#### **TYCS PROJECT 2021-2022**

**NAME: Vikram Satish Chourasiya**

**ROLL NUMBER: 07**

**EMAILID: vikramkumar7208@gmail.com**

**EXAM SEAT NUMBER (SEM V):**

**PROJECT TITLE: PDF 2 PODCAST (PODCAST ENGINE)**

#### **A PROJECT REPORT**

**ON**

**(PODCST ENGINE)**

**[Seat No:374358]**

**UNDER THE GUIDANCE OF**

**Prof: SHRADHA BIRJE**

**SUBMITTED IN PARTIAL FULFILLMENT OF ACADEMIC PROJECT**

**[Bachelors of Science Computer Science]**



**UNIVERSITY OF MUMBAI**

**SHANKAR NARAYAN COLLEGE OF ART, COMMERCE AND SCIENCE**

(DEPARTMENT OF BSC-CS)

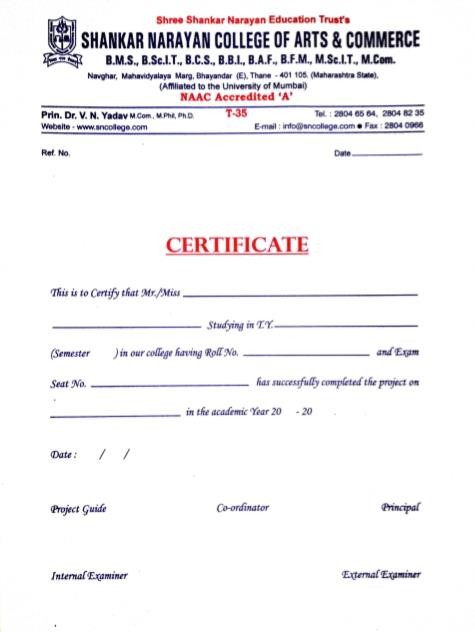
Academic Year 2021-22

## **ACKNOWLEDGEMENT**

## This project could not have been accomplished if not for the direct or indirect contribution from many known and unknown individuals. I would like to express my special thanks of gratitude to my guide Prof. SHEADHA BIRJE who gave us unending support from the initial stage of project. A source of inspiration to us. I would also like to thank our H.O.D. Miss. SMITA DALVI. The foundation that we have been able to develop today owes much credit to her. I am really thankful for her cooperation and guidance. I would even like to thank my parents and friends who helped me a lot in my project within the limited time frame. I have learned new things and this experience is my real achievement.

## Thank you.

## VIKRAM CHOURASIYA



**INDEX**

|  |  |  |
| --- | --- | --- |
| **NO:** | **CONTENTS** | **PAGE** |
| **1. PRELIMINARY INVESTIGATION** | | |
| 1.1 | Introduction | **1** |
| 1.2 | Objective | **2** |
| 1.3 | System Description and Limitation | **3** |
| 1.4 | System Requirement | **4** |
| 1.5 | Activity Sheet | **5** |
| 1.6 | Gantt Chart | **6** |
| 1.7 | Fact Finding | **7** |
| **2. SYSTEM ANALYSIS** | | |
| 2.1 | Process Model | **10** |
| 2.2 | Event Table | **11** |
| 2.3 | Use-case Diagram | **15** |
| 2.4 | E-R Diagram | **17** |
| 2.5 | Class Diagram | **18** |
| 2.6 | Activity Diagram | **21** |
| 2.7 | Flow Chart | **24** |
| 2.8 | Sequence Diagram | **25** |
| **3. SYSTEM DESIGN** | | |
| 3.1 | System Analysis and Design | **26** |
| **4. SYSTEM CODING** | | |
| 4.1 | Screen layouts and User Manual | **28** |
| **5. MAINTENANCES AND EVALUTION** | | |
| 5.1 | Conclusion | **30** |
| 5.2 | Future Recommendation | **31** |
| 5.3 | References and Bibliography | **32** |
|  |  |  |

**Preliminary**

**Investigation**

**1.1 Introduction:**

This Project is selected because of the unique and interesting approach it provides to book readers experience. The basic motive behind the project (digital interface of reading book) is to create innovative platform for people who really loves reading and support the popular culture. The multifactor text to read or listen feature factor is included to create a certain sense of which type of experience are really needed for society among the book lovers.

Main focus is on the vast collection of books present in our world. The art of competing books to each other is carried on over the century that is why this project include a converting mechanism.

Due to advantages in ICT, we can now read books on the go on our smartphones, smart TVs and many other devices. The growing possibilities of modern communication provides us the new mediums which allows us to view books anytime, anywhere and to anyone. Such a new medium is called as podcast streamers platforms.

But this project is lot different than other podcast platforms because it allows you to decide which book you need to convert you want to see and gives you a chance to support your decided books.

This will result in an explosive growth between book reading community.

**1.2 Objectives:**

The goal of Podcast engine is to provide converting PDF. So, a fundamental requirement of this project is to know everything about books. The other goal is to provide the experience of podcast.

This project has following objectives:

* To explore varieties of books, form different publishers, writers.
* To classify them on basis of public review and their quality no matter which language or budget.
* To have extremely good reading experience for users.
* To listen in two different voices.

My project will try its best to achieve these objectives for the growth purpose.

**1.3 System Description:**

This is an application-based system where users can find two option that how to convert PDF. The application has a special setting in which it has four themes for more user experience. Application has a text box where users can write their own data to be converted.

**Limitations of Present System:**

This project is in current stage with limited features and a very major limitation is that once a user converted PDF it can be listen through and saved as well.

