## Contents

[Contents 1](#_Toc65416041)

[**1.** **ACKNWOLEDGEMENT** 2](#_Toc65416042)

[**2.** **CASE STUDY OF THE PROJECT** 3](#_Toc65416043)

[2.1. PROJECT​: 3](#_Toc65416044)

[2.2. DETAILS​: 3](#_Toc65416045)

[2.3. DESIGN IDEA​: 3](#_Toc65416046)

[2.4. MODULES USED: 3](#_Toc65416047)

[**3.** **SCREENSHOTS OF TABLES OR DESCRIPTIONS OF TABLES** 4](#_Toc65416048)

[**4.** **PROCESS DIAGRAM** 5](#_Toc65416049)

[**5.** **FLOW CHARTS** 6](#_Toc65416050)

[5.1. LOGIN FUNCTION 6](#_Toc65416051)

[5.2. EDIT TASK 6](#_Toc65416052)

[**6.** **SOURCE CODE** 7](#_Toc65416053)

[6.1. APP 7](#_Toc65416054)

[6.2. BACKGROUND APP 67](#_Toc65416055)

[**7.** **SAMPLE SCREENSHOTS** 71](#_Toc65416056)

[7.1. HOME SCREEN: 71](#_Toc65416057)

[7.1.1. SIGN – UP: 71](#_Toc65416058)

[7.1.2. LOGIN: 71](#_Toc65416059)

[7.1.3. CHANGE PASSWORD: 72](#_Toc65416060)

[7.2. MAIN SCREEN: 72](#_Toc65416061)

[7.2.1. ADD TASK: 72](#_Toc65416062)

[7.2.2. EDIT TASK: 73](#_Toc65416063)

[7.2.3. DELETE TASK: 73](#_Toc65416064)

[7.2.4. VIEW TASK: 73](#_Toc65416065)

[**8.** **LIMITATIONS OF THE PROJECT** 74](#_Toc65416066)

[8.1. SYSTEM DISK SPACE 74](#_Toc65416067)

[8.2. MAIL NOTIFICATION 74](#_Toc65416068)

[8.3. CLEANUP OF COMPLETED TASKS 74](#_Toc65416069)

[**9.** **SCOPE FOR IMPROVEMENT** 75](#_Toc65416070)

[9.1. REMOTE DATABASE 75](#_Toc65416071)

[9.2. HOSTED APPLICATION 75](#_Toc65416072)

[9.3. FEATURE ENHANCEMENTS 75](#_Toc65416073)

[**10.** **BIBLIOGRAPHY** 76](#_Toc65416074)

[10.1. DOCUMENTATION TKINTER: 76](#_Toc65416075)

[10.2. OFFICIAL PYTHON WEBSITE FOR Tcl/Tk: 76](#_Toc65416076)

[10.3. TTK WIDGETS: 76](#_Toc65416077)

## ACKNWOLEDGEMENT

I would like to thank my Computer Science teacher, Mrs. Kala and our school Principal Mrs. S.Vasanthy for giving me the opportunity to carry out this Computer Science Project. I would also like to thank the school for their constant guidance and support.

## CASE STUDY OF THE PROJECT

## PROJECT​:

Aimed to design and develop a software which keeps track of personal schedule and reminds the user (via Mail) at the given time.The functions that can be performed by this software are given below briefly.

## DETAILS​:

* Users will have access to the following functions:
  + To add a new task
  + To delete existing task
  + To edit any task
  + To view task
* This software will store the tasks that the user feeds into it.
* The reminder option is optional.
* If the reminder option is enabled, the user will be reminded via mail and there will also be apopup reminder.
* In the add task option the user can select the start date and time and end date and timeaccording to their will.
* There will also be a setting that allows the task to be recurring. [For example, setting areminder for Wednesdays alone to take a lab coat for practical session at 7:00 a.m]
* The popup screen reminder will have a dismiss button and a snooze button which ifsnoozed will send another popup in the required time.(mail will be sent only once. Whensnoozed, mail will not be sent the second time.)
* The user will also be able to delete certain tasks.
* The user can view the tasks done or pending in a week wise fashion.

## DESIGN IDEA​:

MENU: (This may not be the menu that is displayed. But all the functions will be covered)

* Add task
* Remove task
* Edit task
* View tasks

## MODULES USED:

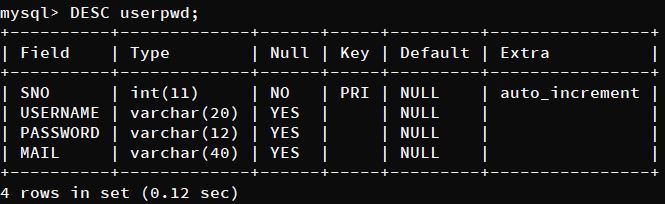
The modules used in this project are:

* tkinter(package)
* tkinter.filedialog
* tkinter.messagebox
* tkinter.ttk
* tkcalendar
* mysql.connector
* datetime
* time
* calendar
* smtplib
* plyer.notification

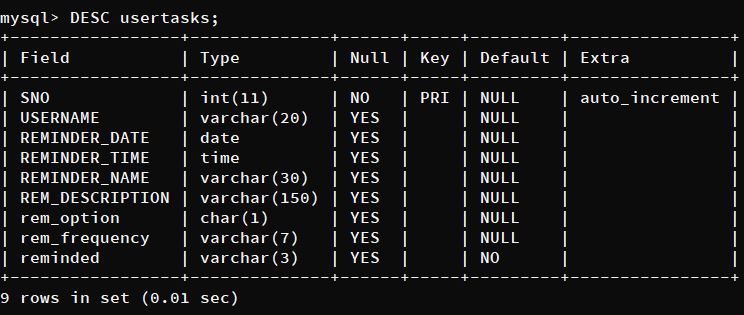
## SCREENSHOTS OF TABLES OR DESCRIPTIONS OF TABLES

Two tables have been user for this project, namely:

1. USERPWD – Contains the username, password and gmail id of the user

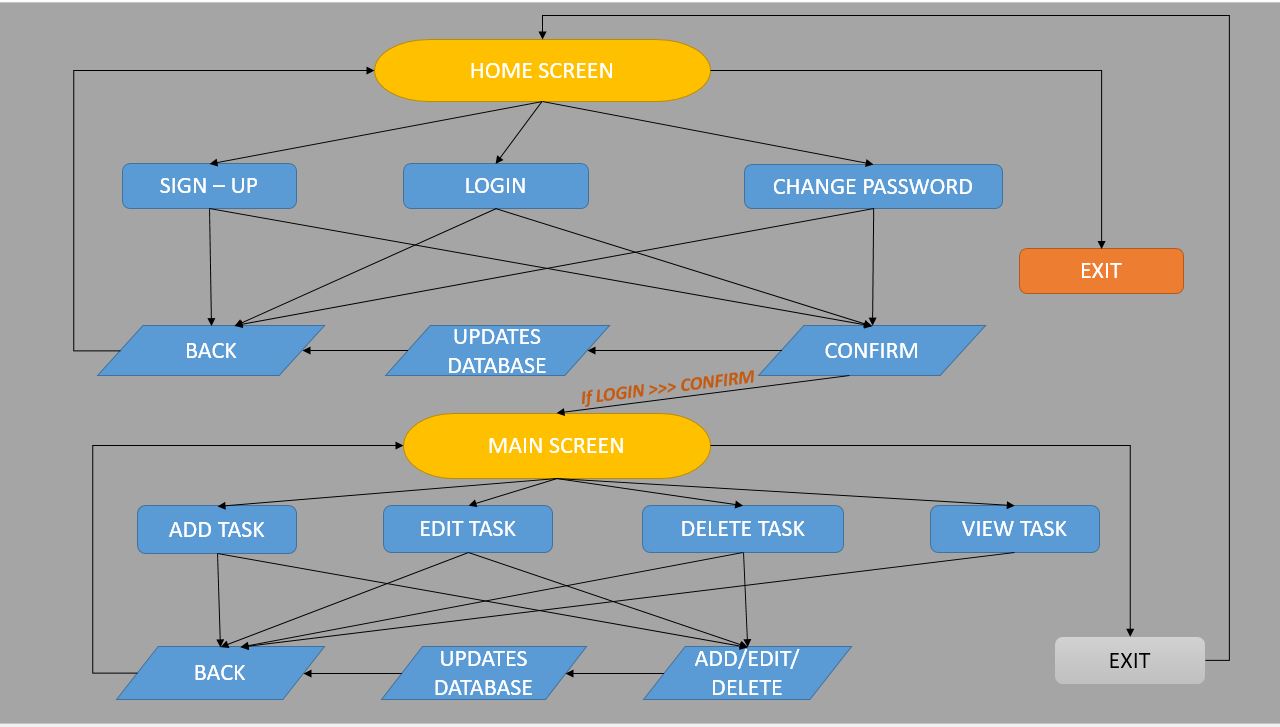


1. USERTASKS – Contains the details regarding tasks input by the user.



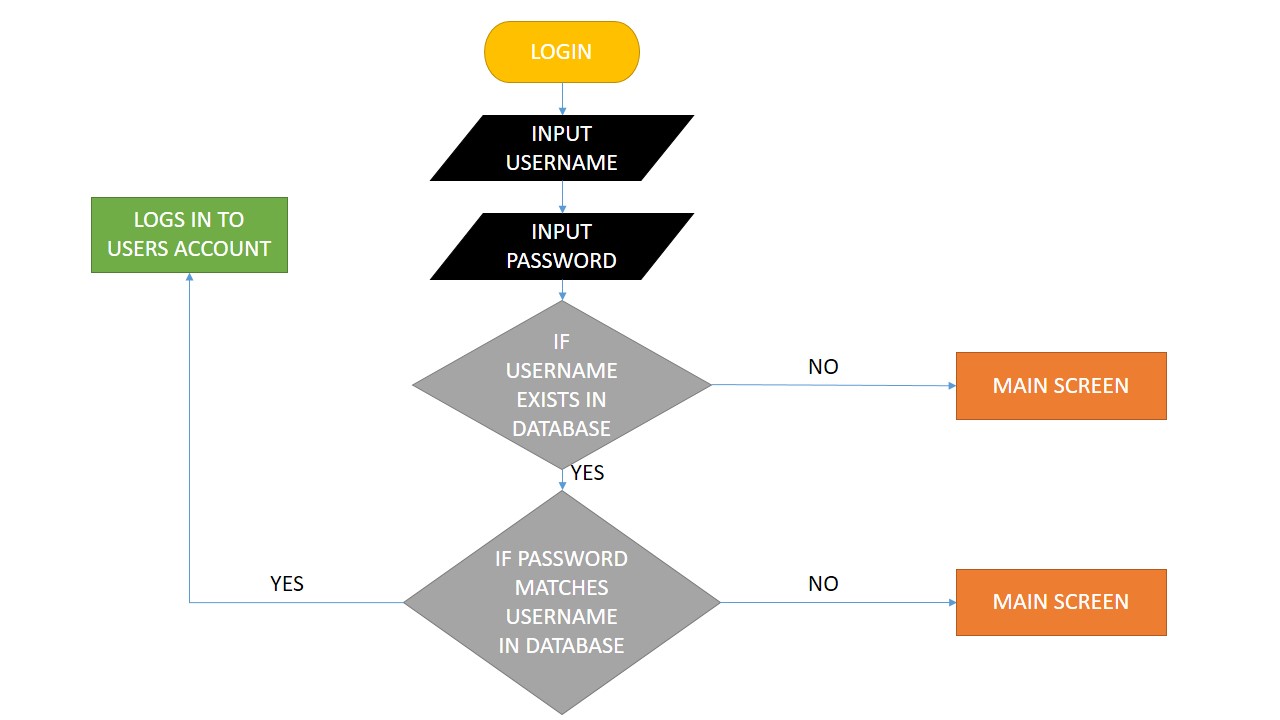
These 2 tables have been put under a database ‘TODOLIST’.

