COMPUTER SCIENCE PROJECT 2021-22

SecureVault
Password Manager

NAME : G. Venkata Aravind Deepak

CLASS : XII MPC CS

ROLL NO :

SRI SATHYA SAI HIGHER SECONDARY SCHOOL
VIDYA GIRI,PRASHANTHI NILAYAM -515134

CERTIFICATE

REGD. NO: _____

CERTIFIED THAT THIS IS THE BONAFIDE PROJECT WORK DONE BY

_____ ,

IN THE AISSCE COURSE IN THE SUBJECT OF _____

DURING THE ACADEMIC YEAR 2021-22.

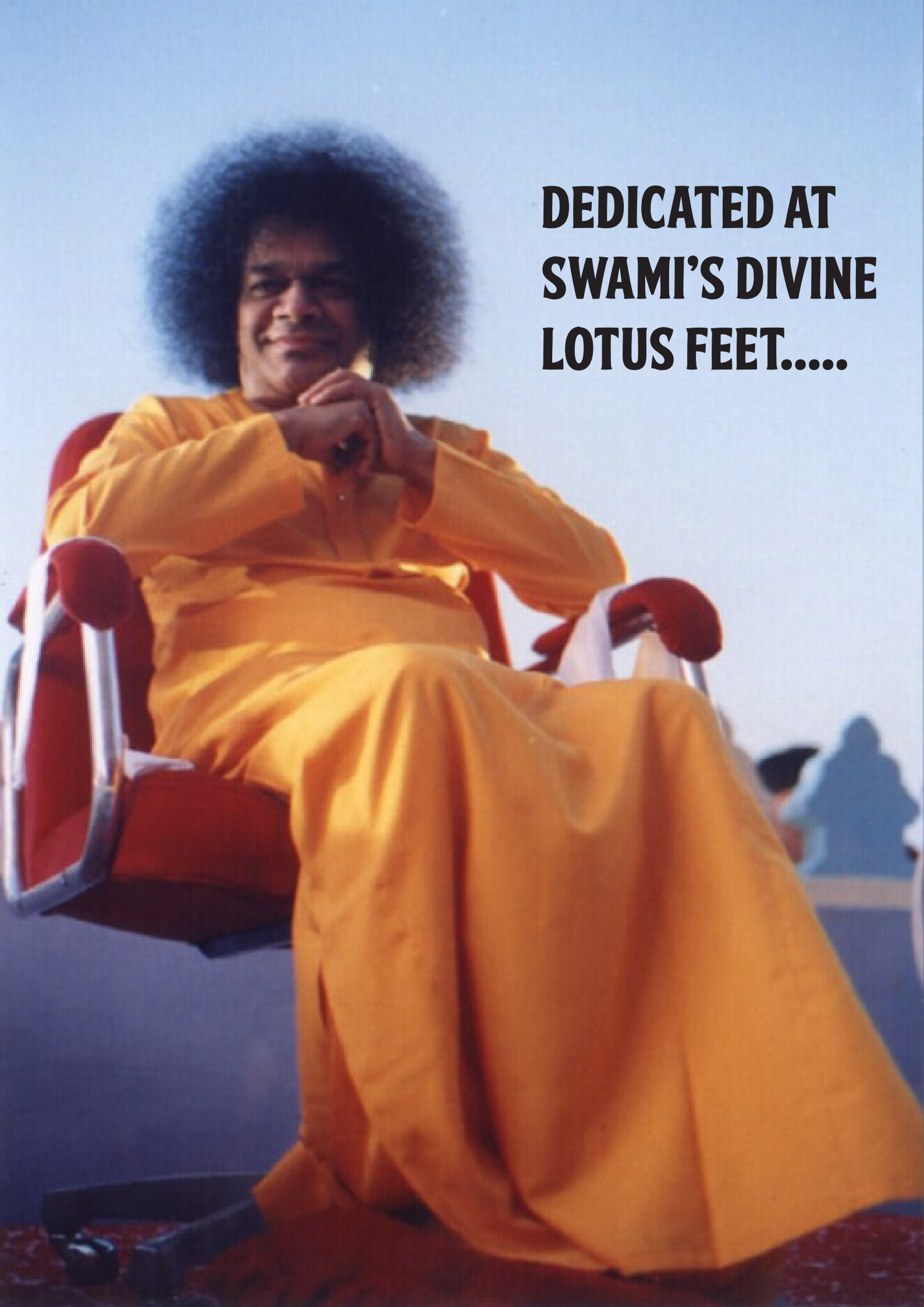
THIS PROJECT WORK IS SUBMITTED FOR THE PRACTICAL EXAMINATION CONDUCTED BY THE CENTRAL BOARD OF SECONDARY EDUCATION, NEW DELHI IN MARCH 2022.

DATE:

PRINCIPAL

INTERNAL EXAMINER

EXTERNAL EXAMINER


DEDICATED AT
SWAMI'S DIVINE
LOTUS FEET.....

ACKNOWLEDGEMENTS

First of all, I would like to thank my dearest SWAMI, the ever compassionate one for each and everything, whoser unseen hand helped me throughout the making of this Application.

My Computer Science Teacher Mr. VENKATESWAR PRUSTY for introducing me to and teaching me Python, without which I would not have been able to this project, I thank him. I would also like to thank our Principal Sir Mr. SIVARAMAKRISHNAIAH for all his support.

I also take this opportunity to thank my parents, who were an integral support all through this project, by giving their guidance and advice. I also thank my brother without whose help I wouldn't have gotten the idea to make this Application.

Finally, I would like to thank my friends who encouraged me throughout to finish this project.

CONTENTS

AIM

To Create an Interative Password Manager with ability to store and retrieve passwords of multiple users with high security using Python and various modules.

INTRODUCTION

Passwords are the first level of security in the digital domain. It could be accessing your email account or your favourite OTT platform or logging into your online cloud gaming account or opening a banking site to make important payments. In all these scenarios, passwords are the first step is to login.

But when you have the same password for all your accounts, even if one of your password is leaked it could lead to risk of your data and accounts.
Nowadays, every online site or service asks you to turn on 2-Factor Authentication and keep a different password for each account to prevent breach of data and privacy even if one of your password is leaked.

A strong password is defined as a password which has a mix of capital and small letters, numbers and symbols has a length more than 7 characters. But it is tough to remember all your passwords and keep them as strong passwords.

So to fix this problem, Password Managers have been developed but most of the password managers available today have a free and pro tier and therefore people who don't want to spend their money and yet have all the features face a tough choice on what to choose. This problem is what inspired me to make a free and secure password manager for all

THEORY

This App, SecureVault Password Manager is an app to store all your passwords safely and securely using encryption. This app helps individuals to remember all their passwords while also maintaining them securely.

This App was developed using the Tkinter GUI module in python. The data management was setup using MySQL which is very good due to its easy accessibility.

The App provides strong password suggestions using the random module and therefore gives truly random passwords to users and users can still customise it.

The Pyperclip module was used to interface the app with the windows clipboard and therefore users have the option to copy and paste their details.

PIL was the module used to manage images in the App and many of its capabilities like resize and more were used.

The App also uses the hashlib module which is responsible for the encryption of passwords and therefore providing real time security.

The OS module was used to intelligently choose where to save details the user wanted to export.

PROGRAM DESIGN

Two Main Functions called Login And Signup were incorporated for the main functioning of the program. For ease of understanding, these main functions and other functions which are required for the functioning of the app are explained here.