**1.4 System Requirements:**

**Software Requirement:**

* Visual Studio Code.
* Python3
* Anaconda navigator
* Python packages

**Languages used:**

* Front End: Python3
* Back End: Python pillow

**Hardware Requirement:**

### **Processor: 1.0 GHz or Greater.**

### **RAM: 2GB or Greater.**

### **Activity Sheet:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.no.** | **Activity** |  | | **Sign** |
| **Planned Date** | **Actual Date** |
| 1. | **Preliminary Investigation** | 15/09/2021 | 24/09/2021 |  |
| 2. | **System Study and Analysis** | 25/09/2021 | 09/10/2021 |  |
| 3. | **System Development** | 09/10/2021 | 12/10/2021 |  |
| 4. | **System Coding and Report** | 13/10/2021 | 14/11/2021 |  |
| 5. | **Project Submission** | 15/11/2021 | 26/11/2021 |  |

**Fact Finding:**

I only used two main techniques to find out correct information for further growth of my project.

The techniques are:

1. Observation
2. Questionnaire
3. **Observation:**

After certain time in Research, I find that there are lots of converting platforms which provide digital converge service but I hardly saw anyone trying to compete movies. So, that’s become the main theme for my project.

### **2.Questionnaire:**

### **A set of questionnaires is prepared to find out some hiding**

### **issues in the system.**

### **Is there any limitation bounding the system?**

### **What is the number of users of the system?**

### **How many admins are there in system?**

### **What is recommended front-end?**

### **What is recommended back-end?**

### **What is the time limit for the development of the system?**

### **What are the return benefits of the system?**

### **Is there any recommended format of the user interface?**

### **What are the hardware specifications?**

### **What are the software specifications?**

System

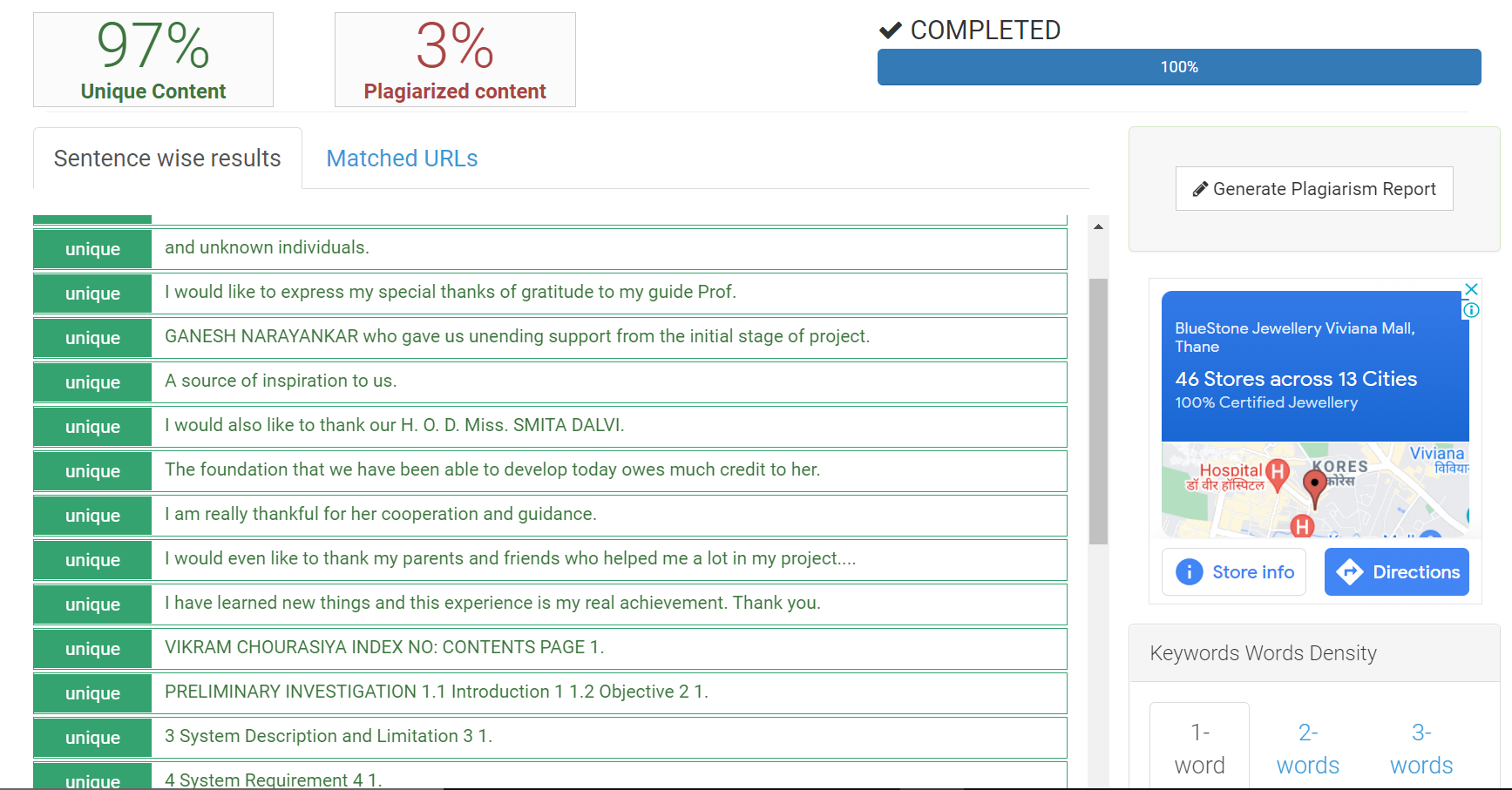
Analysis

## **Process Model:**

## The waterfall model of development has been taken into consideration while developing this project. Every step for updating or improving as this model ensures smooth flow of project development.