## PROCESS DIAGRAM

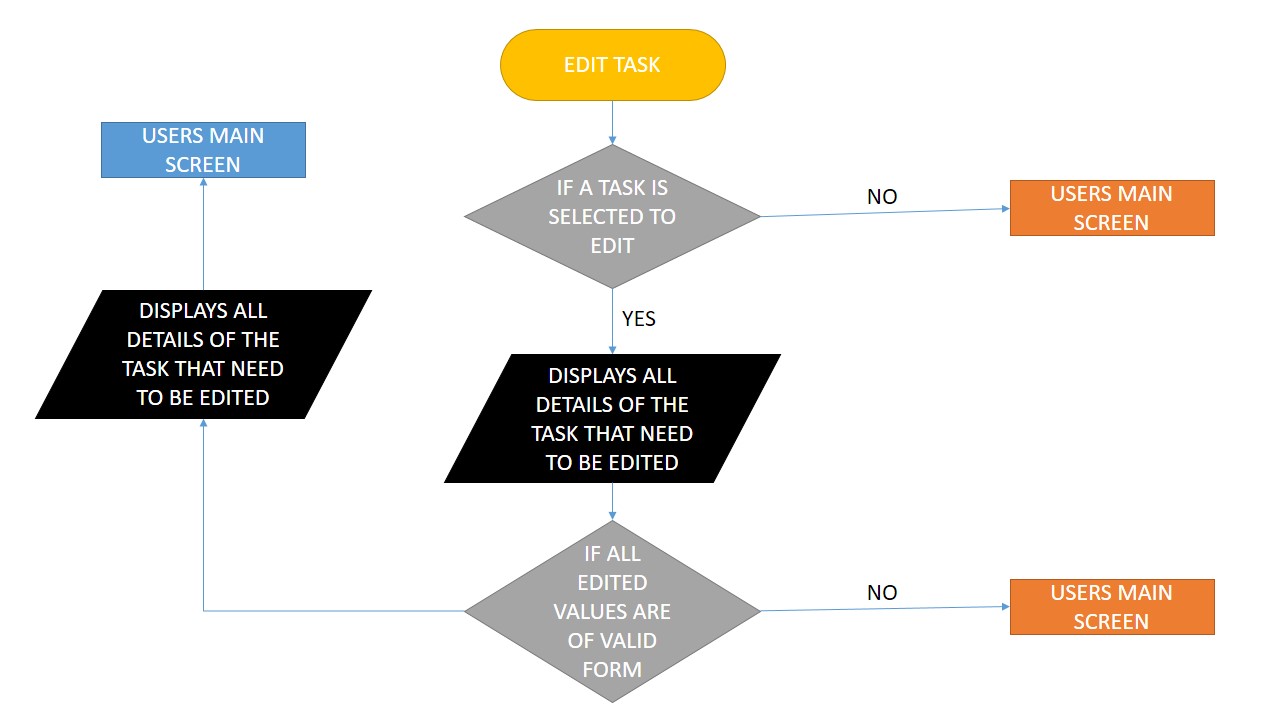


## FLOW CHARTS

## LOGIN FUNCTION



## EDIT TASK



## SOURCE CODE

## APP

# IMPORTING REQUIRED MODULES/LIBRARIES #

import tkinter as tk # FOR THE GRAPHICAL USER INTERFACE

import mysql.connector as my # TO CONNECT TO THE MYSQL DATABASE WHERE THE REQUIRED TABLES ARE STORED

from tkinter import messagebox # TO IMPORT MESSAGEBOX FROM TKINTER TO DISPLAY MESSAGES

import tkcalendar as tkc # TO IMPORT THE CALENDAR WIDGET

import datetime as dt # TO WORK WITH CURRENT DATES AND TIMES AND TIME DELTAS

from tkinter import ttk # TO MODIFY APPEARANCE OF CERTAIN WIDGETS

# GLOABLISING THE REQUIRED VARIABLES

global success

global user

success = False

user = False

# HOME SCREEN FUNCTION #

def homepage():

global success

success = False

# CREATING THE 'Tk' WINDOW WHERE ANY OUTPUT IS DISPLAYED

p\_win = tk.Tk("zoomed")

p\_win.title(

"TO-DO-LIST!!"

) # SETTING THE TITLE OF THE WINDOW TO THE NAME OF THE APPLICATION

p\_win.state("zoomed") # MAKING THE WINDOW GET DISPLAYED IN A MAXIMISED STATE

p\_win.iconbitmap(

"IMAGES\_SUBMISSION\\icon.ico"

) # SETTING ICON OF THE APP WINDOW

bg\_img = tk.PhotoImage(

file="IMAGES\_SUBMISSION\\scenery.png"

) # INITIALISING IMAGE

bg\_label = tk.Label(p\_win, image=bg\_img) # LABEL WITH THE BACKGROUND IMAGE

bg\_label.place(x=0, y=0) # PLACING THE LABEL

global hide\_img

global show\_img

hide\_img = tk.PhotoImage(

file="IMAGES\_SUBMISSION\\hide\_pwd.png"

) # INITIALISING IMAGE

show\_img = tk.PhotoImage(

file="IMAGES\_SUBMISSION\\show\_pwd.png"

) # INITIALISING IMAGE

# DEFINING SIGNIN AND LOGIN FUNCTIONS AND CHANGE PASSWORD FUNCTION #

# SIGN UP FUNCTION #

def signup():

# DISABLING THE SIGNUP, LOGIN AND CHANGE PASSWORD FUNCTION SO THAT USER CANNOT GO TO OTHER BUTTONS WHEN SIGNIN IS ACTIVE

signup\_btn.config(state="disabled")

login\_btn.config(state="disabled")

pwdcng\_btn.config(state="disabled")

# SIGNUP, USERNAME, PASSWORD AND EMAIL LABELS

signup = tk.Label(

p\_win,

text=" SIGN - UP ",

font=("Fugaz One", 26),

fg="black",

padx=15,

pady=10,

)

usn = tk.Label(

p\_win,

text="USERNAME :",

font=("Fugaz One", 18),

fg="black",

padx=15,

pady=10,

)

pwd = tk.Label(

p\_win,

text="PASSWORD :",

font=("Fugaz One", 18),

fg="black",

padx=14,

pady=10,

)

mail = tk.Label(

p\_win,

text="E-MAIL ID :",

font=("Fugaz One", 18),

fg="black",

padx=16,

pady=10,

)

# ENTRY BOXES FOR THE USER TO ENTER USERNAME PASSWORD AND MAIL

e\_usn = tk.Entry(p\_win, width=18, font=("Fugaz One", 18))

e\_mail = tk.Entry(p\_win, width=18, font=("Fugaz One", 18))

e\_pwd = tk.Entry(p\_win, width=16, show="\*", font=("Fugaz One", 18))

e\_usn.focus()

# FUNCTION TO SHOW THE PASSWORD WHEN USER CLICKS ON THE SHOW PASSWORD BUTTON #

def hide\_signup():

pwd\_btn.config(image=hide\_img, command=show\_signup)

e\_pwd.config(show="\*")

# FUNCTION TO HIDE THE PASSWORD WHEN THE USER CLICKS ON THE HIDE PASSWORD BUTTON #

def show\_signup():

pwd\_btn.config(image=show\_img, command=hide\_signup)

e\_pwd.config(show="")

# FUNCTION FOR BACK BUTTON #

def back\_signup():

# TO ENABLE THE SIGNIN, LOGIN AND CHANGE PASSWORD BUTTONS THAT WERE DISABLED BEFORE

signup\_btn.config(state="normal")

login\_btn.config(state="normal")

pwdcng\_btn.config(state="normal")

# TO MAKE THE NEW WIDGETS THAT SHOWED UP WHEN SIGNUP WAS CLICKED TO DIASAPPEAR

signup.destroy()

usn.destroy()

pwd.destroy()

mail.destroy()

e\_usn.destroy()

e\_pwd.destroy()

e\_mail.destroy()

pwd\_btn.destroy()

confirm\_btn.destroy()

back\_btn.destroy()

# FUNCTION FOR CONFIRM BUTTON #

def confirm\_signup():

# TO GET WHAT TEXT IS CURRENTLY PRESENT IN THE ENTRY BOXES

username = e\_usn.get()

password = e\_pwd.get()

email = e\_mail.get()

# TO CLEAR WHAT TEXT IS CURRENTLY PRESENT IN THE ENTRY BOXES

e\_usn.delete(0, "end")

e\_pwd.delete(0, "end")

e\_mail.delete(0, "end")

# CONNECTING TO MySQL TO GET THE DATA IN THE TABLE

c = my.connect(

host="localhost", user="root", passwd="MYSQLVARY", database="todolist"

)

cur = c.cursor()

# TO GET DATA FROM MySQL DATABASE

cur.execute("select \* from userpwd")

data = cur.fetchall()

c.close()

# SETTING A FLAG IN WHICH USER IS TRUE

user = True

for x in data:

# IF CONDITION TO CHECK IF THE USERNAME ALREADY EXISTS AND SETS USER FLAG TO FALSE

if x[1] == username:

user = False

messagebox.showinfo(

"INVALID USERNAME", "SORRY! THIS USERNAME HAS BEEN TAKEN"

)

break

if email.endswith("@gmail.com") == False:

user = False

messagebox.showinfo(

"INVALID MAIL ID",

"MAIL ID IS SUPPOSED TO END WITH '@gmail.com'.",

)

for x in data:

# IF CONDITION TO CHECK IF MAIL HAS ALREADY BEEN USED BY SOME OTHER USER AND SETS USER FLAG TO FALSE

if x[3] == email:

user = False

messagebox.showinfo(

"WRONG MAIL",

"OOPS! AN ACCOUNT ALREADY EXISTS WITH THIS MAIL ID.",

)

break

# IF THE USER FLAG IS TRUE AFTER PASSING THROUGH THE IF CONDITIONS

if user == True:

# CONNECTS TO MySQL DATABASE AND INSERTS THE USERNAME, PASSWORD AND MAIL ENTERED BY USER

c = my.connect(

host="localhost",

user="root",

passwd="MYSQLVARY",

database="todolist",

)

cur = c.cursor()

# TO INSERT THE DATA ADDED BY THE USER

cur.execute(

"insert into userpwd (username, password, mail) values("

+ "'"

+ str(username)

+ "','"

+ str(password)

+ "','"

+ str(email)

+ "')"

)

c.commit()

c.close()

messagebox.showinfo(

"SIGN-UP SUCCESSFUL",

"YOU HAVE COMPLETED SIGNING UP. LOGIN TO GET STARTED",

)

else:

pass

# CALLS THE BELOW FUNCTION TO FO BACK TO THE DEFAULT MAIN MENU AFTER SIGNING UP

back\_signup()

# PASSWORD HIDE OR SHOW BUTTON, CONFIRM BUTTON AND BACK BUTTON

pwd\_btn = tk.Button(image=hide\_img, command=show\_signup)

confirm\_btn = tk.Button(

p\_win,

text="CONFIRM",

font=("Fugaz One", 18),

fg="black",

padx=16,

command=confirm\_signup,

)

back\_btn = tk.Button(

p\_win,

text=" BACK ",

font=("Fugaz One", 18),

fg="black",

padx=15,

command=back\_signup,

)

# PLACING ALL THE SIGN UP WIDGETS #

signup.place(x=653, y=136)

usn.place(x=503, y=236)

pwd.place(x=503, y=381)

mail.place(x=503, y=311)

e\_usn.place(x=701, y=241)

e\_pwd.place(x=701, y=386)

pwd\_btn.place(x=952, y=386)

e\_mail.place(x=701, y=316)

confirm\_btn.place(x=588, y=461)

back\_btn.place(x=768, y=461)

# LOGIN FUNCTION #

def login():

# DISABLING THE SIGNUP, LOGIN AND CHANGE PASSWORD FUNCTION SO THAT USER CANNOT GO TO OTHER BUTTONS WHEN SIGNIN IS ACTIVE

signup\_btn.config(state="disabled")

login\_btn.config(state="disabled")

pwdcng\_btn.config(state="disabled")

