In [1]:

```
import tkinter as tk
from tkinter import messagebox
import os
import hashlib
from PIL import Image, ImageTk
```

Entry Widget config

In [2]:

```
class Entry_widget():

    def __init__(self):
        # h = heighlight
        self.h_background = 'black'
        self.h_color = 'black'
        self.h_thickness = 1

    def highlight_border(self, EntryWidget,color=None):
        if color==None:
            self.h_background = 'black'
            self.h_color = 'black'
        else:
            self.h_background = color
            self.h_color = color

        EntryWidget.config(highlightbackground = self.h_background)
        EntryWidget.config(highlightcolor = self.h_color)
        EntryWidget.config(highlightthickness = self.h_thickness)

    def default_border(self,EntryWidget):
        self.h_background = 'black'
        self.h_color = 'black'
        EntryWidget.config(highlightbackground = self.h_background)
        EntryWidget.config(highlightcolor = self.h_color)
        EntryWidget.config(highlightthickness = self.h_thickness)


```

Label Widget Config

In [3]:

```
class Label_widget():

    def __init__(self):
        self.PasswordMatchError  = "Password is not matching.\nPlease confirm your pass
        self.Error_fg_color = 'red'
        self.Error_bg_color = 'white'

    def showerror(self, rootwindow):
        error = tk.Label(rootwindow, text=self.PasswordMatchError, fg=self.Error_fg_col
        return error
```

Parse User Details

In [4]:

```
class Write():

    def __init__(self, path=None):

        if path == None:
            self.path = "F:/Projects Data/Password manager/Users/"

        else:
            self.path = path

        self.filename ="Config.txt"


    def parse_text(self,Data):
        """
        This method will create a new config file for the current user.
        Data is a list or dict of User detalis.
        """
        filepath  = os.path.join(self.path,Data["User Name"])
        # first check dir already exist

        if os.path.isdir(filepath)!=True:
            os.mkdir(filepath)

            file = open(os.path.join(filepath, self.filename), 'w')

            for key in Data:
                file.write(key+"="+Data[key]+'\n')

            file.close()
        else:
            user = Data["User Name"]
            msg = f"User '{user}' is already exist try different User."
            messagebox.showerror(title="Regestration Error..", message=msg)



```

Get User Details

In [5]:

```
class ReadData():

    MissMatchError = "\nUserName or Password is wrong.\nplease Try different UserName c

    def __init__(self, path=None):
        # path is the root dir it can be changed
        if path == None:
            self.path = "F:/Projects Data/Password manager/Users/"

        else:
            self.path = path

        self.filename ="Config.txt"
#=============================================================================================
    @staticmethod
    def fetch_data(path):
        file = open(path,'r')
        username = None
        password = None

        line = file.readline()
        while(line != ''):
            #print(Data)
            Data = line.split("=")
            if Data[0] =="User Name":
                username = Data[1].strip()

            elif Data[0] == "User Password":
                password = Data[1].strip()
            else:
                break
            line = file.readline()

        return username, password
#=============================================================================================
    def Get_Data(self,UserName=None):

        if os.path.isdir(self.path):
            # Now we need t0 find the User Name (Dir as the Name of user)
            Users = os.listdir(self.path)
           # print("Users:>",Users)

            if UserName in Users:
                Userdir = os.path.join(self.path,UserName)
                # Examople:>
                # Userdir = "F:/Projects Data/Password manager/Users/Manish Kumar"
                # Now we need to read the Config file from the Manish Kumar directory
                if self.filename in os.listdir(Userdir):
                    #print("Found the Config file")
                    username , password = self.fetch_data(os.path.join(Userdir, self.fi
                    return {username:password}
                    #print(f"User Name:'{username}'")
                    #print(f"Password:'{password}'")

                    # we have retrived the user name and password, NOw we have to compa

                else:
                    print("\nError:Configure file Does not Exist.\nyou need to recover
```

```
                    return False
            else:
                print(self.MissMatchError)
                #raise NameError(f"User {UserName} is Not Found: Please check your User
                return False

            #print(Users)
            #return Users

        else:
            print("\nError:Root dir does not have any User.\nYou need to Regester befor
            return False
#=============================================================================================
    def Confirm_details(self, iusername, ipassword):

        UserDetails = self.Get_Data(iusername)

        if UserDetails==False:
            return False
        else:
            if iusername in UserDetails.keys() and UserDetails[iusername]!=ipassword:
                return False
            elif iusername in UserDetails.keys() and UserDetails[iusername]==ipassword:
                return True


        #return UserDetails