Functions:

Signup:

This Function, as the name suggests takes care of all the new signups into the app and dynamically manages it with the help of MySQL DBMS and provides Blake2B encryption which is a very secure encryption algorithm but also has the speed of the first gen encryption algorithms. Within the Signup Function, there is another function called credentialcheck which takes care of the incoming signup requests and checks if the request can be completed.

Login:

This Function also, as the name suggests allows users to login to the app and then use the app functionalities. The function consists of many sub functions which constitute the main functionalities of the App.

i)add_pword():

This Function takes care of adding new passwords. Whenever the user wants to add a new site and its login details, this function is used.

ii)update_pword():

This Function takes care of updating an existing set of details where the user can select the name of the site for which he wants to edit the details and then he can edit them.

iii)delete_pword():

This Function takes care of deleting a set of existing login details where the user can select the name of the site for which he wants to remove the details.

iv)savefiles():

This Function takes care of exporting all of the user details which the user has saved in the software into a csv file and this can then be accessed by the user on microsoft excel or he can import all these passwords into one more account.

v)importfromcsvfile():

This function takes care of importing user login details from a csv file and the data in the file should be in the form of sitename, username, password or else the app will not accept the request.

Other than Login and Signup Main Functions, there are other functions which are required for the app to run.

a)randompg():

This function is the function resonsible for creating random passwords for users which are safe and totally random. This function takes help of the random module to achieve this. The passwords are a combo of upper case, lower case, numbers and special symbols.

b)go_to_next_entry():

This function takes care of setting focus on entries in tkinter windows so that the user can go through entries by pressing enter button.

c)remove_window():

This Function takes care of removing a tkinter window after its use is completed.

d)hashing_password():

This Function takes care of hashing the app user passwords using high level hashing algorithm to increase security of user details.

e)copy_from_treeviewuap():

This Function takes care of allowing users to copy their username and password together to the clipboard allowing the user to paste it elsewhere.

f)copy_from_treeviewcol():

This Function takes care of allowing users to copy only their username or passwords to the clipboard so that the user can paste it elsewhere.

g)images():

This Function takes care of getting all the images the app uses and resizing them to suit the need. The images are used in multiple parts of the app.

h)start_menu():

This Function is the main start function and it takes care of taking the users to the SignUp and Login Functions.

i)refresh():

This Function takes care of refresing the details of the user and again shows the user saved details in a proper way.