# LOGIN, USERNAME AND PASSWORD LABELS

login = tk.Label(

p\_win,

text=" LOGIN ",

font=("Fugaz One", 26),

fg="black",

padx=15,

pady=10,

)

usn = tk.Label(

p\_win,

text="USERNAME :",

font=("Fugaz One", 18),

fg="black",

padx=15,

pady=10,

)

pwd = tk.Label(

p\_win,

text="PASSWORD :",

font=("Fugaz One", 18),

fg="black",

padx=14,

pady=10,

)

# USERNAME AND PASSWORD ENTRY BOXES IN WHICH USER ENTERS TEXT

e\_usn = tk.Entry(p\_win, width=18, font=("Fugaz One", 18))

e\_pwd = tk.Entry(p\_win, width=16, show="\*", font=("Fugaz One", 18))

e\_usn.focus()

# FUNCTION TO HIDE THE PASSWORD #

def hide\_login():

pwd\_btn.config(image=hide\_img, command=show\_login)

e\_pwd.config(show="\*")

# FUNCTION TO SHOW THE PASSWORD #

def show\_login():

pwd\_btn.config(image=show\_img, command=hide\_login)

e\_pwd.config(show="")

# FUNCTION FOR BACK BUTTON #

def back\_login():

# TO ENABLE THE SIGNIN, LOGIN AND CHANGE PASSWORD BUTTONS THAT WERE DISABLED BEFORE

signup\_btn.config(state="normal")

login\_btn.config(state="normal")

pwdcng\_btn.config(state="normal")

# TO MAKE THE NEW WIDGETS THAT SHOWED UP WHEN LOGIN WAS CLICKED TO DIASAPPEAR

login.destroy()

usn.destroy()

pwd.destroy()

e\_usn.destroy()

e\_pwd.destroy()

pwd\_btn.destroy()

confirm\_btn.destroy()

back\_btn.destroy()

# FUNCTION FOR CONFIRM BUTTON #

def confirm\_login():

global success

global user

# TO GET THE USERNAME AND PASSWORD ENTERED BY THE USER

username = e\_usn.get()

password = e\_pwd.get()

# TO CLEAR THE USERNAME AND PASSWORD THE USER ENTERED FROM THE ENTRY BOX

e\_usn.delete(0, "end")

e\_pwd.delete(0, "end")

# CONNECTING TO MySQL DATABASE

c = my.connect(

host="localhost", user="root", passwd="MYSQLVARY", database="todolist"

)

cur = c.cursor()

# TO GET DATA FROM MySQL DATABASE

cur.execute("select \* from userpwd")

data = cur.fetchall()

c.close()

# CREATING A LIST WITH ALL USERNAMES

usernames = []

for x in data:

usernames.append(x[1])

# SETTING A FLAG THAT SAYS USER = TRUE

user = True

# IF CONDITION TO CHECK IF THE USERNAME ENTERED BY THE USER IS THERE IN THE DATABASE, IF NOT SETS USER FLAG TO FALSE

if username not in usernames:

user = False

messagebox.showinfo(

"INVALID USERNAME",

"THE USERNAME DOESNT EXIST. PLEASE SIGNUP AND LOGIN.",

)

if username in usernames:

for x in data:

# IF CONDITION TO CHECK IF PASSWORD IS RIGHT FOR THE CORRESPONDING USERNAME, IF NOT SETS USER FLAG TO FALSE

if x[1] == username and x[2] != password:

user = False

messagebox.showinfo(

"OOPS!!!", "YOU HAVE ENTERED THE WRONG PASSWORD."

)

break

for x in data:

# IF USERNAME AND PASSWORD ARE RIGHT, SETS USER FLAG TO TRUE AND THE GLOBAL VARIABLE SUCCESS TO TRUE

if x[1] == username and x[2] == password:

user = True

success = True

user = username

messagebox.showinfo(

"LOGIN SUCCESFUL", "YOU HAVE SUCCESSFULLY LOGGED IN!!!"

)

p\_win.destroy()

break

# SHOW AND HIDE PASSWORD BUTTONS, CONFIRM AND BACK BUTTONS

pwd\_btn = tk.Button(image=hide\_img, command=show\_login)

confirm\_btn = tk.Button(

p\_win,

text="CONFIRM",

font=("Fugaz One", 18),

fg="black",

padx=16,

command=confirm\_login,

)

back\_btn = tk.Button(

p\_win,

text=" BACK ",

font=("Fugaz One", 18),

fg="black",

padx=15,

command=back\_login,

)

# PLACING ALL THE LOGIN WIDGETS #

login.place(x=653, y=136)

usn.place(x=503, y=236)

pwd.place(x=503, y=311)

e\_usn.place(x=701, y=241)

e\_pwd.place(x=701, y=316)

pwd\_btn.place(x=952, y=316)

confirm\_btn.place(x=693, y=386)

back\_btn.place(x=693, y=461)

# CHANGE PASSWORD FUNCTION #

def pwdcng():

# DISABLING THE SIGNUP, LOGIN AND CHANGE PASSWORD FUNCTION SO THAT USER CANNOT GO TO OTHER BUTTONS WHEN SIGNIN IS ACTIVE

signup\_btn.config(state="disabled")

login\_btn.config(state="disabled")

pwdcng\_btn.config(state="disabled")

# CHANGE PASSWORD, USERNAME, CURRENT PASSWORD, NEW PASSWORD LABELS

pwdcng\_label = tk.Label(

p\_win,

text=" CHANGE PASSWORD ",

font=("Fugaz One", 26),

fg="black",

padx=15,

pady=10,

)

usn = tk.Label(

p\_win,

text="USERNAME :",

font=("Fugaz One", 18),

fg="black",

padx=15,

pady=10,

)

c\_pwd = tk.Label(

p\_win,

text="CURRENT PASSWORD :",

font=("Fugaz One", 18),

fg="black",

padx=14,

pady=10,

)

n\_pwd = tk.Label(

p\_win,

text="NEW PASSWORD :",

font=("Fugaz One", 18),

fg="black",

padx=16,

pady=10,

)

# ENTRY BOXES FOR USERNAME, CURRENT PASSWORD AND NEW PASSWORD

e\_usn = tk.Entry(p\_win, width=18, font=("Fugaz One", 18))

e\_pwd = tk.Entry(p\_win, width=18, show="", font=("Fugaz One", 18))

e\_npwd = tk.Entry(p\_win, width=16, show="\*", font=("Fugaz One", 18))

e\_usn.focus()

# FUNCTION TO HIDE THE PASSWORD #

def hide\_pwdcng():

pwd\_btn.config(image=hide\_img, command=show\_pwdcng)

e\_npwd.config(show="\*")

# FUNCTION TO SHOW THE PASSWORD #

def show\_pwdcng():

pwd\_btn.config(image=show\_img, command=hide\_pwdcng)

e\_npwd.config(show="")

# FUNCTION FOR BACK BUTTON #

def back\_pwdcng():

# TO ENABLE THE SIGNIN, LOGIN AND CHANGE PASSWORD BUTTONS THAT WERE DISABLED BEFORE

signup\_btn.config(state="normal")

login\_btn.config(state="normal")

pwdcng\_btn.config(state="normal")

# TO MAKE THE NEW WIDGETS THAT SHOWED UP WHEN CHANGE PASSWORD WAS CLICKED TO DIASAPPEAR

pwdcng\_label.destroy()

usn.destroy()

c\_pwd.destroy()

n\_pwd.destroy()

e\_usn.destroy()

e\_pwd.destroy()

e\_npwd.destroy()

pwd\_btn.destroy()

confirm\_btn.destroy()

back\_btn.destroy()

# FUNCTION FOR CONFIRM BUTTON #

def confirm\_pwdcng():

# TO GET THE USERNAME, CURRENT PASSWORD AND NEW PASSWORD ENTERED BY THE USER

username = e\_usn.get()

password = e\_pwd.get()

n\_password = e\_npwd.get()

# TO CLEAR THE USERNAME, CURRENT PASSWORD AND NEW PASSWORD ENTERED BY THE USER

e\_usn.delete(0, "end")

e\_pwd.delete(0, "end")

e\_npwd.delete(0, "end")

# TO CONNECT TO MySQL DATABASE

c = my.connect(

host="localhost", user="root", passwd="MYSQLVARY", database="todolist"

)

cur = c.cursor()

# TO GET DATA FROM MySQL DATABASE

cur.execute("select \* from userpwd")

data = cur.fetchall()

c.close()

# LISTS WITH ALL USERNAMES AND EMAILS

usernames = []

emails = []

for x in data:

usernames.append(x[1])

emails.append(x[3])

# SETTING FLAG USER = TRUE

user = True

# IF USERNAME IS NOT PRESENT IN LIST 'usernames', FLAG IS SET TO FALSE

if username not in usernames:

user = False

messagebox.showinfo(

"INVALID USERNAME",

"THE USERNAME DOESNT EXIST. PLEASE SIGNUP FIRST.",

)

if username in usernames:

for x in data:

# IF CURRENT PASSWORDS DONT MATCH, FLAG IS SET TO FALSE

if x[1] == username and x[2] != password:

user = False

messagebox.showinfo(

"OOPS!!!", "YOU HAVE ENTERED THE WRONG PASSWORD."

)

break

for x in data:

# IF USERNAME AND CURRENT PASSWORDS MATCH, THEN FLAG IS SET TO TRUE

if x[1] == username and x[2] == password:

user = True

# CONNECTING TO MySQL DATABASE

c = my.connect(

host="localhost",

user="root",

passwd="MYSQLVARY",

database="todolist",

)

cur = c.cursor()

# TO DELETE THE PREVIOUS ENTRY FROM MySQL DATABASE

cur.execute(

"delete from userpwd where username='" + str(username) + "'"

)

# INSERTING THE SAME ENTRY WITH NEW PASSWORD REPLACING THE OLD PASSWORD

cur.execute(

"insert into userpwd(username, password, mail) values('"

+ str(username)

+ "','"

+ str(n\_password)

+ "','"

+ str(x[2])

+ "')"

)

c.commit()

c.close()

messagebox.showinfo(

"PASSWORD CHANGE SUCCESFUL",

"YOU HAVE CHANGED YOUR PASSWORD!!!",

)

break

# CALLS BELOW FUNCTION TO MAKE ALL CHANGE PASSWORD WIDGETS DISAPPEAR

back\_pwdcng()

# SHOW AND HIDE PASSWORD BUTTONS, CONFIRM AND BACK BUTTONS

pwd\_btn = tk.Button(image=hide\_img, command=show\_pwdcng)

confirm\_btn = tk.Button(

p\_win,

text="CONFIRM",

font=("Fugaz One", 18),

fg="black",

padx=16,

command=confirm\_pwdcng,

)

back\_btn = tk.Button(

p\_win,

text=" BACK ",

font=("Fugaz One", 18),

fg="black",

padx=15,

command=back\_pwdcng,

)

# PLACING ALL THE WIDGETS IN CHANGE PASSWORD #

pwdcng\_label.place(x=533, y=136)

usn.place(x=448, y=236)

e\_usn.place(x=766, y=241)

c\_pwd.place(x=448, y=311)

e\_pwd.place(x=766, y=316)

n\_pwd.place(x=448, y=386)

e\_npwd.place(x=766, y=391)

pwd\_btn.place(x=1018, y=391)

confirm\_btn.place(x=588, y=461)

back\_btn.place(x=768, y=461)

# DISPLAYING HOME SCREEN #

# HOME SCREEN TITLE #

title\_label = tk.Label(

text=" WELCOME TO YOUR PERSONAL TASK REMINDER ! ",

font=("Fugaz One", 30),

padx=25,

pady=25,

fg="yellow",

bg="black",

)

# HOME SCREEN SIGIN BUTTON #

signup\_btn = tk.Button(

text=" SIGN UP ",

font=("Fugaz One", 18),

border=10,

fg="white",

bg="black",

command=signup,

)

# HOME SCREEN LOGIN BUTTON #

login\_btn = tk.Button(

text=" LOGIN ",

font=("Fugaz One", 18),

border=10,

fg="white",

bg="black",

command=login,

)

# HOME SCREEN CHANGE PASSWORD BUTTON #

pwdcng\_btn = tk.Button(

p\_win,

text="CHANGE PASSWORD",

font=("Fugaz One", 18),

border=10,

fg="white",

bg="black",

command=pwdcng,

)

# HOME SCREEN EXIT BUTTON #

exit\_btn = tk.Button(

p\_win,

text=" EXIT ",

font=("Fugaz One", 18),

border=10,

fg="red",

bg="black",

command=p\_win.destroy,

)

# PLACING ALL WIDGETS IN HOME SCREEN #

title\_label.place(x=213, y=6)

signup\_btn.place(x=503, y=538)

login\_btn.place(x=763, y=538)

pwdcng\_btn.place(x=633, y=626)

exit\_btn.place(x=633, y=716)

p\_win.mainloop() # FOR THE WINDOW TO RUN TILL EXIT BUTTON OR THE RED CROSS IN THE TOP RIGHT IS CLICKED

# FUNCTION TO CREATE A NEW WINDOW #

def new\_window(text):

# CREATING NEW TKINTER WINDOW FOR THE VARIOUS FUNCTIONS

n\_win = tk.Tk()

n\_win.state("zoomed") # MAKING THE WINDOW GET DISPLAYED IN A MAXIMISED STATE

n\_win.title(

"Logged Into To Do List!!"