## 

**PLAGIARISM REPORT:**

****

**Gantt Chart:**

|  |  |  |  |
| --- | --- | --- | --- |
| **MONTH**  **PROCESS** | **AUG**  **2021** | **SEP**  **2021** | **OCT 2021** |
| **PERLIMINARY**  **INVESTIGATION** |  |  |  |
| **SYSTEM**  **ANALYSIS** |  |  |  |
| **SYSTEM**  **DESIGN** |  |  |  |
| **MAINTENANCE**  **&**  **EVAUATION** |  |  |  |

**- PLANNED DATE**

**- ACTUAL DATE**

# **Podcast Engine**

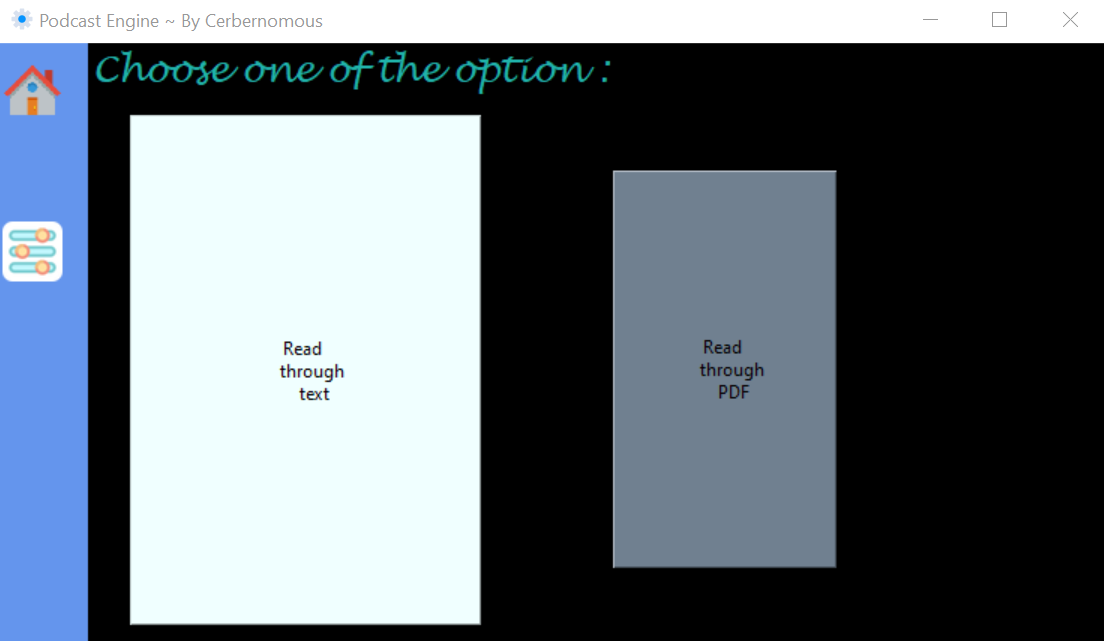
~ Created by **"Cerbrenomous"**

**A python application purely made for the people who like to get lost in harmony.**

# Starting Page:

[](https://github.com/VikramChourasiya-07/Podcast_Engine/blob/main/page_intro.PNG)

# Get Start With:

[](https://github.com/VikramChourasiya-07/Podcast_Engine/blob/main/page_work.PNG)

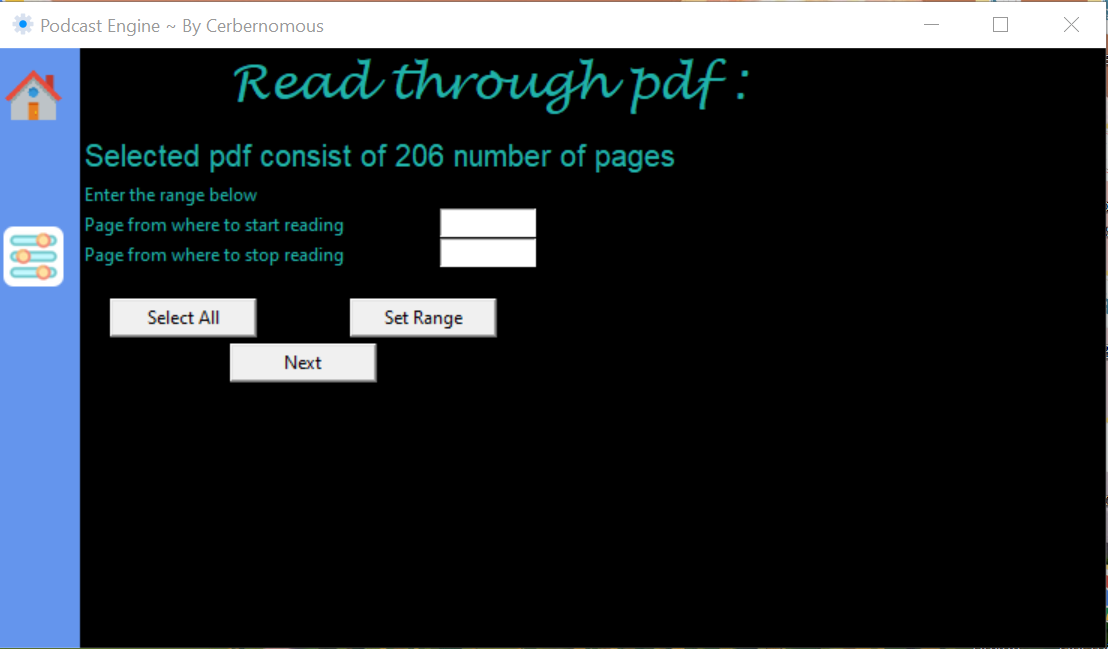
# Play Via Text:

[](https://github.com/VikramChourasiya-07/Podcast_Engine/blob/main/page_text_R.PNG)

# Play Via PDF:

[](https://github.com/VikramChourasiya-07/Podcast_Engine/blob/main/page_read_pdf.PNG)

# play Control:

[](https://github.com/VikramChourasiya-07/Podcast_Engine/blob/main/page_play.PNG)

* here user can put required output to application to read and forward output.