PROGRAM CODE

```
import hashlib
import random
from tkinter import *
from tkinter import filedialog
from tkinter import messagebox
from tkinter import ttk

import mysql.connector
import pyperclip
from PIL import Image, ImageTk

mydb = mysql.connector.connect(host="localhost",
user="root", passwd="sairam123")
mycursor = mydb.cursor()
mycursor.execute("CREATE DATABASE IF NOT EXISTS
forpython")
mydb = mysql.connector.connect(host="localhost",
user="root", passwd="sairam123", database="forpython")
mycursor = mydb.cursor()
mycursor.execute("CREATE TABLE IF NOT EXISTS login ( username VARCHAR(255)
                    PRIMARY KEY , password VARCHAR(255))")
```

```
def images():
    global backButton, showPassword, SignUpPage, hidePassword, mainPage, forgotPage,
                                addPword, updatePword, deletePword, mainPage1, home1
    backButton = Image.open(r'backbutton.jpg')
    showPassword = Image.open(r'showPassword (2).jpg')
    showPassword = showPassword.resize((35, 23))
    hidePassword = Image.open(r'hidePassword (2).jpg')
    hidePassword = hidePassword.resize((35, 27))
    SignUpPage = Image.open('SignUp-01.png')
    SignUpPage = SignUpPage.resize((600, 400))
    mainPage = Image.open(r'Main-01.png')
    mainPage = mainPage.resize((800, 600))
    forgotPage = Image.open(r'forgotpage-01.png')
    forgotPage = forgotPage.resize((400, 300))
    addPword = Image.open(r'AddPword-01.png')
    addPword = addPword.resize((420, 320))
    updatePword = Image.open(r'updatePage-01-01.png')
    updatePword = updatePword.resize((400, 300))
    deletePword = Image.open(r'deletePword-01.png')
    deletePword = deletePword.resize((300, 200))
    mainPage1 = Image.open(r'mainPage-01.png')
    mainPage1 = mainPage.resize((300, 300))
    home1 = Image.open(r'home-01.png')
    home1 = home1.resize((35, 22))
```

```
root = Tk()
root.title("Password Manager")
root.geometry("300x300+520+220")
root.iconbitmap(r"password-manager.ico")



def randompg(x):
    DIGITS = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    LOCASE_CHARACTERS = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
                         'u', 'v', 'w', 'x', 'y', 'z']
    UPCASE_CHARACTERS = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'M', 'N', 'O', 'P', 'Q', 'R',
                         'S', 'T','U', 'V', 'W', 'X', 'Y', 'Z']
    SYMBOLS = ['@', '#', '$', '%', '=', ':', '?', '.', '/', '|', '~', '>', '*', (', ')', '<']
    res = messagebox.askquestion("PASSWORD GENERATOR", "Do You Want to use special
                                 \n symbols such as @#$")
    if res == "yes":
        comblist = DIGITS + LOCASE_CHARACTERS +UPCASE_CHARACTERS + SYMBOLS
        maxlen = 12
        psswd = ""
        for i in range(maxlen):
            psswd += random.choice(comblist)
        print(psswd)
        x.insert(0, psswd)
```

```
    else:
        comblist = DIGITS + LOCASE_CHARACTERS + UPCASE_CHARACTERS
        maxlen = 12
        psswd = ""
        for i in range(maxlen):
            psswd += random.choice(comblist)
        print(psswd)
        x.insert(0, psswd)



def go_to_next_entry(event, entry_list, this_index):
    next_index = (this_index + 1) % len(entry_list)
    entry_list[next_index].focus_set()



def removewindow(r):
    r.destroy()



def hashingpassword(text):
    hash = hashlib.blake2b(text).hexdigest()
    return hash



def copy_from_treeviewuap(tree, event):
    selection = tree.selection()
```

```
    for each in selection:
        try:
            value = tree.item(each)["values"]
        except:
            pass
    copy_string = "\n".join(value[1:])
    pyperclip.copy(copy_string)



def copy_from_treeviewcol(tree, event):
    selection = tree.selection()
    column = tree.identify_column(event.x)
    column_no = int(column.replace("#", "")) - 1

    copy_values = []
    for each in selection:
        try:
            value = tree.item(each)["values"][column_no]
            copy_values.append(str(value))
        except:
            pass

    copy_string = "\n".join(copy_values)
    pyperclip.copy(copy_string)
```

```
def signup():
    root.title('SecureVault-SignUp')
    for widget in root.winfo_children():
        widget.destroy()
    global backButton, SignUpPage
    backButton = backButton.resize((35, 22))
    backButton = ImageTk.PhotoImage(backButton)
    SignUpPage = ImageTk.PhotoImage(SignUpPage)
    root.geometry("600x400+400+125")
    lblb = Label(root, image=SignUpPage)
    lblb.place(x=0, y=0)
    username1 = Entry(root, font=('BentonSans Comp Black', 14), justify='center',
                      bg='#A3A3A3', width=10)
    username1.place(x=257, y=121)
    password1 = Entry(root, width=10, show='*', font=('BentonSans Comp Black', 14),
                      justify='center', bg='#A3A3A3')
    password1.place(x=257, y=210)
    password2 = Entry(root, width=10, show='*', font=('BentonSans Comp Black', 14),
                      justify='center', bg='#A3A3A3')
    password2.place(x=257, y=290)
    lbl = Label(root, bg='#ED2124')
    lbl.place(x=245, y=15)
    entries = [child for child in root.winfo_children() if isinstance(child, Entry)]
```

```
for idx, entry in enumerate(entries):
    entry.bind('<Return>', lambda e, idx=idx: go_to_next_entry(e, entries, idx))


def credentialcheck():
    global funcstocall
    a = username1.get().lower()
    b = hashingpassword(password1.get().encode('utf-8'))
    c = hashingpassword(password2.get().encode('utf-8'))
    sqlFormula = 'SELECT * FROM login'
    mycursor.execute(sqlFormula)
    myresult = mycursor.fetchall()
    if myresult != []:
        for result in myresult:
            if result[0] != a:
                if b == c:
                    details1 = (a, b)
                    sqlFormula = "INSERT INTO login(username,password) VALUES(%s,%s)"
                    try:
                        mycursor.execute(sqlFormula, details1)
                        mycursor.execute(
                            "CREATE TABLE IF NOT EXISTS " + a + " (sitename VARCHAR(255),mailID
                             VARCHAR(255),password  VARCHAR(255))"
                    )
                        mydb.commit()
                        break
```

```
                except mysql.connector.errors.IntegrityError:
                    lbl["text"] = "INVALID USERNAME"
                    lbl['font'] = ('BentonSans Comp Black', 12)
                    break
            elif b != c:
                lbl["text"] = "PASSWORDS DON'T MATCH"
                lbl['font'] = ('BentonSans Comp Black', 12)
                break
        elif result[0] == a:
            lbl["text"] = "INVALID USERNAME"
            lbl['font'] = ('BentonSans Comp Black', 12)
            break
    elif myresult == []:
        if b == c:
            details1 = (a, b)
            sqlFormula = "INSERT INTO login(username, password) VALUES(%s,%s)"
            mycursor.execute(sqlFormula, details1)
            mycursor.execute(
                "CREATE TABLE IF NOT EXISTS " + a + " (sitename VARCHAR(255),mailID
                 VARCHAR(255),password VARCHAR(255))")
            mydb.commit()
button1 = Button(root, text="SIGN UP", command=lambda: [credentialcheck()],
                         fg='White', bg='#ed2024',font=('BentonSans Comp Black', 14))
button1.place(x=275, y=330)
```

```
    button3 = Button(root, command=lambda: {start_menu()}, image=backButton)
    button3.place(x=30, y=35)



def login():
    images()
    global backButton
    backButton = backButton.resize((35, 22))
    backButton = ImageTk.PhotoImage(backButton)
    j = 1
    for widget in root.winfo_children():
        widget.destroy()
    root.geometry("800x600+275+50")
    root.title("SecureVault Password Manager")
    img = Image.open(r"D:\Aravind\Class 12th Project\LoginPage-01.png")
    img = img.resize((800, 600))
    labl = Label(root)
    labl.img = ImageTk.PhotoImage(img)
    labl['image'] = labl.img
    labl.place(x=0, y=0)
    username1 = Entry(root, width=12, font=("BentonSans Comp Black", 18),
                                justify='center')
    username1.place(x=325, y=243)
    password1 = Entry(root, width=12, show='*', font=( "BentonSans Comp Black", 18),
                                justify='center')
    password1.place(x=325, y=347)
```

```
entries = [child for child in root.winfo_children() ifisinstance(child, Entry)]
for idx, entry in enumerate(entries):
    entry.bind('<Return>', lambda e, idx=idx: go_to_next_entry(e, entries, idx))
button3 = Button(root, command=lambda: {start_menu()}, image=backButton)
button3.place(x=30, y=35)

def forgotpassword():
    global forgotPage
    forgotPage = ImageTk.PhotoImage(forgotPage)
    root4 = Toplevel(root)
    root4.geometry('400x300+470+150')
    root4.grab_set()
    lbl1 = Label(root4, image=forgotPage)
    lbl1.place(x=0, y=0)
    enrt = Entry(root4, width=16, font=('BentonSans Comp Black', 12), justify='center')
    enrt.place(x=140, y=70)
    enrt1 = Entry(root4, width=16, font=('BentonSans Comp Black', 12), justify='center')
    enrt1.place(x=140, y=145)
    enrt2 = Entry(root4, width=16, font=('BentonSans Comp Black', 12), justify='center')
    enrt2.place(x=140, y=220)
    enrt1['state'] = 'disabled'
    enrt2['state'] = 'disabled'
    def chk():
        chk1 = enrt.get().lower()
        enrt1['state'] = 'normal'
        enrt2['state'] = 'normal'
        mycursor.execute("SELECT * FROM login")
```

```
myresult = mycursor.fetchall()
for result in myresult:
    if result[0] == chk1:
        btn5.destroy()
        entries = [child for child in root4.winfo_children() if isinstance(child, Entry)]
        for idx, entry in enumerate(entries):
            entry.bind('<Return>', lambda e, idx=idx: go_to_next_entry(e, entries, idx))

        def chkfinal():
            nonlocal chk1
            q = enrt1.get()
            chk1 = """ + chk1 + """
            r = enrt2.get()
            if q == r:
                q = hashingpassword(q.encode('utf-8'))
                q = """ + q + """
                sqlFormula = f"UPDATE login SET password={q} WHERE username={chk1}"
                mycursor.execute(sqlFormula)
                mydb.commit()
        btsn = Button(root4, text="DONE",
command=lambda: [chkfinal(), removewindow(root4)],
                              font=('BentonSans Comp Black', 12))
        btsn.place(x=180, y=255)
        break
    else:
        messagebox.showerror("ERROR", "ENTER A  VALID USERNAME")
        break
```

```
    btn5 = Button(root4, text="Check", command=chk,
                                font=('BentonSans Comp Black', 10))
    btn5.place(x=300, y=69)

butn9 = Button(root, text="FORGOT PASSWORD", command=forgotpassword,
                        bg="#820814",font=("BentonSans Comp Black", 12), fg="White",
                        width=17)
butn9.place(x=340, y=525)

def credentialcheck():
    username = username1.get().lower()
    password = hashingpassword(password1.get().encode('utf-8'))
    det = (username, password)
    mycursor.execute("SELECT * FROM login")
    myresult = mycursor.fetchall()
    global showPassword, hidePassword, mainPage, home1
    for result in myresult:
        if det == result:
            for widget in root.winfo_children():
                widget.destroy()
            mainPage = ImageTk.PhotoImage(mainPage)
            label1 = Label(root, image=mainPage)
            label1.place(x=0, y=0)
            label2 = Label(root, text="Welcome!", font=("American Captain", 40),
                                bg='#ed2124', fg='Black')
            label2.place(x=330, y=25)
```

```
mycursor.execute('SELECT * FROM ' + username)
myresult = mycursor.fetchall()
global mytree
mytree = ttk.Treeview(root)
mytree['columns'] = ("Site Name", "Mail ID", "Password")
mytree.column("#0", width=0, minwidth=25)
mytree.column("Site Name", anchor=CENTER, width=200)
mytree.column("Mail ID", anchor=CENTER, width=200)
mytree.column("Password", anchor=CENTER, width=200)
mytree.heading("#0", text=" ", anchor=CENTER)
mytree.heading("Site Name", text="Site Name", anchor=CENTER)
mytree.heading("Mail ID", text="Mail ID",  anchor=CENTER)
mytree.heading("Password", text="Password",  anchor=CENTER)
i = 0
for result in myresult:
    result = list(result)
    result[2] = '*' * len(result[2])
    mytree.insert(parent="", index='end', iid=i,
        text="", values=result)
    i += 1
mytree.place(x=95, y=210)
mytree.bind("<Control-Key-c>", lambda x: copy_from_treeviewuap(mytree, x))
mytree.bind('<Control-Key-x>', lambda x:  copy_from_treeviewcol(mytree, x))
home1 = ImageTk.PhotoImage(home1)
button3 = Button(root, command=lambda: {start_menu()}, image=home1)
button3.place(x=25, y=15)
```

```
button4 = Button(root, command=lambda: {login()}, image=backButton)
button4.place(x=75, y=15)
showPassword = ImageTk.PhotoImage(
                showPassword)
hidePassword = ImageTk.PhotoImage(
                hidePassword)

def hidepassword():
    global mytree
    nonlocal username
    mycursor.execute('SELECT * FROM ' + username)
    myresult = mycursor.fetchall()
    mytree.destroy()
    mytree = ttk.Treeview(root)
    mytree['columns'] = ("Site Name", "Mail ID", "Password")
    mytree.column("#0", width=0, minwidth=25)
    mytree.column("Site Name", anchor=CENTER, width=200)
    mytree.column("Mail ID", anchor=CENTER, width=200)
    mytree.column("Password", anchor=CENTER, width=200)
    mytree.heading("#0", text=" ", anchor=CENTER)
    mytree.heading("Site Name", text="Site Name", anchor=CENTER)
    mytree.heading("Mail ID", text="Mail ID", anchor=CENTER)
```

```
    mytree.heading("Password", text="Password", anchor=CENTER)
    i = 0
    for result in myresult:
        result = list(result)
        result[2] = '*' * len(result[2])
        mytree.insert(parent="", index='end', iid=i,  text="", values=result)
        i += 1
    mytree.place(x=95, y=210)
    mytree.bind("<Control-Key-c>", lambda x: copy_from_treeviewuap(mytree, x))
    mytree.bind('<Control-Key-x>', lambda x: copy_from_treeviewcol(mytree, x))
    btn5['state'] = 'disabled'
    btn4['state'] = 'active'

def showpassword():
    mycursor.execute('SELECT * FROM ' + username)
    myresult = mycursor.fetchall()
    global mytree
    mytree.destroy()
    mytree = ttk.Treeview(root)
    mytree['columns'] = ("Site Name", "Mail ID", "Password")
    mytree.column("#0", width=0, minwidth=25)
```

```
    mytree.column("Site Name", anchor=CENTER, width=200)
    mytree.column("Mail ID", anchor=CENTER, width=200)
    mytree.column("Password", anchor=CENTER, width=200)
    mytree.heading("#0", text=" ", anchor=CENTER)
    mytree.heading("Site Name", text="Site Name", anchor=CENTER)
    mytree.heading("Mail ID", text="Mail ID", anchor=CENTER)
    mytree.heading("Password", text="Password", anchor=CENTER)
    i = 0
    for result in myresult:
        mytree.insert(parent="", index='end', iid=i, text="", values=result)
        i += 1
    mytree.place(x=95, y=210)
    mytree.bind("<Control-Key-c>", lambda x: copy_from_treeviewuap(mytree, x))
    mytree.bind('<Control-Key-x>', lambda x: copy_from_treeviewcol(mytree, x))
    btn4['state'] = 'disabled'
    btn5['state'] = 'active'

btn4 = Button(root, image=showPassword, command=showpassword)
btn4.place(x=730, y=250)
btn5 = Button(root, image=hidePassword, command=hidepassword)
btn5.place(x=730, y=290)
btn5['state'] = 'disabled'
```

```
def refresh():
    global passwords
    global mytree
    mytree.destroy()
    mytree = ttk.Treeview(root)
    if btn5['state'] == 'disabled':
        mytree['columns'] = ("Site Name", "Mail ID", "Password")
        mytree.column("#0", width=0, minwidth=25)
        mytree.column("Site Name", anchor=CENTER, width=200)
        mytree.column("Mail ID", anchor=CENTER,  width=200)
        mytree.column("Password", anchor=CENTER,
        width=200)
        mytree.heading("#0", text=" ", anchor=CENTER)
        mytree.heading("Site Name", text="Site Name", anchor=CENTER)
        mytree.heading("Mail ID", text="Mail ID", anchor=CENTER)
        mytree.heading("Password", text="Password", anchor=CENTER)
        mycursor.execute('SELECT * FROM ' + username)
        myresult = mycursor.fetchall()
        i = 0
        for result in myresult:
            result = list(result)
            result[2] = '*' * len(result[2])
            mytree.insert(parent="", index='end', iid=i, text="", values=result)
            i += 1
```

```
    mytree.place(x=95, y=210)
    mytree.bind("<Control-Key-c>", lambda x:copy_from_treeviewuap(
                        mytree, x))
    mytree.bind('<Control-Key-x>', lambda x: copy_from_treeviewcol(mytree, x))
else:
    mytree['columns'] = ("Site Name", "Mail ID", "Password")
    mytree.column("#0", width=0, minwidth=25)
   mytree.column("Site Name", anchor=CENTER, width=200)
    mytree.column("Mail ID", anchor=CENTER, width=200)
    mytree.column("Password", anchor=CENTER, width=200)
    mytree.heading("#0", text=" ", anchor=CENTER)
    mytree.heading("Site Name", text="Site Name", anchor=CENTER)
    mytree.heading("Mail ID", text="Mail ID", anchor=CENTER)
    mytree.heading("Password", text="Password", anchor=CENTER)
    mycursor.execute('SELECT * FROM ' + username)
    myresult = mycursor.fetchall()
    i = 0
    for result in myresult:
        mytree.insert(parent="", index='end', iid=i,
    text="", values=result)
        i += 1
    mytree.place(x=95, y=210)
    mytree.bind("<Control-Key-c>", lambda x: copy_from_treeviewuap(
                        mytree, x))
```

```
    mytree.bind('<Control-Key-x>', lambda x: copy_from_treeviewcol(mytree, x))

butn2 = Button(root, text="REFRESH", command=refresh, width=20, font=
                        ("Agency FB", 15), bg='#cec4c5')
butn2.place(x=230, y=110)

def add_pword():
  global addPword
  root1 = Toplevel(root)
  root1.geometry("420x320+472+215")
  root1.grab_set()
  addPword = Image.open(r'AddPword-01.png')
  addPword = addPword.resize((420, 320))
  addPword = ImageTk.PhotoImage(addPword)
  lab1 = Label(root1, image=addPword)
  lab1.place(x=0, y=0)
  sitename = Entry(root1, width=16, font=("BentonSans Comp Black", 9),
                        justify='center')
  sitename.place(x=165, y=70)
  mail = Entry(root1, width=16, font=("BentonSans Comp Black", 9),
                        justify='center')
  mail.place(x=165, y=135)
  password2 = Entry(root1, width=16, font=("BentonSans Comp Black", 9),
                       justify='center')
  password2.place(x=165, y=195)
```

```
entries = [child for child in root1.winfo_children() if isinstance(child, Entry)]
for idx, entry in enumerate(entries):
    entry.bind('<Return>', lambda e, idx=idx: go_to_next_entry(e, entries, idx))


def credadd():
    a1 = ''
    nonlocal j
    a = sitename.get()
    b = mail.get()
    c = password2.get()
    det = (a, b, c)
    sqlFormula = f"SELECT * FROM {username}"
    mycursor.execute(sqlFormula)
    myresult = mycursor.fetchall()
    for result in myresult:
        if result[0] == a and result[0][-1].isdigit() is
    False:
            a1 = f'{j}'
            j += 1
        elif result[0] == a and result[0][-2].isdigit():
            j = int(result[0][-1]) + 1
            a1 = f'{j}'
    a = a + a1
    det = (a, b, c)
```

```
        if det != ('', '', ''):
            sqlFormula = "INSERT INTO " + username + " (sitename,mailID,password)
                                VALUES(%s,%s,%s)"
            mycursor.execute(sqlFormula, det)
            mydb.commit()
        else:
            pass

    btnn = Button(root1, text="Generate Random  Password", command=lambda:
                                        [randompg(password2)],
                font=("BentonSans Comp Black", 10), bg='Black', fg='White')
    btnn.place(x=145, y=230)
    buton1 = Button(root1, text="DONE", command=lambda: [credadd(),
                        refresh(), removewindow(root1)], font=("BentonSans Comp
                        Black", 10), bg='#820814', fg='White')
    buton1.place(x=198, y=260)

butn1 = Button(root, text="ADD PASSWORD",command=add_pword, width=20,
                        font=("Agency FB", 15), bg='#cec4c5')
butn1.place(x=60, y=110)

def update_pword():
    global updatePword
    root2 = Toplevel(root)
    updatePword = Image.open(r'updatePage-01-01.png')
    updatePword = updatePword.resize((400, 300))
    root2.geometry('400x300+472+215')
```

```
root2.grab_set()
l = []
sqlFormula = f'SELECT * FROM {username}'
mycursor.execute(sqlFormula)
myresult = mycursor.fetchall()
for result in myresult:
    l.append(result[0])
b = StringVar()
updatePword = ImageTk.PhotoImage(updatePword)
lbls1 = Label(root2, image=updatePword)
lbls1.place(x=0, y=0)
menu1 = ttk.Combobox(root2, textvariable=b,  values=l, font=
                ('BentonSans Comp Black', 8))
menu1.place(x=130, y=60)
ent4 = Entry(root2, width=20, font=('BentonSans Comp Black', 10),
                    justify='center')
ent4.place(x=130, y=125)
ent6 = Entry(root2, width=20, font=('BentonSans Comp Black', 10),
                      justify='center')
ent6.place(x=130, y=185)
entries = [child for child in root2.winfo_children() if isinstance(child, Entry)]
for idx, entry in enumerate(entries):
    entry.bind('<Return>', lambda e, idx=idx: go_to_next_entry(e, entries, idx))
```

```
def update1():
    a = b.get()
    a = """ + a + """
    d = ent4.get()
    d = """ + d + """
    f = ent6.get()
    f = """ + f + """
    sqlFormula1 = ("UPDATE " + username +
    " SET mailID=" + d +" WHERE sitename=" + a)
    sqlFormula2 = ("UPDATE " + username +
    " SET password=" + f + " WHERE sitename=" + a)
    mycursor.execute(sqlFormula1)
    mycursor.execute(sqlFormula2)
    mydb.commit()
btnn = Button(root2, text="Generate Random  Password", command=lambda:
                    [randompg(ent6)],font=('BentonSans Comp Black', 10))
btnn.place(x=135, y=225)
btn1 = Button(root2, text="DONE",command=lambda: [update1(),refresh(),
                    removewindow(root2)],  font=('BentonSans Comp Black', 10))
btn1.place(x=180, y=260)

butn3 = Button(root, text="UPDATE", command=update_pword, width=20,
                    font=("Agency FB", 15), bg='#cec4c5')
butn3.place(x=400, y=110)
```

```
def confirm(x, y):
    res = messagebox.askquestion("DELETE SITE", "Do You Want to Remove The
                                 Selected Site")
    if res == "yes":
        x()
        refresh()
        y.destroy()
    else:
        pass
def delete_pword():
    global deletePword
    root3 = Toplevel(root)
    root3.geometry('300x200+540+280')
    root3.grab_set()
    l = []
    sqlFormula = f'SELECT * FROM {username}'
    mycursor.execute(sqlFormula)
    myresult = mycursor.fetchall()
    for result in myresult:
        l.append(result[0])
    deletePword = Image.open(r'deletePword-01.png')
    deletePword = deletePword.resize((300, 200))
    deletePword = ImageTk.PhotoImage(deletePword)
    lbl = Label(root3, image=deletePword)
    lbl.place(x=0, y=0)
```