) # SETTING THE TITLE OF THE WINDOW TO THE NAME OF THE APPLICATION

n\_win.iconbitmap(

"IMAGES\_SUBMISSION\\icon.ico"

) # SETTING ICON OF THE APP WINDOW

n\_win\_bg\_img = tk.PhotoImage(

file="IMAGES\_SUBMISSION\\scenery2.png"

) # BACKGROUND IMAGE FOR THE NEW WINDOW

# LABEL THAT CONTAINS THE BACKGROUND IMAGE AND THE WELCOME MESSAGE

n\_win\_bg\_img\_label = tk.Label(

n\_win,

image=n\_win\_bg\_img,

text=" Welcome " + text + " \n\n\n\n\n\n\n\n\n\n\n\n\n",

fg="yellow",

compound="center",

font=("Fugaz One", 30),

)

n\_win\_bg\_img\_label.place(x=0, y=0)

# FUNCTION TO ADD TASK #

def add\_task():

# TO DISABLE THE OTHER FUNCTIONS WHEN ADD TASK BUTTON IS CLICKED AND ADD TASK FUNCTION IS BEING PERFROMED

add\_task\_btn.config(state="disabled")

edit\_task\_btn.config(state="disabled")

delete\_task\_btn.config(state="disabled")

view\_task\_btn.config(state="disabled")

back\_btn.config(state="disabled")

# WIDGETS REQUIRED TO PERFORM ALL FUNCTIONS IN ADD TASK

rem\_name\_label = tk.Label(

n\_win,

text="NAME OF THE REMINDER : ",

fg="white",

bg="black",

font=("Fugaz One", 18),

)

e\_rem\_name = tk.Entry(width=30, font=("Fugaz One", 18))

e\_rem\_name.focus()

rem\_name\_label.place(x=386, y=225)

e\_rem\_name.place(x=706, y=225)

c\_datentime = dt.datetime.now()

c\_date = c\_datentime.date()

c\_year = int(str(c\_date)[0:4])

c\_month = int(str(c\_date)[5:7])

c\_date = int(str(c\_date)[8:10])

c\_hour = int(str(c\_datentime.time())[0:2])

c\_minute = int(str(c\_datentime.time())[3:5])

# SELECTING THE CURRENT DAY'S DATE ON THE CALENDAR BY DEFAULT

cal = tkc.Calendar(

n\_win,

selectmode="day",

year=c\_year,

month=c\_month,

day=c\_date,

pady=10,

fill="BOTH",

)

cal.place(x=458, y=275)

reminder\_option\_label = tk.Label(

n\_win, text="REMIND ME? : ", font=("Fugaz One", 18), fg="white", bg="black"

)

reminder\_option\_label.place(x=730, y=275)

rem\_option = tk.StringVar()

yes\_rem\_rb = tk.Radiobutton(n\_win, text="YES", variable=rem\_option, value="Y")

no\_rem\_rb = tk.Radiobutton(n\_win, text="NO", variable=rem\_option, value="N")

yes\_rem\_rb.place(x=930, y=280)

no\_rem\_rb.place(x=930, y=315)

no\_rem\_rb.select() # TO HAVE 'No' SELECTED BY DEFAULT

frequency\_label = tk.Label(

n\_win,

text="FREQUENCY OF REMINDERS : ",

fg="white",

bg="black",

font=("Fugaz One", 18),

)

frequency\_label.place(x=730, y=380)

rec\_options = ["ONCE", "DAILY", "WEEKLY", "MONTHLY"]

recurring\_freq\_var = tk.StringVar()

recurring\_freq\_var.set(

rec\_options[0]

) # TO SET 'ONCE' AS THE FREQUENCY BY DEFAULT

rec\_or\_not\_dropdown = tk.OptionMenu(n\_win, recurring\_freq\_var, \*rec\_options)

rec\_or\_not\_dropdown.place(x=1088, y=385)

hour\_label = tk.Label(

n\_win, text="HOUR : ", font=("Fugaz One", 18), fg="white", bg="black"

)

hour\_label.place(x=400, y=500)

hours = []

# LOOP TO GET THE STRING OF ALL HOURS AS A 2 CHARACTERS

for i in range(0, 12):

if i < 10:

i = "0" + str(i)

hours.append(i)

else:

i = str(i)

hours.append(i)

hour\_var = tk.StringVar()

am\_pm\_dropdown\_set = "a.m"

if c\_hour > 11:

c\_hour -= 12

am\_pm\_dropdown\_set = "p.m"

# TO KEEP THE CURRENT TIME SELECTED BY DEFAULT

if len(str(c\_hour)) == 1:

c\_hour = "0" + str(c\_hour)

else:

c\_hour = str(c\_hour)

for i in hours:

if i == c\_hour:

hour\_var.set(i)

hour\_dropdown = tk.OptionMenu(n\_win, hour\_var, \*hours)

hour\_dropdown.place(x=500, y=505)

minute\_label = tk.Label(

n\_win, text="MINUTE : ", font=("Fugaz One", 18), fg="white", bg="black"

)

minute\_label.place(x=690, y=500)

minutes = []

for i in range(0, 60):

if i < 10:

i = "0" + str(i)

minutes.append(i)

else:

i = str(i)

minutes.append(i)

minute\_var = tk.StringVar()

if len(str(c\_minute)) == 1:

c\_minute = "0" + str(c\_minute)

else:

c\_minute = str(c\_minute)

for i in minutes:

if i == c\_minute:

minute\_var.set(i)

minute\_dropdown = tk.OptionMenu(n\_win, minute\_var, \*minutes)

minute\_dropdown.place(x=820, y=505)

am\_pm\_label = tk.Label(

n\_win, text="A.M / P.M : ", font=("Fugaz One", 18), fg="white", bg="black"

)

am\_pm\_label.place(x=980, y=500)

am\_pm\_list = ["a.m", "p.m"]

am\_pm\_var = tk.StringVar()

if am\_pm\_dropdown\_set == "a.m":

am\_pm\_var.set(am\_pm\_list[0])

elif am\_pm\_dropdown\_set == "p.m":

am\_pm\_var.set(am\_pm\_list[1])

am\_pm\_dropdown = tk.OptionMenu(n\_win, am\_pm\_var, \*am\_pm\_list)

am\_pm\_dropdown.place(x=1130, y=505)

rem\_desc\_label = tk.Label(

n\_win,

text="REMINDER DESCRIPTION : ",

font=("Fugaz One", 18),

fg="white",

bg="black",

)

rem\_desc\_label.place(x=386, y=550)

rem\_desc\_text = tk.Text(n\_win, font=("Fugaz One", 18), height=5, width=30)

rem\_desc\_text.place(x=706, y=550)

# FUNCTION OF BACK BUTTON INSIDE ADD TASK #

def back\_at\_fn():

# TO ENABLE THE DISABLED WIDGETS

add\_task\_btn.config(state="normal")

edit\_task\_btn.config(state="normal")

delete\_task\_btn.config(state="normal")

view\_task\_btn.config(state="normal")

back\_btn.config(state="normal")

# TO DESTROY ALL ADD TASK WIDGETS THAT WERE ADDED

rem\_name\_label.destroy()

e\_rem\_name.destroy()

cal.destroy()

hour\_label.destroy()

hour\_dropdown.destroy()

minute\_label.destroy()

minute\_dropdown.destroy()

am\_pm\_label.destroy()

am\_pm\_dropdown.destroy()

rem\_desc\_label.destroy()

rem\_desc\_text.destroy()

add\_btn.destroy()

back\_at\_btn.destroy()

reminder\_option\_label.destroy()

yes\_rem\_rb.destroy()

no\_rem\_rb.destroy()

frequency\_label.destroy()

rec\_or\_not\_dropdown.destroy()

# FUNCTION OF ADD BUTTON INSIDE ADD TASK #

def add\_btn\_fn():

# COLLECTING THE DATA ENTERED BY THE USER AS SOON AS ADD TASK BUTTON IS CLICKED

reminder\_option = rem\_option.get()

reminder\_frequency = recurring\_freq\_var.get()

reminder\_name = e\_rem\_name.get()

reminder\_desc = rem\_desc\_text.get(1.0, "end")

date\_raw = str(cal.get\_date())

ymd = date\_raw.split("/")

year = "20" + ymd[2]

if int(ymd[0]) > 9:

month = ymd[0]

else:

month = "0" + ymd[0]

if int(ymd[1]) > 9:

day = ymd[1]

else:

day = "0" + ymd[1]

date = year + "-" + month + "-" + day

hour = hour\_var.get()

minute = minute\_var.get()

am\_pm = am\_pm\_var.get()

if am\_pm == "p.m":

hour = str(int(hour) + 12)

time = str(hour) + ":" + str(minute) + ":" + "00"

# SETTING A FLAG ADD TASK TO TABLE = TRUE

add\_task\_table = True

# IF NAME OF THE REMINDER IS NULL, FLAG BECOMES FALSE

if len(reminder\_name) == 0:

messagebox.showinfo(

"INVALID NAME OF REMINDER",

"NOTE : NAME OF THE REMINDER CANNOT BE NULL. ",

)

add\_task\_table = False

# IF NAME OF THE REMINDER IS MORE THAN 30 CHARACTERS, FLAG BECOMES FALSE

if len(reminder\_name) > 30:

messagebox.showinfo(

"INVALID NAME OF REMINDER",

"NOTE : NAME OF THE REMINDER HAS EXCEEDED 30 CHARACTERS.",

)

add\_task\_table = False

# IF REMINDER DESCRIPTION IS NULL, FLAG BECOMES FALSE

if len(reminder\_desc) == 0:

messagebox.showinfo(

"INVALID DESCEIPTION OF REMINDER",

"NOTE : DESCRIPTION OF REMINDER CANNOT BE NULL.",

)

add\_task\_table = False

# IF REMINDER DESCRIPTION IS MORE THAN 150 CHARACTERS, FLAG BECOMES FALSE

if len(reminder\_desc) > 150:

messagebox.showinfo(

"INVALID DESCEIPTION OF REMINDER",

"NOTE : DESRIPTION OF REMINDER HAS REACHED 150 CHARACTERS",

)

add\_task\_table = False

# AT THE END OF ALL REQUIRED IF CONDITIONS, IF FLAG IS TRUE TASK IS ADDED TO THE DATABASE

if add\_task\_table == True:

# CONNECTING TO MySQL DATABASE

c = my.connect(

host="localhost",

user="root",

passwd="MYSQLVARY",

database="todolist",

)

cur = c.cursor()

# TO INSERT THE TASK THAT THE USER WANTS TO ADD

cur.execute(

"INSERT INTO usertasks(username, reminder\_date, reminder\_time, reminder\_name, rem\_description, rem\_option, rem\_frequency) VALUES('"

+ text

+ "','"

+ date

+ "','"

+ time

+ "','"

+ reminder\_name

+ "','"

+ reminder\_desc

+ "','"

+ reminder\_option

+ "','"

+ reminder\_frequency

+ "')"

)

c.commit()

c.close()

messagebox.showinfo(

"TASK ADDED!!", "TASK HAS BEEN ADDED SUCCESFULLY!!!"