# For Non-Coders/Coders -install setup [ just install setup]

Link to download exe file (Window Installer) *Google Drive:  
Mega:*Link to download code (Source Code) *GitHub:  
Google Drive :*

# 

# Function:

Text to Audio and PDF to audio (OCR)

There are two functions in this application

**1:**It can read text in two voices.  
Male (windows David)  
Female (windows Zira)

**2:**It can convert PDF file in two voices.  
*It use Tesaerate OCR for voice support.*  
Male (windows David)  
Female (windows Zira)

# Important notes

To run this Application user, need following packages.  
  
Require Tesseract OCR and Popper Windows tts voices David and Zira are added in the registry Home and Settings icons are from flat icons and I don't own it.

#Installation and Setup of requirements

For Tesseract OCR

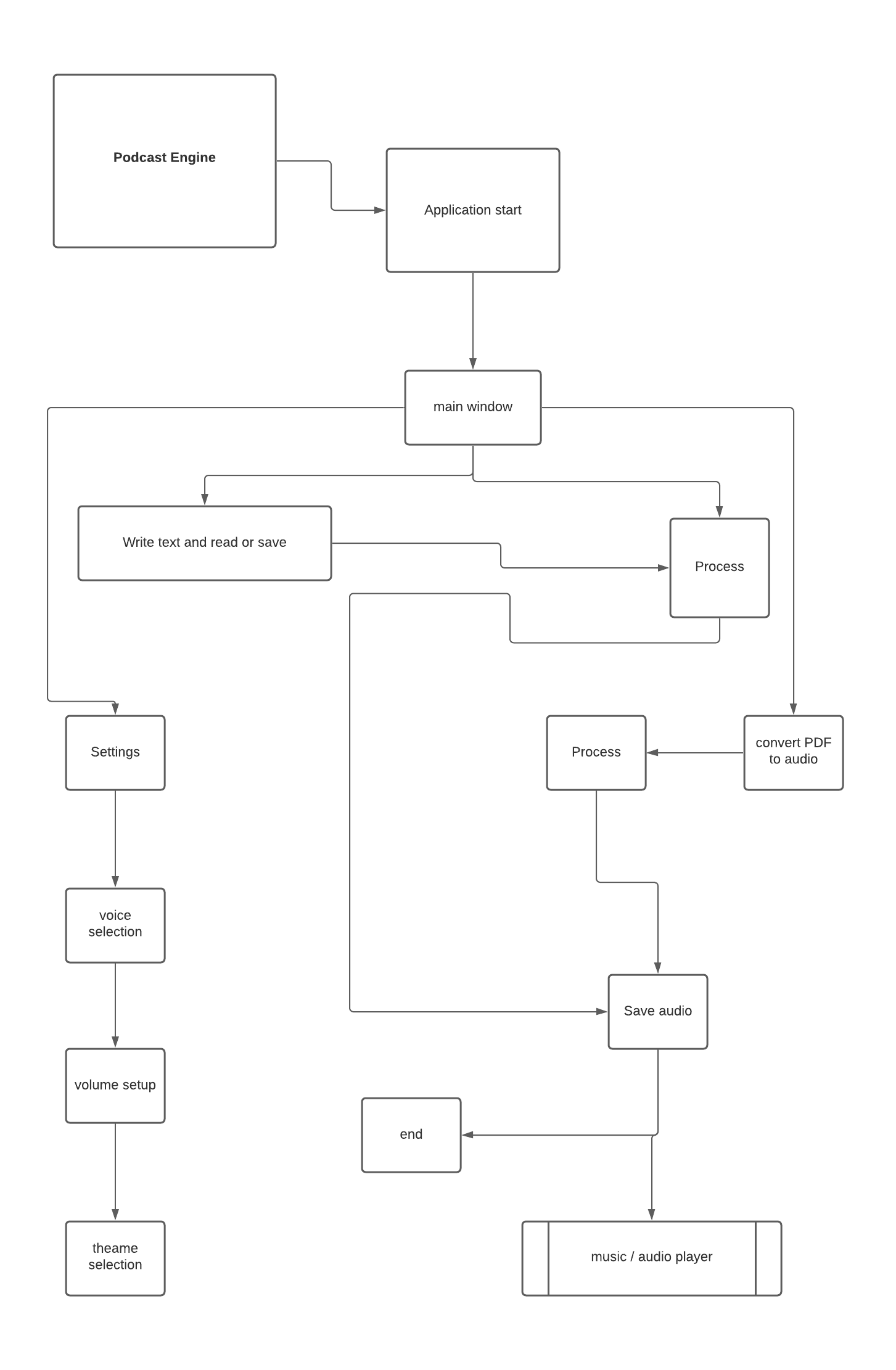
go to <https://github.com/UB-Mannheim/tesseract/wiki> and select the version 5.0.0 alpha add new system variable as PATH and give address as C:\Program Files\Tesseract-OCR

For Popper

go to <http://blog.alivate.com.au/poppler-windows/> and select poppler-0.68.0\_x86 ,then extract it and copy to C:\Program Files edit your PATH variable in system variable and add new address as C:\Program Files\poppler-0.68.0

**Activity Diagram:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with iteration and concurrency. Activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of processes.