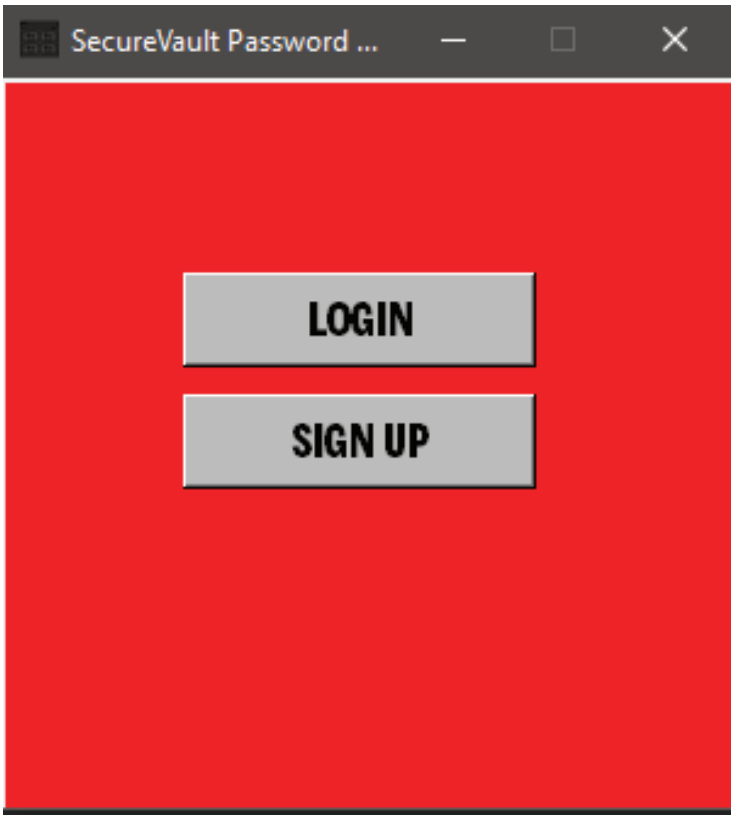
```
    b = StringVar()
    menu1 = ttk.OptionMenu(root3, b, 'SELECT ', *l)
    menu1.place(x=105, y=80)
    entries = [child for child in root3.winfo_children() if isinstance(child, Entry)]
    for idx, entry in enumerate(entries):
        entry.bind('<Return>', lambda e, idx=idx: go_to_next_entry(e, entries, idx))
    def delete2():
        a = b.get()
        a = """ + a + """
        sqlFormula = f"DELETE FROM {username} WHERE sitename={a}"
        mycursor.execute(sqlFormula)
        mydb.commit()

    btn1 = Button(root3, text="DONE", command=lambda: [confirm(delete2,
                                root3)], font=("BentonSans Comp Black", 15))
    btn1.place(x=130, y=150)

btn1 = Button(root, text="DELETE", command=delete_pword, width=20,
                                font=("Agency FB", 15), bg='#cec4c5')
btn1.place(x=570, y=110)
def savefiles():
    import csv
    data = []
    file2 = filedialog.asksaveasfilename( defaultextension='.csv',
                                 filetypes=[  ("Comma Separated Values", '.csv'),
                                ("Excel Sheet", '.xlsx')])
```