```

Test of user Details

In [6]:

```
def inputdata():
    iuser = input("Please Enter User Name: ")
    ipass = input("Please enter User Password: ")
    getdata = ReadData()
    Status = getdata.Confirm_details(iuser,ipass)

    print("\nInside inputdata function.",end="")
    print("\nStatus: ",Status)
    if Status == False:
        print("\nGot Mismatch Error.")
    elif Status == True:
        print("\nUnlocked..")


```

Regester an account

In [7]:

```
class RegersterAccount(tk.Frame):

    EntryConfig = {'bg':"White",'fg':'black','highlightbackground':"black",'highlightc
                   'highlightthickness':1}

    Rootdir = r"F:/Projects Data/Password manager"


    def __init__(self,master=None):
        # super().__init__(), method is used to initate the init method of the pared c
        # The Parent class for RegesterAccount is tk.Frame from which we are initiatin
        # We can also use tk.Frame.__init__(self, master)
        super().__init__(master)

        self.master = master # Now master is assigned to self.master
        self.master.title("Password Manager (Regester Account)")
        self.master.geometry("400x400")
        self.master.config(bg='white')
        self.master.iconbitmap(r"F:/Projects Data/Password manager/PadLock.ico")
        self.pack()

        self.logosize = (200,200)
        self.logoImage = None
        self.wentry = Entry_widget()
        self.wlabel = Label_widget()
        self.Create_Account = Write()

        self.RegesterationBlock()
        self.LoginProfileImage()

    def loadlogo(self,logopath="default"):
        if logopath=="default":
            logopath = "F:/Projects Data/Password manager/login.png"
            logoImage = Image.open(logopath,'r')
            logoImage = logoImage.resize(self.logosize)
            return logoImage
        else:
            logoImage = Image.open(logopath,'r')
            logoImage = Image.open(logopath,'r')
            logoImage = logoImage.resize(self.logosize)
            return logoImag

    def LoginProfileImage(self):
        LoginLogo = tk.Canvas(self.master, bg='white',width=200, height=200,)
        LoginLogo.pack(side=tk.TOP, fill=tk.BOTH)

        c_width = int(LoginLogo.cget('width'))
        c_height = int(LoginLogo.cget('height'))

        print("C-width :",c_width)
        print("C-height :",c_height)
        # loading the logo and passing it to PhotoImage Method
        self.logoImage = ImageTk.PhotoImage(self.loadlogo())
        LoginLogo.create_image(c_width,c_height/2,anchor=tk.CENTER, image=self.logoIma


    def RegesterationBlock(self):
        global regester
        regester = tk.Frame(self.master, bg='white',width=400, height=200)
```

```
60          regester.pack(side=tk.BOTTOM, fill=tk.BOTH)
61
62          self.UserNameLabel = tk.Label(regester,text="User Name :", bg="white").place(r
63          self.PasswordLabel = tk.Label(regester,text="Password :", bg="white").place(re
64          self.ConfirmPasswordLabel = tk.Label(regester,text="Confirm Password :", bg="w
65
66          self.UserNameText = tk.StringVar()
67          self.PasswordText = tk.StringVar()
68          self.PasswordText2 = tk.StringVar()
69
70          self.UserNameEntry = tk.Entry(regester, cnf=self.EntryConfig,textvariable=self
71          self.UserNameEntry.place(relx=0.3, x=30,y=0,anchor=tk.NW, width=200)
72
73          self.PasswordEntry = tk.Entry(regester, cnf=self.EntryConfig, textvariable=sel
74          self.PasswordEntry.place(relx=0.3, x=30,y=50,anchor=tk.NW, width=200)
75
76          self.ConfirmPasswordEntry =tk.Entry(regester, cnf=self.EntryConfig, textvariab
77          self.ConfirmPasswordEntry.place(relx=0.3, x=30,y=100,anchor=tk.NW, width=200)