**MAINTENANCES**

**AND**

**EVALUTION**

### **Conclusion:**

I came to this point where I know my project very well. I know its plus points and backdrops.

Movies has certain kind of magic to gather different kind of under same roof, I hope my project has same kind of power to gather different kind of users with different kind of taste for movies. The main focus of my project is to provide best digital movie watching experience.

Everything is becoming digital and streaming oriented and I conclude that this website can be essential for this digital world

### **Future Requirement:**

The major limitation of the system is lack of features and not having much time-consuming elements.

The future work on this project is to improve the interfaces and to have many innovative themes and adding different film genre

**PYTHON CODE:**

import os

from os import close

from tkinter import \*

from tkinter import ttk

from tkinter import scrolledtext

import tkinter

from tkinter.font import BOLD

from tkinter.scrolledtext import ScrolledText

from tkinter.messagebox import showinfo

from typing import Iterable, Tuple

import pandas as pd

import PIL.ImageTk,PIL.Image

import pytesseract as pt

import pdf2image

import PyPDF2

import pyttsx3

from pyttsx3.drivers import sapi5

from tkinter.filedialog import \*

sidebar = "#6495ED"

mainbg = "black"

label\_colour = "lightseagreen"

button\_colour = "#9932CC"

light\_colour = 'slate gray'

dark\_colour= "azure"

voiceid=0

rate\_value=150

book = None

content = None

start= None

end = None

current\_value=150

*#background\_image=ImageTk.PhotoImage(file='name\_of\_the\_image.extension')*

desktop = os.path.join(os.path.join(os.environ['USERPROFILE']), 'Desktop')

home\_icon='home.png'

settings\_icon='settings.png'

root = Tk()

root.title("Podcast Engine ~ By Cerbernomous")

root.maxsize('738', '400')

root.minsize('738', '400')

root.geometry('738x400')

root.iconbitmap('logo.ico')

root.configure(bg=mainbg)

min\_w = 60 *# Minimum width of the frame*

max\_w = 120 *# Maximum width of the frame*

cur\_width = min\_w *# Increasing width of the frame*

expanded = False *# Check if it is completely exanded*

ex=False

def expand():

    global cur\_width, expanded

    cur\_width += 10 *# Increase the width by 10*

    rep = root.after(5,expand) *# Repeat this func every 5 ms*

    frame.config(width=cur\_width) *# Change the width to new increase width*

    if cur\_width >= max\_w: *# If width is greater than maximum width*

        expanded = True *# Frame is expended*

        root.after\_cancel(rep) *# Stop repeating the func*

        fill()

def contract():

    global cur\_width, expanded

    cur\_width -= 10 *# Reduce the width by 10*

    rep = root.after(5,contract) *# Call this func every 5 ms*

    frame.config(width=cur\_width) *# Change the width to new reduced width*

    if cur\_width <= min\_w: *# If it is back to normal width*

        expanded = False *# Frame is not expanded*

        root.after\_cancel(rep) *# Stop repeating the func*

        fill()

def fill():

    if expanded: *# If the frame is exanded*

*# Show a text, and remove the image*

        home\_b.config(text='Home',image='',font=(0,18),bg=sidebar,command=None)

        set\_b.config(text='Settings', image='', font=(0,18), bg=sidebar, command=None)

    else:

*# Bring the image back*

        home\_b.config(image=home,font=(0,21))

        set\_b.config(image=settings,font=(0,21))

*#background\_label =Label(root, image=background\_image)*

*#background\_label.place(x=0,y=0)*

*# Adding Widgets....................*

def win1():

    labelstart.destroy()

    labelstart1.destroy()

    butstart.destroy()

    credit.destroy()

    def expandbut1():

        global cur\_width, ex

        cur\_width += 2 *# Increase the width by 2*

        rep = root.after(5,expand) *# Repeat this func every 5 ms*

        but1.config(width=28) *# Change the width to new increase width*

        but1.config(height=22)

        but2.config(bg= light\_colour)

        but2.config(width=20)

        but2.config(height=17)

        if cur\_width >= 28: *# If width is greater than maximum width*

            ex = True *# Frame is expended*

            root.after\_cancel(rep) *# Stop repeating the func*

            fill()

        if cur\_width <= 25: *# If width is greater than maximum width*

            ex = True *# Frame is expended*

            root.after\_cancel(rep) *# Stop repeating the func*

            fill()

    def contractbut1():

        global cur\_width, ex

        cur\_width -= 2 *# Reduce the width by 10*

        rep = root.after(5,contract) *# Call this func every 5 ms*

        but1.config(width=25) *# Change the width to new reduced width*

        but1.config(height=20)

        but2.config(width=25) *# Change the width to new reduced width*

        but2.config(height=20)

        but2.config(bg=dark\_colour)

        if cur\_width <= 25: *# If it is back to normal width*

            ex = False *# Frame is not expanded*

            root.after\_cancel(rep) *# Stop repeating the func*

            fill()

        if cur\_width >= 28: *# If width is greater than maximum width*

            ex = False *# Frame is expended*

            root.after\_cancel(rep) *# Stop repeating the func*

            fill()

    def expandbut2():

        global cur\_width, ex

        cur\_width += 2 *# Increase the width by 2*

        rep = root.after(5,expand) *# Repeat this func every 5 ms*

        but2.config(width=28) *# Change the width to new increase width*

        but2.config(height=22)

        but1.config(bg=light\_colour)

        but1.config(width=20)

        but1.config(height=17)

        if cur\_width >= 28: *# If width is greater than maximum width*

            ex = True *# Frame is expended*

            root.after\_cancel(rep) *# Stop repeating the func*

            fill()