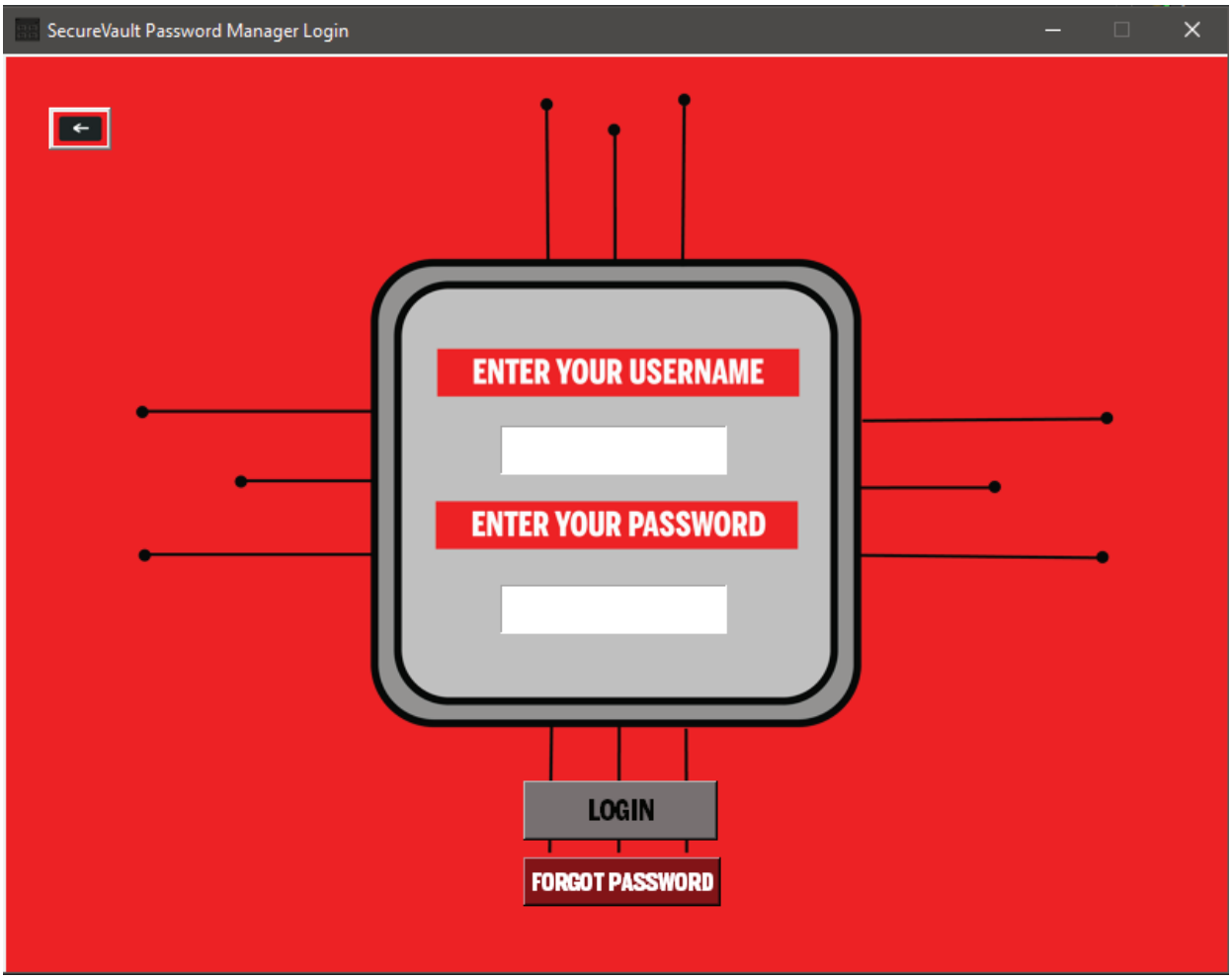
```
    if file2 != None or file2 != '':
        import os
        file2 = os.getcwd()
        file2 += f'\\{username}.csv'
        file1 = open(file2, 'w', newline='')
        writetofile = csv.writer(file1)
        mycursor.execute("SELECT * FROM " +username)
        myresult = mycursor.fetchall()
        for result in myresult:
            print(result)
            writetofile.writerow(result)
        file1.close()
    else:
        pass

btn2 = Button(root, text="EXPORT", command=savefiles, width=20,
                                font=("Agency FB", 12), bg='#cec4c5')
btn2.place(x=670, y=50)
def importfromcsvfile():
    import csv
    file = filedialog.askopenfilename( defaultextension='.csv',filetypes=[("Comma
                                        Separated Values", '.csv')])
    if file is not None and file != '':
        file1 = open(file, 'r', newline='')
        readed = csv.reader(file1)
        sqlformula = "INSERT INTO " + username + " (sitename,mailID,password)
                        VALUES(%s,%s,%s)"
```