)

# CALLS THE BELOW FUNCTION SO THAT USER IS TAKEN BACK TO THE MAIN MENU AFTER ADDING A TASK

back\_at\_fn()

# ADD AND BACK BUTTONS

add\_btn = tk.Button(

n\_win,

text=" ADD ",

font=("Fugaz One", 18),

bg="black",

fg="green",

command=add\_btn\_fn,

)

add\_btn.place(x=650, y=735)

back\_at\_btn = tk.Button(

n\_win,

text=" BACK ",

font=("Fugaz One", 18),

bg="black",

fg="red",

command=back\_at\_fn,

)

back\_at\_btn.place(x=750, y=735)

# FUNCTION TO EDIT TASK #

def edit\_task():

# SETTING A FLAG WITH EDIT TASK STATUS = FALSE

edit\_task\_status = False

# DISABLING ALL THE MENU BUTTONS SO THAT THE USER CANNOT ACCESS ANOTHER FUNCTION WHILE INSIDE THIS FUNCTION

add\_task\_btn.config(state="disabled")

edit\_task\_btn.config(state="disabled")

delete\_task\_btn.config(state="disabled")

view\_task\_btn.config(state="disabled")

back\_btn.config(state="disabled")

# WIDGETS TO DISPLAY THE LISTBOX WITH EVENTS AND THE EDIT AND BACK BUTTONS

events\_label = tk.Label(

n\_win,

text="THESE ARE YOUR EVENTS : ",

font=("Fugaz One", 18),

fg="white",

bg="black",

)

events\_label.place(x=165, y=225) # x=615, y=275

style = ttk.Style()

style.theme\_use("clam")

style.configure("Treeview", rowheight=25)

style.map("Treeview", background=[("selected", "blue")])

listbox\_scrollbar\_frame = tk.Frame(n\_win)

scrollbar = tk.Scrollbar(listbox\_scrollbar\_frame, orient="vertical")

h\_scrollbar = tk.Scrollbar(listbox\_scrollbar\_frame, orient="horizontal")

scrollbar.pack(side="right", fill="y")

h\_scrollbar.pack(side="bottom", fill="x")

events\_treeview = ttk.Treeview(

listbox\_scrollbar\_frame,

yscrollcommand=scrollbar.set,

xscrollcommand=h\_scrollbar.set,

selectmode="browse",

height=16,

)

events\_treeview.pack()

scrollbar.config(command=events\_treeview.yview)

h\_scrollbar.config(command=events\_treeview.xview)

events\_treeview.pack()

events\_treeview["columns"] = (

"TASK ID",

"REMINDER NAME",

"DATE",

"TIME",

"FREQUENCY",

"REMINDER OPTION",

)

# FORMATE THE COLUMNS

events\_treeview.column("#0", width=0)

events\_treeview.column("TASK ID", width=40, minwidth=40, anchor="center")

events\_treeview.column("REMINDER NAME", width=200, minwidth=200, anchor="w")

events\_treeview.column("DATE", width=130, minwidth=130, anchor="center")

events\_treeview.column("TIME", width=110, minwidth=110, anchor="w")

events\_treeview.column("FREQUENCY", width=100, minwidth=100, anchor="w")

events\_treeview.column(

"REMINDER OPTION", width=90, minwidth=90, anchor="center"

)

# CREATE HEADINGS

events\_treeview.heading("#0", text="", anchor="center")

events\_treeview.heading("TASK ID", text="S.NO", anchor="center")

events\_treeview.heading("REMINDER NAME", text="REMINDER NAME", anchor="center")

events\_treeview.heading("DATE", text="DATE", anchor="center")

events\_treeview.heading("TIME", text="TIME", anchor="center")

events\_treeview.heading("FREQUENCY", text="FREQUENCY", anchor="center")

events\_treeview.heading("REMINDER OPTION", text="REMINDER?", anchor="center")

# CONNECTING TO MySQL DATABASE

c = my.connect(

host="localhost", user="root", passwd="MYSQLVARY", database="todolist"

)

cur = c.cursor()

# TO GET ALL DATA FROM DATABASE FOR THE CURRENT USER

cur.execute(

"SELECT SNO, REMINDER\_NAME, REMINDER\_DATE, REMINDER\_TIME, REM\_FREQUENCY, REM\_OPTION, REM\_DESCRIPTION FROM usertasks WHERE username = '"

+ text

+ "' ORDER BY REMINDER\_DATE DESC"

)

tasks\_dt\_listbox = cur.fetchall()

c.close()

events\_treeview.tag\_configure(

"oddrow", background="white", font=("Source Code Pro", 12)

)

events\_treeview.tag\_configure(

"evenrow", background="lightblue", font=("Source Code Pro", 12)

)

c = 0

for i in tasks\_dt\_listbox:

if c % 2 == 0:

events\_treeview.insert(

parent="",

index="end",

iid=c,

values=(i[0], i[1], i[2], i[3], i[4], i[5]),

tags=("evenrow",),

)

else:

events\_treeview.insert(

parent="",

index="end",

iid=c,

values=(i[0], i[1], i[2], i[3], i[4], i[5]),

tags=("oddrow",),

)

c += 1

listbox\_scrollbar\_frame.place(x=10, y=275) # x=395, y=325

# FUNCTION OF BACK BUTTON INSIDE DELETE TASK #

def back\_et\_fn():

# TO NORMALISE THE BUTTONS THAT WERE DISABLED EARLIER

add\_task\_btn.config(state="normal")

edit\_task\_btn.config(state="normal")

delete\_task\_btn.config(state="normal")

view\_task\_btn.config(state="normal")

back\_btn.config(state="normal")

# TO MAKE ALL THE EDIT TASK WIDGETS DISAPPEAR

events\_label.destroy()

listbox\_scrollbar\_frame.destroy()

edit\_btn.destroy()

back\_et\_btn.destroy()

# FUNCTION OF EDIT BUTTON INSIDE EDIT TASK #

def edit\_btn\_fn():

if len(events\_treeview.selection()) == 1:

edit\_btn.config(state="disabled")

back\_et\_btn.config(state="disabled")

et = events\_treeview.selection()[0]

print(et)

print(tasks\_dt\_listbox[int(et)])

rem\_name\_label = tk.Label(

n\_win,

text="NAME OF THE REMINDER : ",

fg="white",

bg="black",

font=("Fugaz One", 18),

)

e\_rem\_name = tk.Entry(width=30, font=("Fugaz One", 18))

e\_rem\_name.insert(0, tasks\_dt\_listbox[int(et)][1])

e\_rem\_name.focus()

rem\_name\_label.place(

x=726, y=225

)

e\_rem\_name.place(x=1046, y=225)

cal = tkc.Calendar(

n\_win,

selectmode="day",

year=int(str(tasks\_dt\_listbox[int(et)][2])[0:4]),

month=int(str(tasks\_dt\_listbox[int(et)][2])[5:7]),

day=int(str(tasks\_dt\_listbox[int(et)][2])[8:]),

pady=10,

fill="BOTH",

)

cal.place(x=798, y=275)

reminder\_option\_label = tk.Label(

n\_win,

text="REMIND ME? : ",

font=("Fugaz One", 18),

fg="white",

bg="black",

)

reminder\_option\_label.place(x=1070, y=275)

rem\_option = tk.StringVar()

yes\_rem\_rb = tk.Radiobutton(

n\_win, text="YES", variable=rem\_option, value="Y"

)

no\_rem\_rb = tk.Radiobutton(

n\_win, text="NO", variable=rem\_option, value="N"

)

yes\_rem\_rb.place(x=1270, y=280)

no\_rem\_rb.place(x=1270, y=315)

if tasks\_dt\_listbox[int(et)][5] == "Y":

yes\_rem\_rb.select()

elif tasks\_dt\_listbox[int(et)][5] == "N":

no\_rem\_rb.select()

frequency\_label = tk.Label(

n\_win,

text="FREQUENCY OF REMINDERS : ",

fg="white",

bg="black",

font=("Fugaz One", 18),

)

frequency\_label.place(x=1070, y=380)

rec\_options = ["ONCE", "DAILY", "WEEKLY", "MONTHLY"]

recurring\_freq\_var = tk.StringVar()

if tasks\_dt\_listbox[int(et)][4] == "ONCE":

recurring\_freq\_var.set(rec\_options[0])

elif tasks\_dt\_listbox[int(et)][4] == "DAILY":

recurring\_freq\_var.set(rec\_options[1])

elif tasks\_dt\_listbox[int(et)][4] == "WEEKLY":

recurring\_freq\_var.set(rec\_options[2])

elif tasks\_dt\_listbox[int(et)][4] == "MONTHLY":

recurring\_freq\_var.set(rec\_options[3])

rec\_or\_not\_dropdown = tk.OptionMenu(

n\_win, recurring\_freq\_var, \*rec\_options

)

rec\_or\_not\_dropdown.place(

x=1428, y=385

)

hour\_label = tk.Label(

n\_win,

text="HOUR : ",

font=("Fugaz One", 18),

fg="white",

bg="black",

)

hour\_label.place(x=740, y=500)

hours = []

for i in range(0, 12):

if i < 10:

i = "0" + str(i)

hours.append(i)

else:

i = str(i)

hours.append(i)

hour\_var = tk.StringVar()

if len(str(tasks\_dt\_listbox[int(et)][3])) == 8:

if int(str(tasks\_dt\_listbox[int(et)][3])[0:2]) < 12:

if len(str(tasks\_dt\_listbox[int(et)][3])[0:2]) == 1:

hour\_var.set("0" + str(tasks\_dt\_listbox[int(et)][3])[0:2])

else:

hour\_var.set(str(tasks\_dt\_listbox[int(et)][3])[0:2])

amorpm = "a.m"

else:

if (

len(str(int(str(tasks\_dt\_listbox[int(et)][3])[0:2]) - 12))

== 1

):

hour\_var.set(

"0"

+ str(int(str(tasks\_dt\_listbox[int(et)][3])[0:2]) - 12)

)

else:

hour\_var.set(

str(int(str(tasks\_dt\_listbox[int(et)][3])[0:2]) - 12)

)

amorpm = "p.m"

elif len(str(tasks\_dt\_listbox[int(et)][3])) == 7:

if int(str(tasks\_dt\_listbox[int(et)][3])[0:1]) < 12:

hour\_var.set("0" + str(tasks\_dt\_listbox[int(et)][3])[0:1])

amorpm = "a.m"

hour\_dropdown = tk.OptionMenu(n\_win, hour\_var, \*hours)

hour\_dropdown.place(x=840, y=505)

minute\_label = tk.Label(

n\_win,

text="MINUTE : ",

font=("Fugaz One", 18),

fg="white",

bg="black",

)

minute\_label.place(x=1030, y=500)

minutes = []

for i in range(0, 60):

if i < 10:

i = "0" + str(i)

minutes.append(i)

else:

i = str(i)

minutes.append(i)

minute\_var = tk.StringVar()

if len(str(tasks\_dt\_listbox[int(et)][3])) == 8:

minute\_var.set(str(tasks\_dt\_listbox[int(et)][3])[3:5])

elif len(str(tasks\_dt\_listbox[int(et)][3])) == 7:

minute\_var.set(str(tasks\_dt\_listbox[int(et)][3])[2:4])

minute\_dropdown = tk.OptionMenu(n\_win, minute\_var, \*minutes)

minute\_dropdown.place(x=1160, y=505)

am\_pm\_label = tk.Label(

n\_win,

text="A.M / P.M : ",

font=("Fugaz One", 18),

fg="white",

bg="black",

)

am\_pm\_label.place(x=1320, y=500)

am\_pm\_list = ["a.m", "p.m"]

am\_pm\_var = tk.StringVar()

if amorpm == "a.m":

am\_pm\_var.set(am\_pm\_list[0])

elif amorpm == "p.m":

am\_pm\_var.set(am\_pm\_list[1])

am\_pm\_dropdown = tk.OptionMenu(n\_win, am\_pm\_var, \*am\_pm\_list)

am\_pm\_dropdown.place(x=1470, y=505)

rem\_desc\_label = tk.Label(

n\_win,

text="REMINDER DESCRIPTION : ",

font=("Fugaz One", 18),

fg="white",

bg="black",

)

rem\_desc\_label.place(

x=726, y=550

)

rem\_desc\_text = tk.Text(

n\_win, font=("Fugaz One", 18), height=5, width=30

)