78
79          print("In side Regestration Block",type(self.PasswordEntry))
80
81  #=============================================Get and Confirm the Regestration Detail
82
83      def GetDetails(self,event):
84          password_error = self.wlabel.showerror(regester)
85          user_created=None
86           # it will return StringVar object not the actual string of user name and Pass
87          # You need to access these values by get method
88          self.UserName = self.UserNameText.get()
89          self.Password = self.PasswordText.get()
90          self.Password2 = self.PasswordText2.get()
91
92          Userdict = {"User Name":self.UserName, "User Password":self.Password}
93
94          print(f"User Name : {self.UserName}")
95          print(f"Password : {self.Password}")
96          print(f"Password2 : {self.Password2}")
97
98          if len(self.Password) == 0:
99              # change the border color if password is empty
100             print("In side Get Details Conditions",type(self.PasswordEntry))
101             self.wentry.highlight_border(self.PasswordEntry,'red')
102
103         if len(self.Password2) == 0 :
104             # change the border color if password is empty
105             self.wentry.highlight_border(self.ConfirmPasswordEntry,'red')
106
107         if len(self.UserName)== 0:
108             # change the border color if User Name is empty
109             self.wentry.highlight_border(self.UserNameEntry,'red')
110
111         if self.Password == self.Password2 and (len(self.Password )!= 0 and len(self.P
112             self.wentry.highlight_border(self.PasswordEntry, 'green')
113             self.wentry.highlight_border(self.ConfirmPasswordEntry,'green')
114             self.wentry.highlight_border(self.UserNameEntry,'green')
115
116             self.Create_Account.parse_text(Userdict)
117             user_created =True
118
119         else:
120             self.wentry.highlight_border(self.PasswordEntry,'red')
```

```
            self.wentry.highlight_border(self.ConfirmPasswordEntry,'red')
            password_error.place(relx=0.3,x=0 ,y=130, anchor=tk.NW )


        def UserEvent(event):
            """
            Left click => Button-1
            Right Click => Button-3
            scroller/ Midddle Button  => Button-2
            """
            self.wentry.default_border(self.UserNameEntry)
            print("UserName Entry => Left click is pressed")

        def Pass1Event(envet):
            """
            Left click => Button-1
            Right Click => Button-3
            scroller/ Midddle Button  => Button-2
            """
            self.wentry.default_border(self.PasswordEntry)
            print("Password Entry => Left click is pressed")

        def Pass2Event(event):
            """
            Left click => Button-1
            Right Click => Button-3
            scroller/ Midddle Button  => Button-2
            """
            self.wentry.default_border(self.ConfirmPasswordEntry)
            print("Confirm Entry => Left click is pressed")

        # These bind should be inside of GetDetails Function.
        # Since it should Excute after Enter Excution and see if there is any error oc
        self.UserNameEntry.bind("<Button-1>",UserEvent)
        self.PasswordEntry.bind("<Button-1>",Pass1Event)
        self.ConfirmPasswordEntry.bind("<Button-1>",Pass2Event)

        if user_created:
            messagebox.showinfo(title="Regestration", message="Regestration completed.

            self.master.destroy()

```

In [8]:

```
#Reg = tk.Tk()

#RegApp = RegersterAccount(Reg)
#Reg.bind("<Return>",RegApp.GetDetails)

#RegApp.mainloop()
```