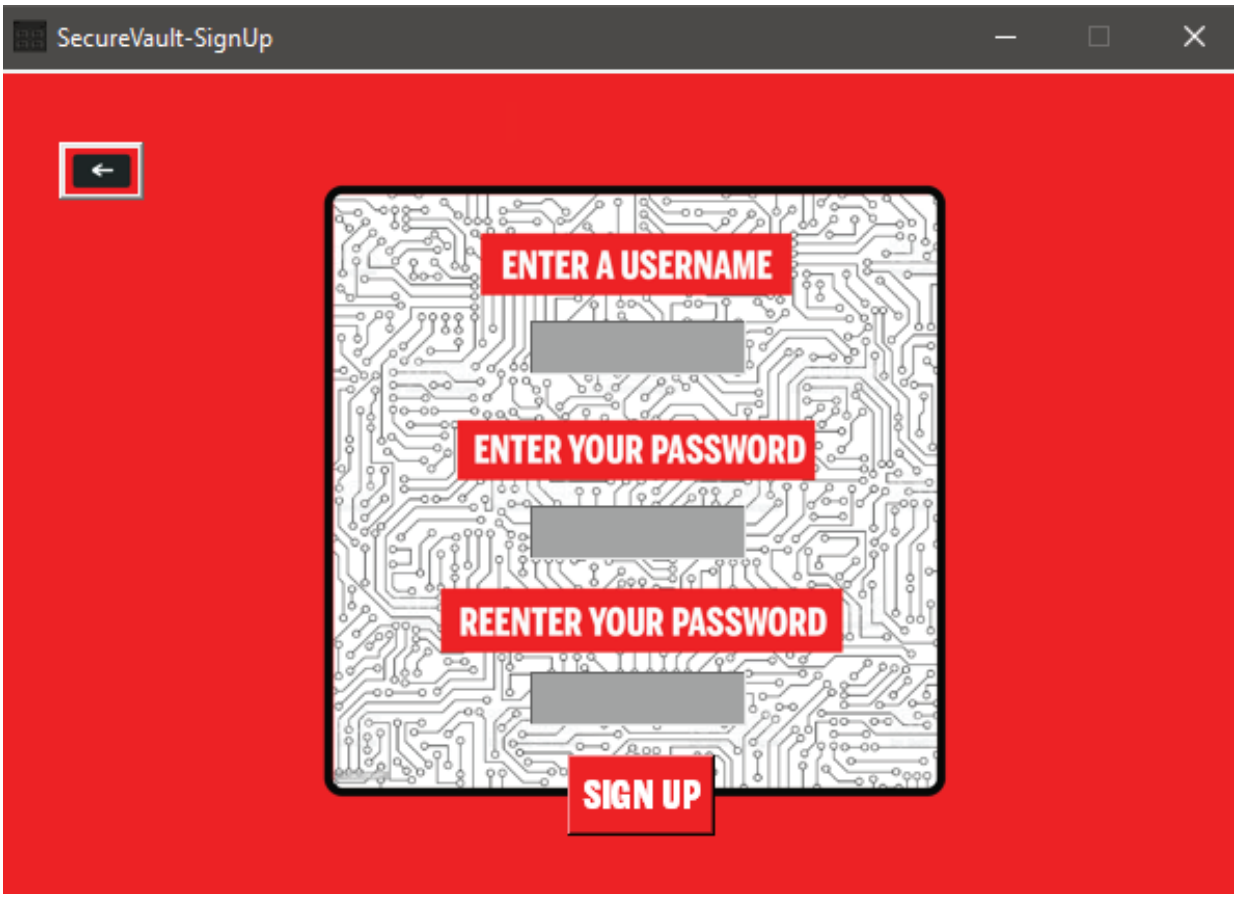
```
            for i in readed:
                mycursor.execute(sqlformula, i)
                mydb.commit()
            file1.close()
        else:
            pass

    btn3 = Button(root, text='IMPORT', command=lambda: [importfromcsvfile(),
                refresh()], width=20, font=('Agency FB', 12), bg='#cec4c5')
    btn3.place(x=20, y=60)

button1 = Button(root, text="LOGIN", bg="#777071", font=("BentonSans Comp Black",
                14), fg="Black",command=credentialcheck, width=13)
button1.place(x=340, y=475)
def start_menu():
    images()
    global mainPage1
    if root:
        for widget in root.winfo_children():
            widget.destroy()
        else:
            pass
    root.geometry("300x300+520+220")
    root.iconbitmap(r"password-manager.ico")
    root.title('SecureVault Password Manager')
    mainPage1 = ImageTk.PhotoImage(mainPage1)
    lbl = Label(root, image=mainPage1)
```