# reminder\_description = tasks\_dt\_listbox[int(et)][6]

rem\_desc\_text.insert(1.0, str(tasks\_dt\_listbox[int(et)][6]))

rem\_desc\_text.place(x=1046, y=550)

# FUNCTION OF BACK BUTTON INSIDE ADD TASK #

def back\_at\_fn():

# TO NORMALISE THE EARLIER EDIT TASK WIDGETS THAT WERE DISABLED

edit\_btn.config(state="normal")

# listbox\_scrollbar\_frame.config(state="normal")

back\_et\_btn.config(state="normal")

# TO MAKE THE ADD TASK WIDGETS IN EDIT TASK FUNCTION TO DISAPPEAR

rem\_name\_label.destroy()

e\_rem\_name.destroy()

cal.destroy()

hour\_label.destroy()

hour\_dropdown.destroy()

minute\_label.destroy()

minute\_dropdown.destroy()

am\_pm\_label.destroy()

am\_pm\_dropdown.destroy()

rem\_desc\_label.destroy()

rem\_desc\_text.destroy()

add\_btn.destroy()

back\_at\_btn.destroy()

reminder\_option\_label.destroy()

yes\_rem\_rb.destroy()

no\_rem\_rb.destroy()

frequency\_label.destroy()

rec\_or\_not\_dropdown.destroy()

# FUNCTION OF UPDATE BUTTON INSIDE ADD TASK #

def update\_btn\_fn():

# TO GET ALL THE DETAILS ENTERED IN BY THE USER AS SOON AS UPDATE BUTTON IS CLICKED

reminder\_option = rem\_option.get()

reminder\_frequency = recurring\_freq\_var.get()

reminder\_name = e\_rem\_name.get()

print(reminder\_name)

reminder\_desc = rem\_desc\_text.get(1.0, "end")

print(reminder\_desc)

date\_raw = str(cal.get\_date())

ymd = date\_raw.split("/")

year = "20" + ymd[2]

if int(ymd[0]) > 9:

month = ymd[0]

else:

month = "0" + ymd[0]

if int(ymd[1]) > 9:

day = ymd[1]

else:

day = "0" + ymd[1]

date = year + "-" + month + "-" + day

hour = hour\_var.get()

minute = minute\_var.get()

am\_pm = am\_pm\_var.get()

if am\_pm == "p.m":

hour = str(int(hour) + 12)

time = str(hour) + ":" + str(minute) + ":" + "00"

# SETTING A FLAG ADD TASK STATUS = TRUE

add\_task\_table = True

# IF NAME OF THE REMINDER IS NULL, FLAG IS SET TO FALSE

if len(reminder\_name) == 0:

messagebox.showinfo(

"INVALID NAME OF REMINDER",

"NOTE : NAME OF THE REMINDER CANNOT BE NULL. ",

)

add\_task\_table = False

# IF NAME OF REMINDER EXCEEDS 30 CHARACTERS, FLAG IS SET TO FALSE

if len(reminder\_name) > 30:

messagebox.showinfo(

"INVALID NAME OF REMINDER",

"NOTE : NAME OF THE REMINDER HAS EXCEEDED 30 CHARACTERS.",

)

add\_task\_table = False

# IF REMINDER DESCRIPTION IS NULL, FLAG IS SET TO FALSE

if len(reminder\_desc) == 0:

messagebox.showinfo(

"INVALID DESCEIPTION OF REMINDER",

"NOTE : DESCRIPTION OF REMINDER CANNOT BE NULL.",

)

add\_task\_table = False

# IF REMINDER DESCRIPTION IS MORE THAN 150 CHARACTERS, FLAG IS SET TO FALSE

if len(reminder\_desc) > 150:

messagebox.showinfo(

"INVALID DESCEIPTION OF REMINDER",

"NOTE : DESRIPTION OF REMINDER HAS REACHED 150 CHARACTERS",

)

add\_task\_table = False

# AT END OF ALL IF CONDITIONS, IF ADD TASK TO TABLES FLAG IS TRUE, TASK IS UPDATE

if add\_task\_table == True:

print(date)

print(time)

print(reminder\_name)

print(reminder\_desc)

# CONNECTING TO MySQL DATABASE

c = my.connect(

host="localhost",

user="root",

passwd="MYSQLVARY",

database="todolist",

)

cur = c.cursor()

# UPDATING THE TASK IN TABLE WITH THE EDITED INFO AS THE USER WANTS IT TO BE

cur.execute(

"UPDATE usertasks SET reminder\_date='"

+ str(date)

+ "', reminder\_time='"

+ str(time)

+ "',reminder\_name='"

+ reminder\_name

+ "',rem\_description='"

+ reminder\_desc

+ "',rem\_option='"

+ reminder\_option

+ "',rem\_frequency='"

+ reminder\_frequency

+ "' where SNO="

+ str(tasks\_dt\_listbox[int(et)][0])

)

c.commit()

c.close()

messagebox.showinfo(

"TASK EDITED!!",

"TASK HAS BEEN EDITED SUCCESFULLY!!!",

)

back\_at\_fn()

back\_et\_fn()

# UPDATE AND BACK BUTTONS INSIDE EDIT TASK FUNCTION

add\_btn = tk.Button(

n\_win,

text=" UPDATE ",

font=("Fugaz One", 18),

bg="black",

fg="green",

command=update\_btn\_fn,

)

add\_btn.place(x=975, y=735)

back\_at\_btn = tk.Button(

n\_win,

text=" BACK ",

font=("Fugaz One", 18),

bg="black",

fg="red",

command=back\_at\_fn,

)

back\_at\_btn.place(x=1110, y=735)

# IF THERE ARE CURRENTLY NO TASKS PRESENT IN THE USER'S ACCOUNT

else:

messagebox.showinfo(

"NO TASK SELECTED",

"SELECT A TASK TO CONTINUE",

)

edit\_btn.config(state="normal")

back\_et\_btn.config(state="normal")

# EDIT AND BACK BUTTON

edit\_btn = tk.Button(

n\_win,

text=" EDIT ",

font=("Fugaz One", 18),

bg="black",

fg="red",

command=edit\_btn\_fn,

)

edit\_btn.place(x=210, y=735)

back\_et\_btn = tk.Button(

n\_win,

text=" BACK ",

font=("Fugaz One", 18),

bg="black",

fg="white",

command=back\_et\_fn,

)

back\_et\_btn.place(x=320, y=735)

# FUNCTION TO DELETE TASK #

def delete\_task():

# TO DISABLE THE OTHER FUNCTIONS IN THE MAIN MENU SO THAT THE USER CANNOT ACCESS OTHER TASKS FROM DELETE TASK FUNCTION

add\_task\_btn.config(state="disabled")

edit\_task\_btn.config(state="disabled")

delete\_task\_btn.config(state="disabled")

view\_task\_btn.config(state="disabled")

back\_btn.config(state="disabled")

# WIDGETS IN THE DELETE TASK FUNCTION

events\_label = tk.Label(

n\_win,

text="THESE ARE YOUR EVENTS : ",

font=("Fugaz One", 18),

fg="white",

bg="black",

)

events\_label.place(x=615, y=275)

style = ttk.Style()

style.theme\_use("clam")

style.configure("Treeview", rowheight=25)

style.map("Treeview", background=[("selected", "blue")])

listbox\_scrollbar\_frame = tk.Frame(n\_win)

scrollbar = tk.Scrollbar(listbox\_scrollbar\_frame, orient="vertical")

ho\_scrollbar = tk.Scrollbar(listbox\_scrollbar\_frame, orient="horizontal")

scrollbar.pack(side="right", fill="y")

ho\_scrollbar.pack(side="bottom", fill="x")

events\_treeview = ttk.Treeview(

listbox\_scrollbar\_frame,

yscrollcommand=scrollbar.set,

xscrollcommand=ho\_scrollbar.set,

selectmode="browse",

height=14,

)

events\_treeview.pack()

scrollbar.config(command=events\_treeview.yview)

ho\_scrollbar.config(command=events\_treeview.xview)

events\_treeview.pack()

events\_treeview["columns"] = (

"TASK ID",

"REMINDER NAME",

"DATE",

"TIME",

"FREQUENCY",

"REMINDER OPTION",

)

# FORMATE THE COLUMNS

events\_treeview.column("#0", width=0)

events\_treeview.column("TASK ID", width=60, minwidth=60, anchor="center")

events\_treeview.column("REMINDER NAME", width=200, minwidth=200, anchor="w")

events\_treeview.column("DATE", width=150, minwidth=150, anchor="center")

events\_treeview.column("TIME", width=110, minwidth=110, anchor="w")

events\_treeview.column("FREQUENCY", width=100, minwidth=100, anchor="w")

events\_treeview.column(

"REMINDER OPTION", width=140, minwidth=140, anchor="center"

)

# CREATE HEADINGS

events\_treeview.heading("#0", text="", anchor="center")

events\_treeview.heading("TASK ID", text="TASK ID", anchor="center")

events\_treeview.heading("REMINDER NAME", text="REMINDER NAME", anchor="center")

events\_treeview.heading("DATE", text="DATE", anchor="center")

events\_treeview.heading("TIME", text="TIME", anchor="center")

events\_treeview.heading("FREQUENCY", text="FREQUENCY", anchor="center")

events\_treeview.heading(

"REMINDER OPTION", text="REMINDER OPTION", anchor="center"

)

# CONNECTING TO MySQL DATABASE

c = my.connect(

host="localhost", user="root", passwd="MYSQLVARY", database="todolist"

)

cur = c.cursor()

# TO GET ALL DATA FROM DATABASE FOR THE CURRENT USER

cur.execute(

"SELECT SNO, REMINDER\_NAME, REMINDER\_DATE, REMINDER\_TIME, REM\_FREQUENCY, REM\_OPTION FROM usertasks WHERE username = '"

+ text

+ "' ORDER BY REMINDER\_DATE DESC"

)

tasks\_dt\_listbox = cur.fetchall()

c.close()

events\_treeview.tag\_configure(

"oddrow", background="white", font=("Source Code Pro", 12)

)

events\_treeview.tag\_configure(

"evenrow", background="lightblue", font=("Source Code Pro", 12)

)

c = 0

for i in tasks\_dt\_listbox:

if c % 2 == 0:

events\_treeview.insert(

parent="", index="end", iid=c, values=i, tags=("evenrow",)

)

else:

events\_treeview.insert(

parent="", index="end", iid=c, values=i, tags=("oddrow",)

)

c += 1

listbox\_scrollbar\_frame.place(x=395, y=325)

# FUNCTION OF BACK BUTTON INSIDE DELETE TASK #

def back\_dt\_fn():

# TO NORMALISE THE OTHER FUNCTIONS THAT WERE DISABLED EARLIER

add\_task\_btn.config(state="normal")

edit\_task\_btn.config(state="normal")

delete\_task\_btn.config(state="normal")

view\_task\_btn.config(state="normal")

back\_btn.config(state="normal")

# TO MAKE ALL DELETE TASK WIDGETS TO DISAPPEAR

events\_label.destroy()

listbox\_scrollbar\_frame.destroy()

delete\_btn.destroy()

back\_dt\_btn.destroy()

# FUNCTION OF DELETE BUTTON INSIDE DELETE TASK #

def delete\_dt\_fn():

if len(events\_treeview.selection()) == 1:

dt = events\_treeview.selection()[0]

print(dt)

print(tasks\_dt\_listbox[int(dt)])

c = my.Connect(

host="localhost",

user="root",

passwd="MYSQLVARY",

database="todolist",

)

cur = c.cursor()

qry = (

"DELETE FROM usertasks WHERE username='"

+ text

+ "' and SNO="

+ str(tasks\_dt\_listbox[int(dt)][0])

)

cur.execute(qry)

c.commit()

c.close()

messagebox.showinfo(

"TASK DELETED!!", "THE SELECTED TASK HAS BEEN DELETED!"