Loging Section

In [9]:

```
class Login(tk.Frame):
    EntryConfig = {'bg':"White",'fg':'black','highlightbackground':"black",'highlightcc
                   'highlightthickness':1}

    Rootdir = r"F:/Projects Data/Password manager"

    def __init__(self,master=None):
        super().__init__(master)
        self.master = master
        self.master.title("Password Manager(Login)")
        self.master.geometry("400x400")
        self.master.config(bg='white')
        self.master.iconbitmap(os.path.join(self.Rootdir,"PadLock.ico"))
        self.pack()
        self.logosize = (200,200)
        self.UserDetails = ReadData()
        self.LoginBlock()
        self.LoginProfileImage()
        self.USER_NAME = None

    def loadlogo(self,logopath="default"):
        if logopath=="default":
            logopath = "F:/Projects Data/Password manager/login.png"
            logoImage = Image.open(logopath,'r')
            logoImage = logoImage.resize(self.logosize)
            return logoImage
        else:
            logoImage = Image.open(logopath,'r')
            logoImage = Image.open(logopath,'r')
            logoImage = logoImage.resize(self.logosize)
            return logoImag

    def UserName(self, username):
        # get user name from Entry
        self.username = username

    def Password(self,password):
        # get Password from Entry
        self.Password = Password

    def LoginProfileImage(self):
        global canvasImage
        # Fisrt Set the canvs for the login image
        ProfileImageCanvas = tk.Canvas(self.master, bg='white', width=400, height=200)
        ProfileImageCanvas.pack(side=tk.TOP)

        c_width = int(ProfileImageCanvas.cget('width'))
        c_height = int(ProfileImageCanvas.cget('height'))

        # Set the login image at the center of the image
        canvasImage = ImageTk.PhotoImage(self.loadlogo())
        ProfileImageCanvas.create_image(c_width/2,c_height/2,
                                                anchor=tk.CENTER,image=canvasImage

    def LoginBlock(self):
        global login
        login = tk.Frame(self.master, bg='white',width=400, height=200)
        login.pack(side=tk.BOTTOM, fill=tk.BOTH)

```

```
        self.UserNameLabel = tk.Label(login,text="User Name :", bg="white")
        self.UserNameLabel.place(relx=0.1 ,x=0, y=0, anchor=tk.NW)

        self.PasswordLabel = tk.Label(login,text="Password :", bg="white")
        self.PasswordLabel.place(relx= 0.1, x=0, y=50, anchor=tk.NW)

        self.UserNameText = tk.StringVar()
        self.PasswordText = tk.StringVar()

        self.UserNameEntry = tk.Entry(login, textvariable=self.UserNameText, cnf=self.E
        self.UserNameEntry.place(relx=0.3, x=10,y=0,anchor=tk.NW, width=200)

        self.PasswordEntry = tk.Entry(login, textvariable=self.PasswordText, cnf=self.E
        self.PasswordEntry.place(relx=0.3, x=10,y=50,anchor=tk.NW, width=200)


    def GetDetails(self,event):
        iusername = self.UserNameText.get()
        ipassword = self.PasswordText.get()

        result = self.UserDetails.Confirm_details(iusername, ipassword)
        if result==False:
            warningtext = "UserName or Password is Wrong.\nPlease try different UserNam
            WrongPasswordLabel = tk.Label(login,bg='white',fg='red', text=warningtext,
            WrongPasswordLabel.place(relx=0.5, rely=0.5, x=0,y=50, anchor=tk.CENTER)
        elif result==True:
            self.USER_NAME = iusername
            print("System unlocked..")
```

In [10]:

```
#root =tk.Tk()
#app = Login(root)
#root.bind("<Return>",app.GetDetails)
#app.mainloop()
```