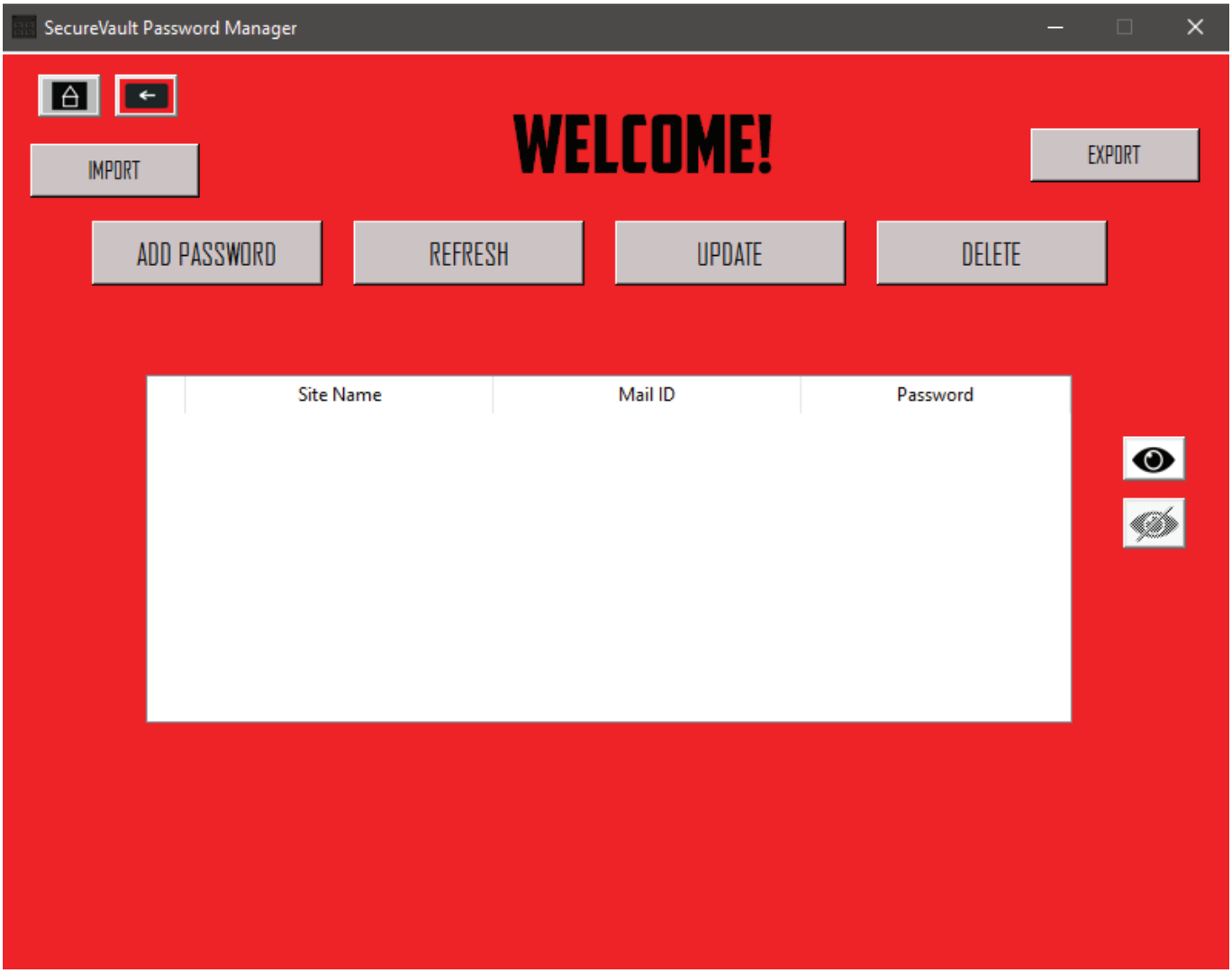
```
    lbl.place(x=0, y=0)
    button2 = Button(text="LOGIN", width=15, command=login, font=('BentonSans Comp
                                Black', 14), bg='#bcbcbc',fg='Black')
    button2.place(x=75, y=80)
    button1 = Button(text="SIGN UP", width=15, command=signup,
                        font=('BentonSans Comp Black', 14), bg='#bcbcbc',fg='Black')
    button1.place(x=75, y=130)

start_menu()
root.resizable(False, False)
root.mainloop()
```