        if cur\_width <= 25: *# If width is greater than maximum width*

            ex = True *# Frame is expended*

            root.after\_cancel(rep) *# Stop repeating the func*

            fill()

    def contractbut2():

        global cur\_width, ex

        cur\_width -= 2 *# Reduce the width by 10*

        rep = root.after(5,contract) *# Call this func every 5 ms*

        but2.config(width=25) *# Change the width to new reduced width*

        but2.config(height=20)

        but1.config(bg=dark\_colour)

        but1.config(width=25) *# Change the width to new reduced width*

        but1.config(height=20)

        if cur\_width <= 25: *# If it is back to normal width*

            ex = False *# Frame is not expanded*

            root.after\_cancel(rep) *# Stop repeating the func*

            fill()

        if cur\_width >= 28: *# If width is greater than maximum width*

            ex = False *# Frame is expended*

            root.after\_cancel(rep) *# Stop repeating the func*

            fill()

    def read\_through\_text():

        but1.destroy()

        but2.destroy()

        label1.destroy()

        def speak():

            player = pyttsx3.init()

            audio\_string = text.get('0.0',END)

            voices = player.getProperty('voices')

            player.setProperty('voice', voices[voiceid].id)

            player.setProperty('rate', rate\_value)

            if audio\_string:

                player.say(audio\_string)

                player.runAndWait()

                player.stop()

        def save\_aud():

            player = pyttsx3.init()

            audio\_string = text.get('0.0',END)

            voices = player.getProperty('voices')

            player.setProperty('voice', voices[voiceid].id)

            player.setProperty('rate', rate\_value)

            if audio\_string:

                player.save\_to\_file(audio\_string,asksaveasfilename(defaultextension='.mp3',filetypes=(("audio/mpeg", "\*.mp3"),("All Files", "\*.\*"))))

                player.runAndWait()

                player.stop()

        def bac():

            label2.destroy()

            text.frame.destroy()

            listen\_b.destroy()

            clear\_b.destroy()

            save\_b.destroy()

            win1()

*# Adding Widgets....................*

        label2 = Label(root,text="Type the text below :",font=("Lucida Handwriting",18),background=mainbg,fg=label\_colour)

        label2.grid(column=1,row=0,columnspan=3)

        text = ScrolledText(root,width=60,height=19,wrap = WORD,padx= 10, pady= 10,bd=5,relief = RIDGE)

        text.grid(row=1,column=2,columnspan=3)

*# Adding Buttons....................*

        listen\_b = ttk.Button(root,text='Listen',width=7,command=speak,)

        listen\_b.grid(row=2,column=2,ipadx=2)

        clear\_b = ttk.Button(root,text='clear',width=7,command=lambda: text.delete('0.0',END))

        clear\_b.grid(row=2,column=3,ipadx=2)

        save\_b = ttk.Button(root,text='save',width=7,command=save\_aud)

        save\_b.grid(row=2,column=4,ipadx=2)

        home\_b.config(image=home,font=(0,18),bg=sidebar,command=bac)

    def read\_through\_pdf():

        but1.destroy()

        but2.destroy()

        label1.destroy()

        pt.pytesseract.tesseract\_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"

        label3 = Label(root,text="Read through pdf :",font=("Lucida Handwriting",24),bg=mainbg,fg=label\_colour)

        label3.grid(column=1,row=0)

        def browse():

            browse\_b.destroy()

            global book,content

            book = askopenfilename()

            pdfReader = PyPDF2.PdfFileReader(book)

            no\_page = pdfReader.numPages

            no\_pagestr = "Selected pdf consist of "+str(no\_page)+" number of pages"

            page\_dis = Label(root,text=no\_pagestr,font= 20,background=mainbg,foreground=label\_colour)

            page\_dis.grid(row=1,column=1,sticky='nw',pady=10)

            range\_mess = Label(root,text='Enter the range below',background=mainbg,foreground=label\_colour)

            range\_mess.grid(row=1,column=1,sticky='nw',pady=40)

            st = Label(root,text='Page from where to start reading',background=mainbg,foreground=label\_colour)

            st.grid(row=1,column=1,sticky=NW,pady=60)

            txt1= Entry(root,width=10)

            txt1.grid(column=1,row=1,sticky=NW,pady=60,padx=240)

            en = Label(root,text='Page from where to stop reading',background=mainbg,foreground=label\_colour)

            en.grid(row=1,column=1,sticky=NW,pady=80)

            txt2= Entry(root,width=10)

            txt2.grid(column=1,row=1,sticky=NW,pady=80,padx=240)

            def all1():

                global start,end

                start = 0

                end = no\_page

                allb.config(text='Selected',command=None)

            def play1():

                global start,end

                start = int(txt1.get())-1

                end = int(txt2.get())

                ren.config(text='Setted',command=None)

            ren = Button(root,text='Set Range',width=12,command=play1)

            ren.grid(row=1,column=1,ipadx=2,sticky=NW,pady=120,padx=180)

            allb = Button(root,text='Select All',width=12,command=all1)

            allb.grid(row=1,column=1,ipadx=2,sticky=NW,pady=120,padx=20)

            def play\_button():

                play\_bu.destroy()

                def speak():

                    player = pyttsx3.init()

                    voices = player.getProperty('voices')

                    player.setProperty('voice', voices[voiceid].id)

                    player.setProperty('rate', rate\_value)

                    pages = pdf2image.convert\_from\_path(pdf\_path = book, dpi=200, size=(1654,2340))

                    for i in range(start,end):

                        content = pt.image\_to\_string(pages[i], lang='eng')

                        if content:

                            player.say(content)

                            player.runAndWait()