Main Class of the Applocation

Class for Managing Enteries

In [11]:

```
class ManageEnteries():

    def __init__(self):
        self.Account = dict()
        self.AccountList = []
        self.count_iter = 0
#=============================================================================================
    def ReadEnteries(self, path=None):
        account = None
        ids=None
        password=None

        # check for entry file existance
        if os.path.isfile(path):
            # we need ACname, id, password
            file = open(path,'r')

            line = file.readline()

            while(line!=""):
                values =[]
                self.count_iter +=1 # Count +1 after every line.

                #print(line,end="")
                code = line.split("=")[0].strip()
                value = line.split("=")[1].strip()

                if code == "AcName":
                    #print("Fetched AcName")
                    account = line.split("=")[1].strip()

                elif code=="id":
                    #print("Fetched id")
                    ids = line.split("=")[1].strip()

                elif code=="password":
                    #print("Fetched password")
                    password = line.split("=")[1].strip()

                if self.count_iter == 3:
                    self.Account = {account:{ids:password}}
                    self.AccountList.append(self.Account)
                    self.count_iter=0

                line = file.readline()

            return self.AccountList

        else:
            print("Directory of enteries does not found please check your entery dirs.'

    def FindEnteries_pass(self,Account,path):

        if os.path.isfile(path):
            # checking if the file exist or not
            file = open(path, 'r')

            account = file.readline().split("=")[1].strip()
            while(account!=""):
```

```
60
61                 if account==Account:
62                     file.readline()
63                     password = file.readline().split("=")[1].strip()
64                     return password
65                     break
66                 try:
67                     account = file.readline().split("=")[1].strip()
68                 except:
69                     break
70             file.close()
71 #=============================================================================================
72
73     def readEnteriesxml(self):
74         print("This Function is not ready to use yet.")
75
```

In [12]:

```
1 Menteries = ManageEnteries()
2 path = "F:/Projects Data/Password manager/Users/Manish Kumar/Enteries.txt"
3 Enteries = Menteries.FindEnteries_pass("Gmail",path)
4 print(Enteries)
```

```
moni9782
```

In [13]:

```
1  Menteries = ManageEnteries()
2  path = "F:/Projects Data/Password manager/Users/Manish Kumar/Enteries.txt"
3  Enteries = Menteries.ReadEnteries(path)
4
5  for entry in Enteries:
6      Account = list(entry.keys())[0]
7      Accountid = list(list(entry.values())[0].keys())[0]
8      Password = list(list(entry.values())[0].values())[0]
9      print(f"Account : {Account}")
10     print(f"Account id : {Accountid}")
11     print(f"Password : {Password}")
12     print()
```

```
Account : Gmail
Account id : manishkumaraim9782@gmail.com
Password : moni9782

Account : Microsoft
Account id : manishkumaraim9782@hotmail.com
Password : moni72525

Account : Facebook
Account id : manish9782
Password : moni
```

Note:>

To get the variable of a child label or entry widget we need to follow

1. Get access the parent window and use `parent.winfo_childern()` it will return all the widget.

2. if you want to work on a specific widget you can use to identify `isinstance(object, class)` method.
3. Now we have a list of all the widget and found the our desired widget label or Entry
4. Label or Entry both have `textvariable` option so we need to get their respective variable
5. For that we need to make a variable for every label or entry which we want access later.
6. We should define the variable at time of difining the Label and Entry widgets.
7. EVery Label of Entry widget Variable can be accessed from their optinal functions
8. So we will use `getvar()` and `setvar()` metod to get and set the variable values
9. for that we need to assign the name before accessing the variable
10. this is need to done at the defination of the variable like `var = StringVar(name="var1")`
11. `StringVar(master, name, value)` it takes three parametere