)

events\_treeview.delete(dt)

back\_dt\_fn()

elif len(events\_treeview.selection()) == 0:

messagebox.showinfo(

"NO TASK SELECTED",

"SELECT A TASK TO DELETE\nIF YOU DO NOT HAVE TASKS,\nADD TASKS AND USE THIS FUNCTIONALITY",

)

# DELETE AND BACK BUTTONS

delete\_btn = tk.Button(

n\_win,

text="DELETE",

font=("Fugaz One", 18),

bg="black",

fg="red",

command=delete\_dt\_fn,

)

delete\_btn.place(x=680, y=725)

back\_dt\_btn = tk.Button(

n\_win,

text=" BACK ",

font=("Fugaz One", 18),

bg="black",

fg="white",

command=back\_dt\_fn,

)

back\_dt\_btn.place(x=800, y=725)

# FUNCTION TO VIEW TASK #

def view\_task():

# TO DISABLE ALL THE OTHER MAIN MENU FUNCTIONS TO THAT USER CANNOT ACCESS OTHER FUNCTIONS INSIDE VIEW TASK FUNCTION

add\_task\_btn.config(state="disabled")

edit\_task\_btn.config(state="disabled")

delete\_task\_btn.config(state="disabled")

view\_task\_btn.config(state="disabled")

back\_btn.config(state="disabled")

# WIDGETS IN VIEW TASK FUNCTION

events\_label = tk.Label(

n\_win,

text="THESE ARE YOUR EVENTS : ",

font=("Fugaz One", 18),

fg="white",

bg="black",

)

events\_label.place(x=615, y=275)

style = ttk.Style()

style.theme\_use("clam")

style.configure("Treeview", rowheight=25)

style.map("Treeview", background=[("selected", "blue")])

listbox\_scrollbar\_frame = tk.Frame(n\_win)

scrollbar = tk.Scrollbar(listbox\_scrollbar\_frame, orient="vertical")

hor\_scrollbar = tk.Scrollbar(listbox\_scrollbar\_frame, orient="horizontal")

scrollbar.pack(side="right", fill="y")

hor\_scrollbar.pack(side="bottom", fill="x")

events\_treeview = ttk.Treeview(

listbox\_scrollbar\_frame,

yscrollcommand=scrollbar.set,

xscrollcommand=hor\_scrollbar.set,

selectmode="browse",

height=14,

)

events\_treeview.pack()

scrollbar.config(command=events\_treeview.yview)

hor\_scrollbar.config(command=events\_treeview.xview)

events\_treeview.pack()

events\_treeview["columns"] = (

"TASK ID",

"REMINDER NAME",

"DATE",

"TIME",

"FREQUENCY",

"REMINDER OPTION",

)

# FORMATE THE COLUMNS

events\_treeview.column("#0", width=0)

events\_treeview.column("TASK ID", width=60, minwidth=60, anchor="center")

events\_treeview.column("REMINDER NAME", width=200, minwidth=200, anchor="w")

events\_treeview.column("DATE", width=150, minwidth=150, anchor="center")

events\_treeview.column("TIME", width=110, minwidth=110, anchor="w")

events\_treeview.column("FREQUENCY", width=100, minwidth=100, anchor="w")

events\_treeview.column(

"REMINDER OPTION", width=140, minwidth=140, anchor="center"

)

# CREATE HEADINGS

events\_treeview.heading("#0", text="", anchor="center")

events\_treeview.heading("TASK ID", text="TASK ID", anchor="center")

events\_treeview.heading("REMINDER NAME", text="REMINDER NAME", anchor="center")

events\_treeview.heading("DATE", text="DATE", anchor="center")

events\_treeview.heading("TIME", text="TIME", anchor="center")

events\_treeview.heading("FREQUENCY", text="FREQUENCY", anchor="center")

events\_treeview.heading(

"REMINDER OPTION", text="REMINDER OPTION", anchor="center"

)

# CONNECTING TO MySQL DATABASE

c = my.connect(

host="localhost", user="root", passwd="MYSQLVARY", database="todolist"

)

cur = c.cursor()

# TO GET ALL DATA FROM DATABASE FOR THE CURRENT USER

cur.execute(

"SELECT SNO, REMINDER\_NAME, REMINDER\_DATE, REMINDER\_TIME, REM\_FREQUENCY, REM\_OPTION FROM usertasks WHERE username = '"

+ text

+ "' ORDER BY REMINDER\_DATE DESC"

)

tasks\_dt\_listbox = cur.fetchall()

c.close()

events\_treeview.tag\_configure(

"oddrow", background="white", font=("Source Code Pro", 12)

)

events\_treeview.tag\_configure(

"evenrow", background="lightblue", font=("Source Code Pro", 12)

)

c = 0

for i in tasks\_dt\_listbox:

if c % 2 == 0:

events\_treeview.insert(

parent="", index="end", iid=c, values=i, tags=("evenrow",)

)

else:

events\_treeview.insert(

parent="", index="end", iid=c, values=i, tags=("oddrow",)

)

c += 1

listbox\_scrollbar\_frame.place(x=395, y=325)

# FUNCTION OF BACK BUTTON INSIDE EDIT TASK #

def back\_vt\_fn():

# TO NORMALISE THE FUNCTIONS THAT WERE DISABLED EARLIER

add\_task\_btn.config(state="normal")

edit\_task\_btn.config(state="normal")

delete\_task\_btn.config(state="normal")

view\_task\_btn.config(state="normal")

back\_btn.config(state="normal")

# TO MAKE THE VIEW TASKS WIDGETS DISAPPEAR

listbox\_scrollbar\_frame.destroy()

events\_label.destroy()

back\_vt\_btn.destroy()

# BACK BUTTON IN VIEW TASKS

back\_vt\_btn = tk.Button(

n\_win,

text=" BACK ",

font=("Fugaz One", 18),

bg="black",

fg="white",

command=back\_vt\_fn,

)

back\_vt\_btn.place(x=730, y=725)

# SETTING UP ADD TASK, EDIT TASK, DELETE TASK, VIEW TASKS WIDGETS, BUTTONS, IMAGES, LABELS, ETC.

add\_task\_img = tk.PhotoImage(

file="IMAGES\_SUBMISSION\\add\_task.png"

)

add\_task\_btn = tk.Button(n\_win, image=add\_task\_img, command=add\_task)

add\_task\_label = tk.Label(n\_win, text="ADD TASK")

edit\_task\_img = tk.PhotoImage(

file="IMAGES\_SUBMISSION\\edit\_task.png"

)

edit\_task\_btn = tk.Button(n\_win, image=edit\_task\_img, command=edit\_task)

edit\_task\_label = tk.Label(n\_win, text="EDIT TASK")

delete\_task\_img = tk.PhotoImage(

file="IMAGES\_SUBMISSION\\delete\_task.png"

)

delete\_task\_btn = tk.Button(n\_win, image=delete\_task\_img, command=delete\_task)

delete\_task\_label = tk.Label(n\_win, text="DELETE TASK")

view\_img = tk.PhotoImage(

file="IMAGES\_SUBMISSION\\view\_task.png"

)

view\_task\_btn = tk.Button(n\_win, image=view\_img, command=view\_task)

view\_task\_label = tk.Label(n\_win, text="VIEW TASKS")

back\_img = tk.PhotoImage(

file="IMAGES\_SUBMISSION\\back.png"

)

back\_btn = tk.Button(n\_win, image=back\_img, command=n\_win.destroy)

# PLACING ALL WIDGETS IN THE NEW WINDOW

back\_btn.place(x=0, y=0)

add\_task\_btn.place(x=483, y=100)

add\_task\_label.place(x=488, y=180)

edit\_task\_btn.place(x=653, y=100)

edit\_task\_label.place(x=658, y=180)

delete\_task\_btn.place(x=823, y=100)

delete\_task\_label.place(x=820, y=180)

view\_task\_btn.place(x=983, y=100)

view\_task\_label.place(x=983, y=180)

n\_win.mainloop() # TO RUN THE MAIN WINDOW TILL BACK BUTTON OR CROSS

while True:

# STARTS WITH THE HOMEPAGE

homepage()

# IF GLOBAL VARIABLE IS TRUE, CALLS NEW WINDOW FUNCTION

if success == True:

new\_window(user)

# IF GLOBAL VARIABLE IS FALSE, CALLS NEW WINDOW FUNCTION, BREAKS THE LOOP AND COMES BACK RO HOMEPAGE

else:

break

## BACKGROUND APP

import mysql.connector as mysql

import smtplib

import datetime as dt

import time

import calendar

from plyer import notification

while True:

sys\_date = dt.date.today()

sys\_date\_str = str(sys\_date)

sys\_time = str(dt.datetime.now().time())[0:5]

c = mysql.connect(

host="localhost", user="root", passwd="MYSQLVARY", database="todolist"

)

cur = c.cursor()

cur.execute(

"SELECT usertasks.SNO, usertasks.username, mail, reminder\_date, reminder\_time, reminder\_name, rem\_description, rem\_option, rem\_frequency, reminded FROM USERPWD JOIN USERTASKS ON USERPWD.username = USERTASKS.username where reminder\_date = '"

+ str(sys\_date\_str)

+ "'"

)

user\_data = cur.fetchall()

c.close()

for i in user\_data:

if i[9] == "NO":

if i[7] == "Y":

if str(i[4])[0:5] == sys\_time:

print("time matched")

if i[8] == "ONCE":

delta = ""

new\_date = ""

c = mysql.connect(

host="localhost",

user="root",

passwd="MYSQLVARY",

database="todolist",

)

cur = c.cursor()

cur.execute(

"UPDATE USERTASKS SET reminded = 'YES' WHERE SNO="

+ str(i[0])

)

c.commit()

c.close()

elif i[8] == "DAILY":

delta = dt.timedelta(days=1)

new\_date = sys\_date + delta

elif i[8] == "WEEKLY":

delta = dt.timedelta(days=7)

new\_date = sys\_date + delta

elif i[8] == "MONTHLY":

days\_in\_month = calendar.monthrange(

sys\_date.year, sys\_date.month

)[1]

new\_date = sys\_date + dt.timedelta(days=days\_in\_month)

with smtplib.SMTP("smtp.gmail.com", 587) as smtp:

smtp.ehlo()

smtp.starttls()

smtp.ehlo()

smtp.login("apptodolist24@gmail.com", "todolist123")

subject = i[5]

body = i[6]

msg = f"Subject: {subject}\n\n{body}"

smtp.sendmail(

"apptodolist24@gmail.com",

i[2],

msg,

)

print(subject)

print(body)

notification.notify(

title="TO DO LIST",

message=i[5],

app\_icon="D://Varun//Class XII//CS XII//BOARD PROJECT//IMAGES//icon.ico",

timeout=20,

)

print("test" + str(new\_date))

if new\_date == "":

continue

else:

c = mysql.connect(

host="localhost",

user="root",

passwd="MYSQLVARY",

database="todolist",

)

cur = c.cursor()

cur.execute(

"UPDATE USERTASKS SET REMINDER\_DATE = '"

+ str(new\_date)

+ "' WHERE SNO="

+ str(i[0])

)