                            player.stop()

                play\_aud = Button(root,text='Play',width=12,command=speak)

                play\_aud.grid(row=1,column=1,ipadx=2,sticky=NW,pady=155,padx=20)

                def save\_aud():

                    player = pyttsx3.init()

                    voices = player.getProperty('voices')

                    player.setProperty('voice', voices[voiceid].id)

                    player.setProperty('rate', rate\_value)

                    pages = pdf2image.convert\_from\_path(pdf\_path = book, dpi=200, size=(1654,2340))

                    for i in range(start,end):

                        content = pt.image\_to\_string(pages[i], lang='eng')

                        if content:

                            player.save\_to\_file(content,desktop+'/AudioConvertorpage'+str(i+1)+'.mp3')

                            player.runAndWait()

                            player.stop()

                save\_au = Button(root,text='Save',width=12,command=save\_aud)

                save\_au.grid(row=1,column=1,ipadx=2,sticky=NW,pady=155,padx=180)

                def bac():

                    label3.destroy()

                    page\_dis.destroy()

                    range\_mess.destroy()

                    st.destroy()

                    txt1.destroy()

                    en.destroy()

                    txt2.destroy()

                    ren.destroy()

                    allb.destroy()

                    play\_aud.destroy()

                    save\_au.destroy()

                    play\_bu.destroy()

                    win1()

                home\_b.config(image=home,font=(0,18),bg=sidebar,command=bac)

            play\_bu = Button(root,text='Next',width=12,command=play\_button)

            play\_bu.grid(row=1,column=1,ipadx=2,sticky=NW,pady=150,padx=100)

            def bac():

                label3.destroy()

                page\_dis.destroy()

                range\_mess.destroy()

                st.destroy()

                txt1.destroy()

                en.destroy()

                txt2.destroy()

                ren.destroy()

                allb.destroy()

                play\_bu.destroy()

                win1()

            home\_b.config(image=home, font=(0,18), bg=sidebar, command=bac)

        browse\_b = ttk.Button(root, text='Browse the pdf file.', width=16, command=browse)

        browse\_b.grid(row=1,column=1,ipadx=2,sticky=NW)

        def bac():

            browse\_b.destroy()

            label3.destroy()

            win1()

        home\_b.config(image=home, font=(0,18), bg=sidebar, command=bac)

*# Read a pdf file as image pages*

*# We do not want images to be to big, dpi=200*

*# All our images should have the same size (depends on dpi), width=1654 and height=2340*

    Grid.rowconfigure(root,0,weight=0)

    Grid.columnconfigure(root,0,weight=0)

    Grid.rowconfigure(root,1,weight=1)

    Grid.columnconfigure(root,1,weight=0)

*#ku = ImageTk.PhotoImage(Image.open('ku.png'))*

    label1 = Label(root,text="Choose one of the option :",font=("Lucida Handwriting",18),background=mainbg,fg=label\_colour)

    label1.grid(column=1,row=0,columnspan=5)

    longtext1 = '''Read

    through

    text'''

    but1 = Button(root,text=longtext1,width=25,height=20,command=read\_through\_text,bg=dark\_colour)

    but1.grid(row=1,column=2,sticky= EW)

    longtext2 = '''Read

    through

    PDF'''

    but2 = Button(root,text=longtext2,width=25,height=20,command=read\_through\_pdf,bg=dark\_colour)

    but2.grid(row=1,column=8,sticky=EW)

    but1.bind('<Enter>',lambda e: expandbut1())

    but1.bind('<Leave>',lambda e: contractbut1())

    but2.bind('<Enter>',lambda e: expandbut2())

    but2.bind('<Leave>',lambda e: contractbut2())

    def sett1():

        root1 = Tk()

        root1.geometry('580x450')

        root1.title("Settings")

        def david():

            global voiceid

            voiceid=1

        def zira():

            global voiceid

            voiceid=2

        def bac():

            root1.destroy()

        saveb = ttk.Button(root1,text='Save',width=12,command=bac)

        saveb.grid(row=11,column=1,ipadx=2)

        label3 = Label(root1,text="Settings:",font=("Lucida Handwriting",24))

        label3.grid(column=1,row=0)

        label3\_1 = Label(root1,text="Voice",font=("Lucida Handwriting",18))

        label3\_1.grid(column=0,row=1,sticky=NW)

        label3\_2 = Label(root1,text="Choose the Voice you want :",font=("Lucida Handwriting",12))

        label3\_2.grid(column=1,row=2,sticky=NW)

        david\_b = ttk.Button(root1,text='David (Male)',width=12,command=david)

        david\_b.grid(row=3,column=1,ipadx=2,sticky=NW)

        zira\_b = ttk.Button(root1,text='Zira (Female)',width=12,command=zira)

        zira\_b.grid(row=3,column=3,ipadx=2,sticky=NE)

        label4 = Label(root1,text="Rate",font=("Lucida Handwriting",18))

        label4.grid(column=0,row=4,sticky=NW)

        label4\_1 = Label(root1,text="Select the rate : ",font=("Lucida Handwriting",12))

        label4\_1.grid(column=1,row=5,sticky=NW)

*# slider current value*

        def get\_current\_value():

            global rate\_value

            global current\_value

            val=int(slider.get())

            rate\_value=val

            current\_value=val

            return '{: .2f}'.format(val)

        def slider\_changed(event):

            value\_label.configure(text=get\_current\_value())

*# label for the slider*

        slider\_label = ttk.Label(root1,text='Slider:')

        slider\_label.grid(column=0,row=6,sticky='w')

*#  slider*

        slider = ttk.Scale(root1,from\_=50,to=350,value=current\_value,orient='horizontal', command=slider\_changed)

        slider.grid(column=1,row=6,sticky='we')

