of text in it	Converted text read out loud	Achieved expected result	Pass

Table 2

5.3 Image editing module

Si no	Module name	Test scenario	Test case	Test data	Expected result	Actual result	Pass/fail
3	Image processing module	Check image module	Check whether the required image is properly cropped	An image of correct format or incorrect format browsed	Properly cropped image without quality loss	Properly cropped image without quality loss	Pass
			Check whether the flipped image is obtained	An image of correct format or incorrect format browsed	The exact flipped image of the image inserted.	The exact flipped image of the image inserted.	Pass
			Check whet her the image colour is transformed with clarity	An image of correct format or incorrect format browsed	Image with expected colour transformation	Image with expected colour transformation	Pass
		Accept image	Browse for image file	Correct Image file such as .jpg,.gif, and also incorrect formats	Accept correct formats and raise error for incorrect ones	Achieved expected result	Pass

Table 3

5.4 Text module

Sno	Module name	Test scenario	Test case	Test data	Expected result	Actual result	Pass/fail
4	Text formattin g module	Checking the format of texts	Capitaliz e the required text	Text input	Intended text capitalised	Intended text capitalised	Pass
		Check find and replace with whole words and	Find and replace text	Text input	The intended text is found and replaced	The intended text is found and replaced	Pass

		ignore case functions tested					
		Clipboard function	Add the text to the clipboard	Text input	The intended text is added to the clipboard	The intended text is added to the clipboard	Pass
	Text analysing module	Analyse the character, word and sentence count	Passage given is analysed	A text with large enough words and sentences	Correct number of characters, non-whitespace characters, words, sentences	Achieved expected result	Pass
	Translating module	Check the translation	Passage given is translated	Some text with either language specified or detected	Translate to some destination language	Achieved expected result	Pass

Table 4

5.5 Speech Module

SNo	**Module**	**Test scenario**	**Test case**	**Test Data**	**Expected result**	**Actual result**	**Pass/fail**
5	Speech	Text to speech	Checking audio for a given text	Some non-empty text typed	Get the correct audio with clarity, audibility and pause	Got the result	Pass
		Speech To TEXT	Accept audio from user	Some recognisable sound	Get the text for the audio spoken	Got the result	Pass

Table 5

6 CONCLUSION AND FUTURE ENHANCEMENTS

The project has a great future scope of enhancement especially in the image area. Better processing would enhance the recognition of letters especially by the use of AI and Machine Learning. Better image editing and processing could also be added like filters and collage, etc. Improvement in OCR could make very unclear handwritten notes be converted to text and maybe even documents in specified formats. Feedback from users could be an added value in enhancing the app and to maintain it.

.

7 BIBLIOGRAPHY

1. https://techwithtim.net/tutorials/kivy-tutorial/multiple-screens/
2. https://medium.com/@kuldeepgrewal/kivy-is-an-open-source-cross-platform-python-framework-for-the-development-of-applications-that-4d63199fd4ea
3. https://medium.com/datadriveninvestor/4-simple-steps-in-building-ocr-1f41c66099c1
4. https://github.com/kivy/kivy/wiki/List-of-Kivy-Projects
5. https://pypi.org/project/translate/
6. https://www.geeksforgeeks.org/working-images-python/
7. https://kivymd.readthedocs.io/en/latest/
8. https://kivy.org/doc/stable/
9. https://kivy.org/doc/stable/guide-index.html