In [16]:

```
class PasswordManager(tk.Frame):


    EntryConfig = {'bg':"White",'fg':'black','highlightbackground':"black",'highlightc
                   'highlightthickness':1}

    enteriesLabelConfig = {'bg':"White",'fg':'black' ,'relief':tk.FLAT,'padx':10,'pady

    entryFrameConfig = {'bg':"white",'bd':1 ,'relief':tk.RAISED,'padx':10,'pady':0,
                        'highlightbackground':"black",'highlightcolor':"black",
                       'highlightthickness':1,'width':700,'height':40}

    Rootdir = r"F:/Projects Data/Password manager"

    def __init__(self,master=None):
        super().__init__(master)

        self.master.title("Password Manager")
        self.master.geometry("900x700")
        self.master.iconbitmap(os.path.join(self.Rootdir, "PadLock.ico"))
        self.master.config(bg="white")
        self.enteries = ManageEnteries()
        self.FramebuttonList = []
        self.EntryFrameList =[]



    # This main application shows the all User and password enteries saved in the root
    # First Get enteries

    def Create_EntryFrame(self, master):
        print("Created one Frame")
        EntryFrame = tk.Frame(master, cnf=self.entryFrameConfig)
        EntryFrame.pack(side=tk.TOP)
        return EntryFrame

    def showpassword(self, master):
        pass

    def hidepassword(self, master):
        # now we need to change the value of on the button
        # master , button
        pass
#============================================Entry Labes ============================


    def EntryLabels(self,master,account,_id,password):
        self.value =  tk.StringVar()
        #print("Created one Account Labels")
        csp=1
        space=200
        # .grid_columnconfigure need to use to set the label at their desired position
        # Note :> we need to make a list of the frames in which all the labels are pos
        # Frame Tracking will help use to update the values of the labes in future.

        master.grid_columnconfigure(0,minsize=150)
        AccountLabel = tk.Label(master,cnf=self.enteriesLabelConfig, text =account, an
        AccountLabel.grid(row=0, column=0, padx=0, pady=5,sticky=tk.W)

        master.grid_columnconfigure(1,minsize=space+50)
```

```
60          AccoitIdLabel = tk.Label(master,cnf=self.enteriesLabelConfig, text=_id ,anchor
61          AccoitIdLabel.grid(row=0, column=1, padx=0, pady=5,sticky=tk.W)
62
63          master.grid_columnconfigure(2,minsize=space)
64          Passwordvar = tk.StringVar(master=master,name=account, value=password)
65
66          PasswordLabel = tk.Label(master,cnf=self.enteriesLabelConfig, text=password ,t
67          PasswordLabel.grid(row=0, column=2, padx=0, pady=5,sticky=tk.W)
68          #print(Passwordvar.get())
69
70          master.grid_columnconfigure(3,minsize=40)
71          self.value.set("show")
72          visibleButton = tk.Button(master, textvariable=self.value, anchor=tk.E,bg='whi
73          #visibleButton.
74          visibleButton.grid(row=0, column=3, sticky=tk.E,padx=10)
75
76          self.FramebuttonList.append(visibleButton)
77
78  #=============================================Configure Section=======================
79      # In this Section we will configure all the Frames, Labels, Menus and Buttons
80
81      def ButtonList_Config_command(self):
82              # This function sets the command to the Buttons individually by theirs ids
83          for i, entry_button in enumerate(self.FramebuttonList):
84              entry_button.config(command=(lambda c=i: self.visible(c)))
85
86      def Frame_Config_pass(self,Button_id,Status=False):
87          # tested on 0
88          # Frame config will take the Button_id , by using the Button id we will select
89          # Now we can make the password visible and non-visible (hide)
90
91          path = "F:/Projects Data/Password manager/Users/Manish Kumar/Enteries.txt"
92
93          #passwordlabel receives the password Label id from its Master Frame
94          passwordlabel = self.EntryFrameList[Button_id].winfo_children()[2]
95          #AccountLabel receives the Account label id from its Master Frame
96          AccountLabel = self.EntryFrameList[Button_id].winfo_children()[0]
97          AcName =AccountLabel.cget('text')
98
99          #getting the password by the name of account
100         password =passwordlabel.getvar(name= AcName)
101         passlen = len(password)
102         #.....................................................
103         def showdots(length):