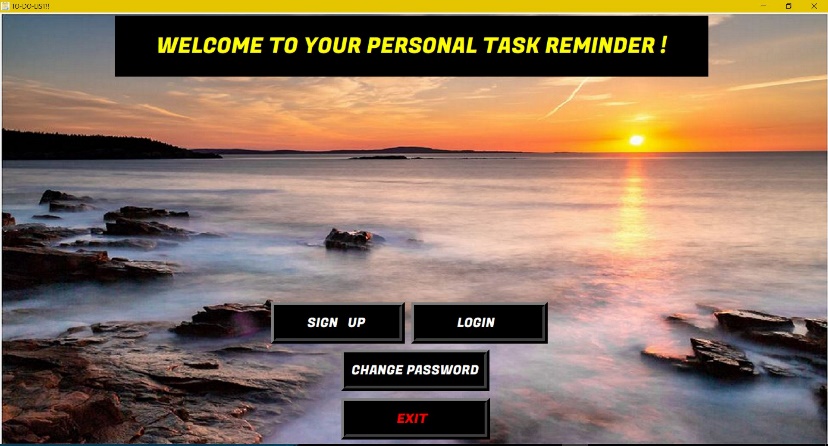
c.commit()

c.close()

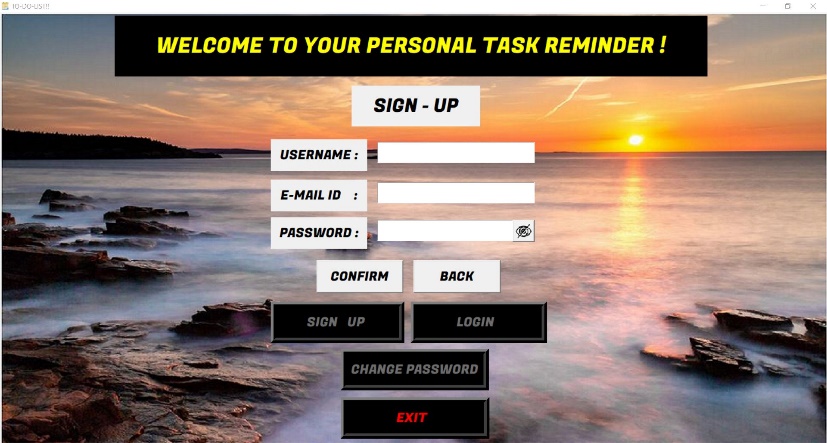
print("test2" + str(new\_date))

## SAMPLE SCREENSHOTS

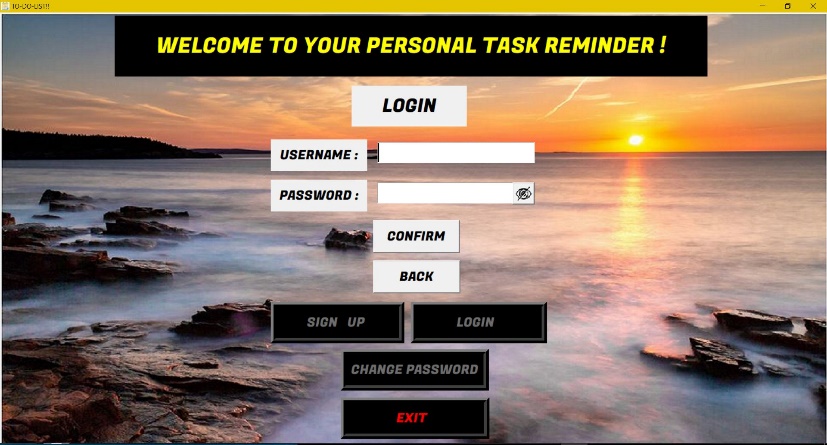
## HOME SCREEN:



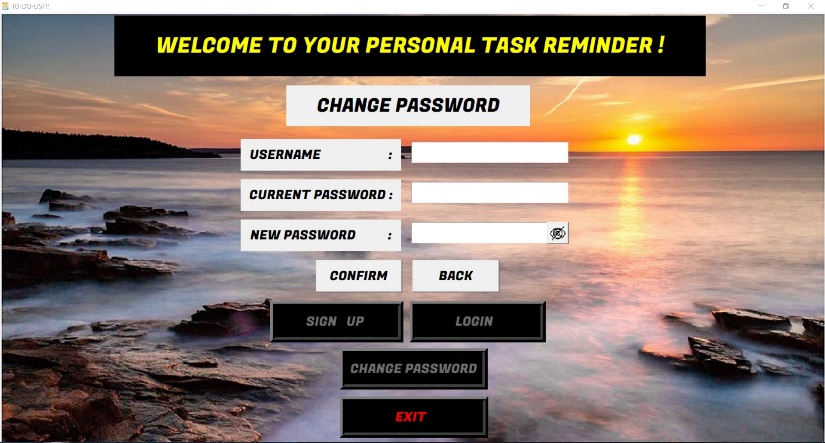
### SIGN – UP:



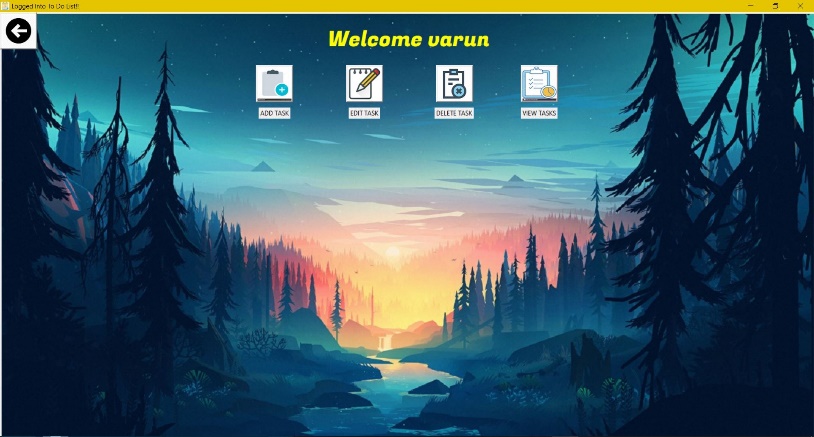
### LOGIN:



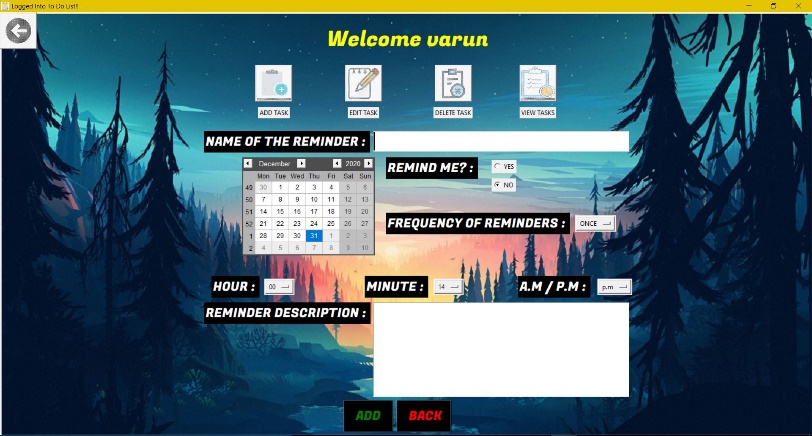
### CHANGE PASSWORD:



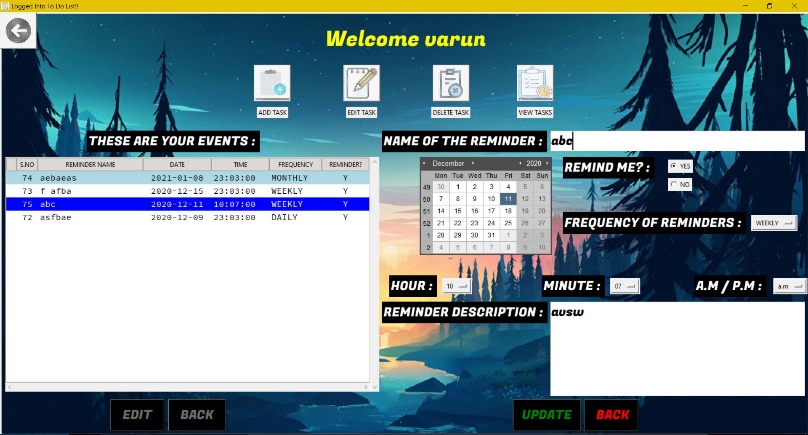
## MAIN SCREEN:



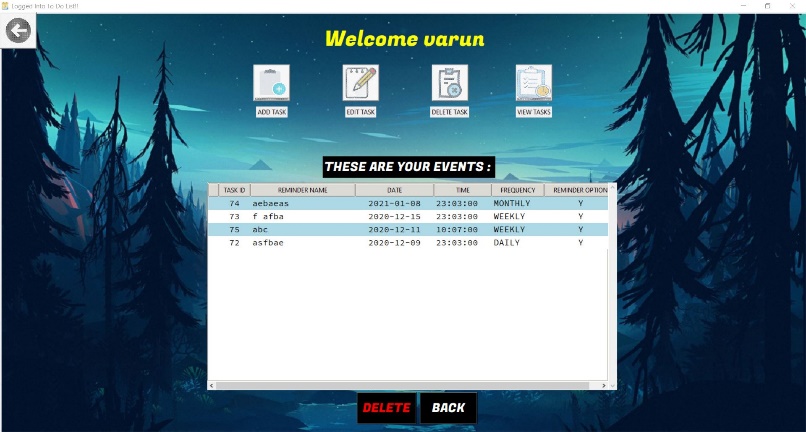
### ADD TASK:



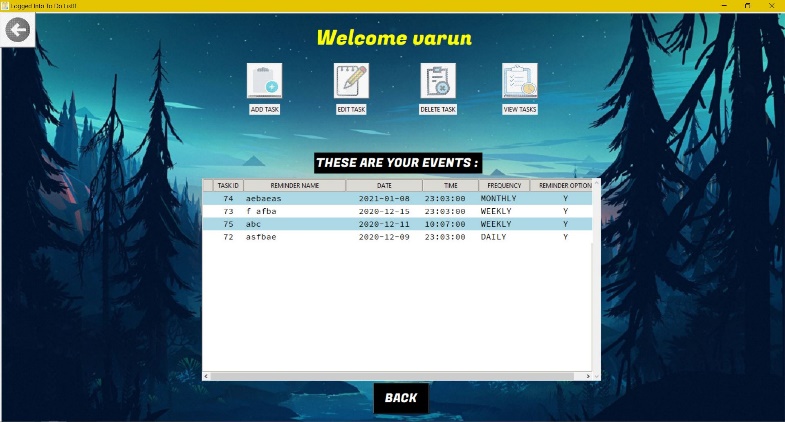
### EDIT TASK:



### DELETE TASK:



### VIEW TASK:



## LIMITATIONS OF THE PROJECT

## SYSTEM DISK SPACE

As a pre-requisite, application uses MySQL as backend for database management, so the system needs disk space (~1 GB) for the installation of MYSQL. The user needs to follow the steps given in the README.md file to install and create the database manually before using the application.

## MAIL NOTIFICATION

The system should be online at the time of the reminder scheduled.

## CLEANUP OF COMPLETED TASKS

A good practice to follow while using this application is to clean up the old tasks or completed tasks on a regular basis. The application allows the user to delete or edit one task at a time.

## SCOPE FOR IMPROVEMENT

## REMOTE DATABASE

The application can be enhanced to use a remote database rather than the local database which requires MySQL to be installed in the system.

## HOSTED APPLICATION

The application can be enhanced to a hosted application, where it can be deployed on cloud. This will provide flexibility for the user to create his/her to-do-list and get the notification offline. Also, a local database is not required in the system.

## FEATURE ENHANCEMENTS

The application can be enhanced with a few more user features such as:-

1. Snooze option for the reminder.
2. Add/delete multiple tasks at a time.
3. Search bar to find the tasks by name, date, time or frequency of the task.

## BIBLIOGRAPHY

## DOCUMENTATION TKINTER:

<https://www.tutorialspoint.com/python/python_gui_programming.htm>

## OFFICIAL PYTHON WEBSITE FOR Tcl/Tk:

<https://docs.python.org/3/library/tk.html>

## TTK WIDGETS:

<https://ttkwidgets.readthedocs.io/en/latest/>