*# current value label*

        current\_value\_label = ttk.Label(root1,text='Current Value:')

        current\_value\_label.grid(row=7,columnspan=2,column=1,sticky='n',ipadx=10,ipady=1)

*# value label*

        value\_label = ttk.Label(root1,text=get\_current\_value())

        value\_label.grid(row=7,columnspan=2,column=2,sticky=N)

        label5 = Label(root1,text="Themes",font=("Lucida Handwriting",18))

        label5.grid(column=0,row=8,sticky=NW)

        def homewarn():

            label5\_1 = Label(root1,text="(Please, Return to home to update latest theme)",font=("Lucida Handwriting",10),fg='red')

            label5\_1.grid(column=1,row=8,sticky=NW)

        def theme1():

            global sidebar ,mainbg ,label\_colour

            sidebar= "sky blue"

            mainbg= "grey"

            label\_colour="white"

            frame.config(bg=sidebar)

            root.config(bg=mainbg)

            home\_b.config(bg=sidebar)

            set\_b.config(bg=sidebar)

            label1.config(bg=mainbg)

            homewarn()

        the1 = Button(root1,text='Gold Black',width=12,height=4,command=theme1)

        the1.grid(row=9,column=1,ipadx=2,sticky=NW)

        def theme2():

            global sidebar ,mainbg ,label\_colour

            sidebar= "gold"

            mainbg= "black"

            label\_colour="white"

            frame.config(bg=sidebar)

            root.config(bg=mainbg)

            home\_b.config(bg=sidebar)

            set\_b.config(bg=sidebar)

            label1.config(bg=mainbg)

            homewarn()

        the2 = Button(root1,text='Lavender',width=12,height=4,command=theme2)

        the2.grid(row=9,column=7,ipadx=2,sticky=NE)

        def theme3():

            global sidebar ,mainbg ,label\_colour

            sidebar= "lavender"

            mainbg= "misty rose"

            label\_colour="Black"

            frame.config(bg=sidebar)

            root.config(bg=mainbg)

            home\_b.config(bg=sidebar)

            set\_b.config(bg=sidebar)

            label1.config(bg=mainbg,fg=label\_colour)

            homewarn()

        the3 = Button(root1,text='Indigo',width=12,height=4,command=theme3)

        the3.grid(row=10,column=1,ipadx=2,sticky=SW)

        def theme4():

            global sidebar ,mainbg ,label\_colour

            sidebar= "indigo"

            mainbg= "steel blue"

            label\_colour="white"

            frame.config(bg=sidebar)

            root.config(bg=mainbg)

            home\_b.config(bg=sidebar)

            set\_b.config(bg=sidebar)

            label1.config(bg=mainbg,fg=label\_colour)

            homewarn()

        the4 = Button(root1,text='Theme 4',width=12,height=4,command=theme4)

        the4.grid(row=10,column=7,ipadx=2,sticky=SE)

        root1.columnconfigure(0, weight=1)

        root1.columnconfigure(1, weight=3)

        root1.mainloop()

    set\_b.config(image=settings,font=(0,18),bg=sidebar,command=sett1)

labelstart=Label(root,text='Welcome to Harmony',font=('Freestyle Script',22),fg='crimson',bg=mainbg)

labelstart.grid(row=1,column=1,padx=130)

labelstart1=Label(root,text='Podcast Engine',font=('Lucida Handwriting',34),fg='crimson',bg=mainbg)

labelstart1.grid(row=2,column=1,padx=130)

butstart = Button(root,text='Start', width=25,command=win1)

butstart["background"] = "#FFD700"

butstart.grid(row=4,column=1,padx=150)

def crie():

    cre = Tk()

    cre.title("Creator Of Application")

    cre.iconbitmap('about\_logo.ico')

    cre.geometry("500x300")

    backg="#7FFFD4"

    cre.config(bg=backg)

    label\_head = Label(cre,text='Credits',font=("Bookman Old Style", 50, BOLD),anchor=CENTER,fg='dark green',bg=backg)

    label\_head.pack()

    label\_name1= Label(cre,text='Vikram Chourasiya',font=("Lucida Handwriting",32,BOLD),fg='dark green',bg=backg)

    label\_name1.pack()

    cre.mainloop()

credit= Button(root,text='i',width=2,command=crie)

credit.grid(row=0,column=5,sticky=NE)

home = PIL.ImageTk.PhotoImage(PIL.Image.open(home\_icon).resize((40,40),PIL.Image.ANTIALIAS))

settings = PIL.ImageTk.PhotoImage(PIL.Image.open(settings\_icon).resize((40,40),PIL.Image.ANTIALIAS))

root.update() *# For the width to get updated*

frame = Frame(root,bg=sidebar,width=55,height=400)

frame.grid(row=0,column=0,rowspan=11,sticky=NS)

*# Make the buttons with the icons to be shown*

home\_b = Button(frame,image=home,bg=sidebar,relief='flat')

set\_b = Button(frame,image=settings,bg=sidebar,relief='flat')

*# Put them on the frame*

home\_b.grid(row=0,column=0,pady=10)

set\_b.grid(row=1,column=0,pady=50)

*# Bind to the frame, if entered or left*

frame.bind('<Enter>',lambda e: expand())

frame.bind('<Leave>',lambda e: contract())

*# So that it does not depend on the widgets inside the frame*

frame.grid\_propagate(False)

root.mainloop()

### **References:**

**YouTube**

**Google**

**Stack overflow**

**GitHub**

### **Website:**

<http://www.google.com>

<http://www.microsoft.com>

<http://www.vbcode.com>

<http://www.codeproject.com>

### **Books:**

Python and Tkinter: Design and Build Application (Paperback)