104             dots = str()
105
106             for i in range(length):
107                 dots +="*"
108                 if i>8:
109                     break
110             return dots
111         #.....................................................
112
113         if Status == False:
114             passwordlabel.setvar(name= AcName,value=showdots(passlen))
115
116         elif Status==True:
117             # Accessing Gmail at 0
118             password = self.enteries.FindEnteries_pass(AcName,path)
119             passwordlabel.setvar(name= AcName,value=password)
120
```

```
121
122
123         #print(AccountLabel.cget('text'))
124         #print("master :> ",Master)
125         #print(passwordlabel.getvar(name= AcName))
126
127         # Since Every fram has their children list , we can use this list
128
129
130     #.....................................................................
131
132 #=======================================Visibility Section=========================
133
134     # This function will helpt use to change the Buttton text of image or bitmap  when
135     # and  The main use of these function is to show and hide the password values
136
137     def Button_show(self,ids):
138         # if the Button text is show , then first we need to call decode function for
139         vision = self.FramebuttonList[ids].cget('text')
140         cr_value = tk.StringVar()
141
142         self.FramebuttonList[ids].config(textvariable=cr_value)
143         if vision == "show":
144             cr_value.set("hide")
145             #Status = True
146             self.Frame_Config_pass(ids,True)
147
148         else:
149             pass
150
151
152     def Button_hide(self,ids):
153         vision = self.FramebuttonList[ids].cget('text')
154         cr_value = tk.StringVar()
155         self.FramebuttonList[ids].config(textvariable=cr_value)
156         if vision == "hide":
157             cr_value.set("show")
158             # Status False
159             self.Frame_Config_pass(ids,False)
160         else:
161             pass
162
163
164     def visible(self,ids):
165         vision = self.FramebuttonList[ids].cget('text')
166         if vision == "show":
167             self.Button_show(ids)
168         else:
169             self.Button_hide(ids)
170
171
172 #=================================== Get Enteries of Accounts====================
173     # This section will Get the enteries saved in the files and make them to show on t
174     # by making new Frame for every entry which contains AcName, Account_id, Password,
175
176     def GetEnteries(self,UserName):
177         EnteriesPath = f"User/{UserName}/Enteries.txt"
178         print(EnteriesPath)
179         # this is a test path
180         path = "F:/Projects Data/Password manager/Users/Manish Kumar/Enteries.txt"
181
```

```
        Enteries = self.enteries.ReadEnteries(path)

        if Enteries !=None:
            x=int(self.master.cget('width'))/2
            y=0

            for entry in Enteries:
                # getting the all key and value paire into Account , AAccountid, Passw
                y+=50

                Account = list(entry.keys())[0]
                Accountid = list(list(entry.values())[0].keys())[0]
                Password = list(list(entry.values())[0].values())[0]

                # This is EntryFrame in which every entry will be holded by this Frame
                EntryFrame = tk.Frame(self.master,cnf=self.entryFrameConfig)
                EntryFrame.place(relx=0.1,x=x,y=y,anchor=tk.NW)
                EntryFrame.grid_propagate(0)

                self.EntryLabels(EntryFrame,Account,Accountid,Password)
                # We need to make a Entry FrameList to track every enteery with their
                self.EntryFrameList.append(EntryFrame)

            self.ButtonList_Config_command()
            # Buttons has been set for the command individualy now we need to function

            # initiating the hiding of the passwords

            for i in range(len(self.EntryFrameList)):
                self.Frame_Config_pass(i,False)
        else:
            print("Unable to read the Enteries please check your root dir regestry.")
#================================================================================
print("Run Complete.")
```

```
Run Complete.
```

In [17]:

```
root = tk.Tk()

MainApp =PasswordManager(root)
MainApp.GetEnteries("Manish Kumar")

MainApp.mainloop()

```

```
User/Manish Kumar/Enteries.txt
```

In [ ]:

```

```