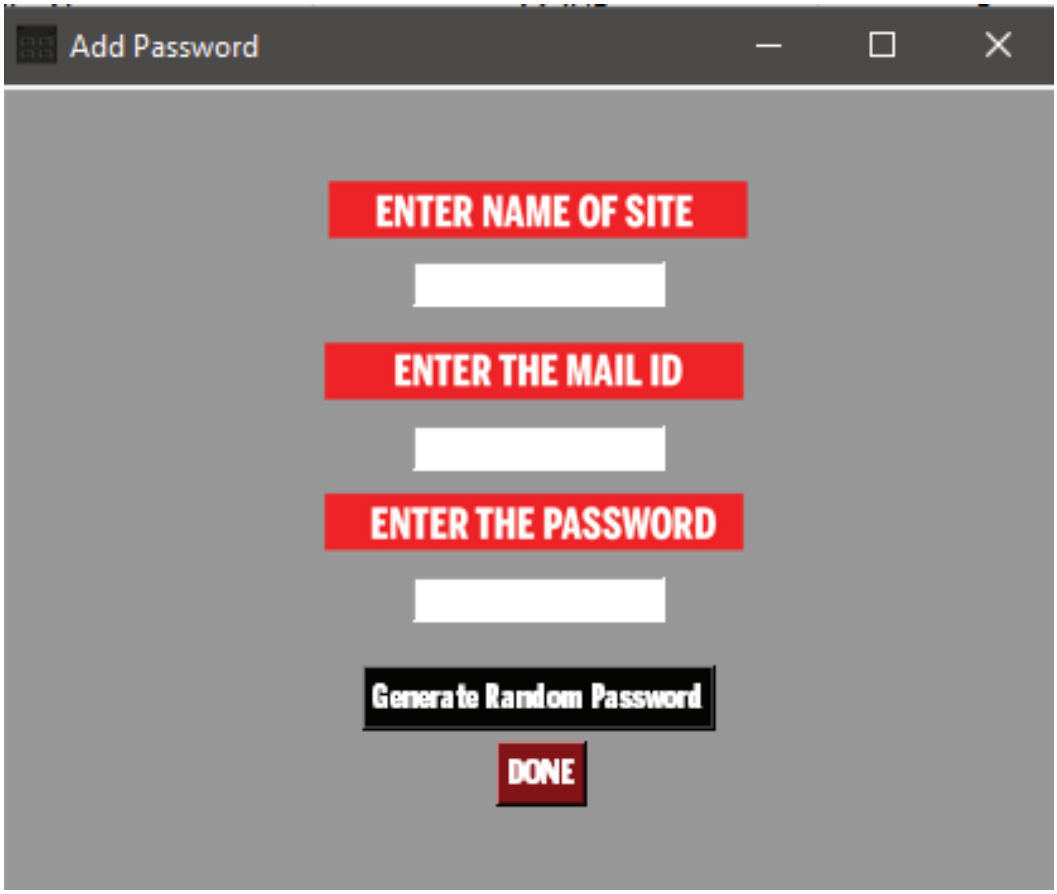
SNAP SHOTS

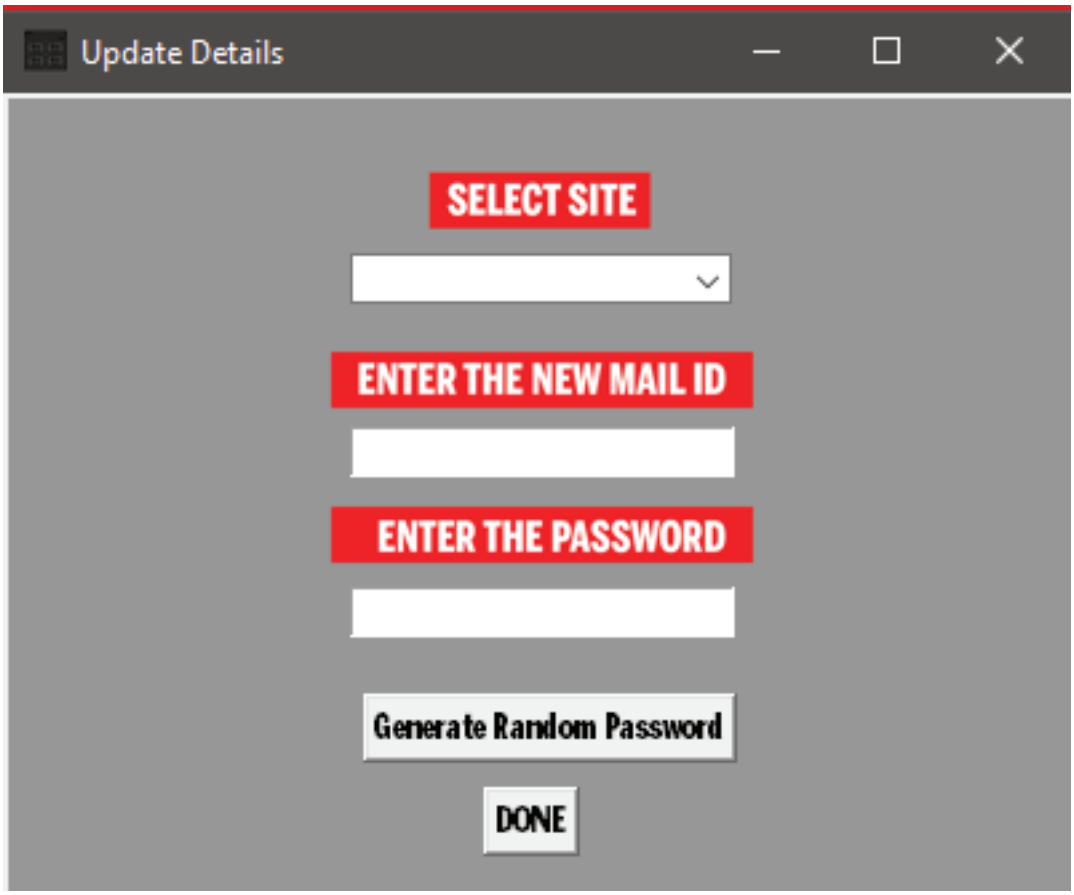

SecureVault Password ...
LOGIN
SIGN UP

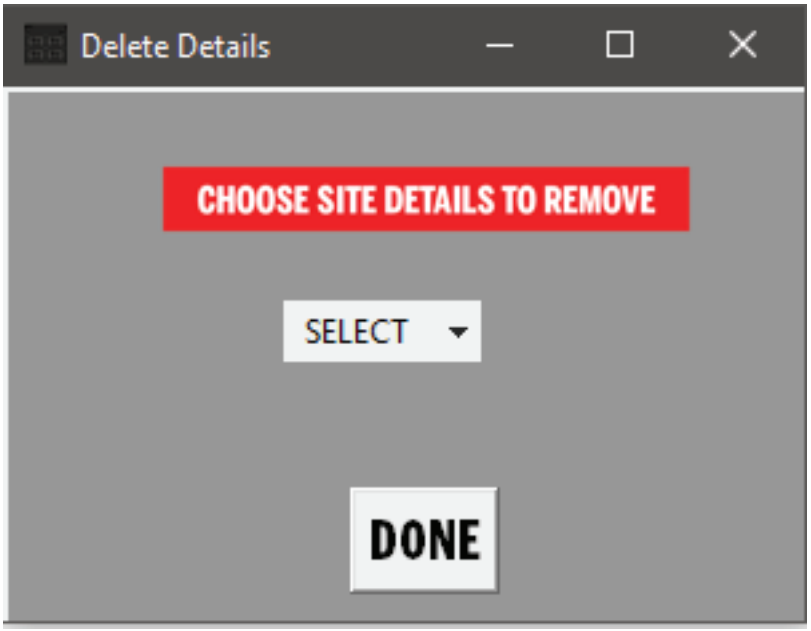

SecureVault Password Manager Login
ENTER YOUR USERNAME
ENTER YOUR PASSWORD
LOGIN
FORGOT PASSWORD

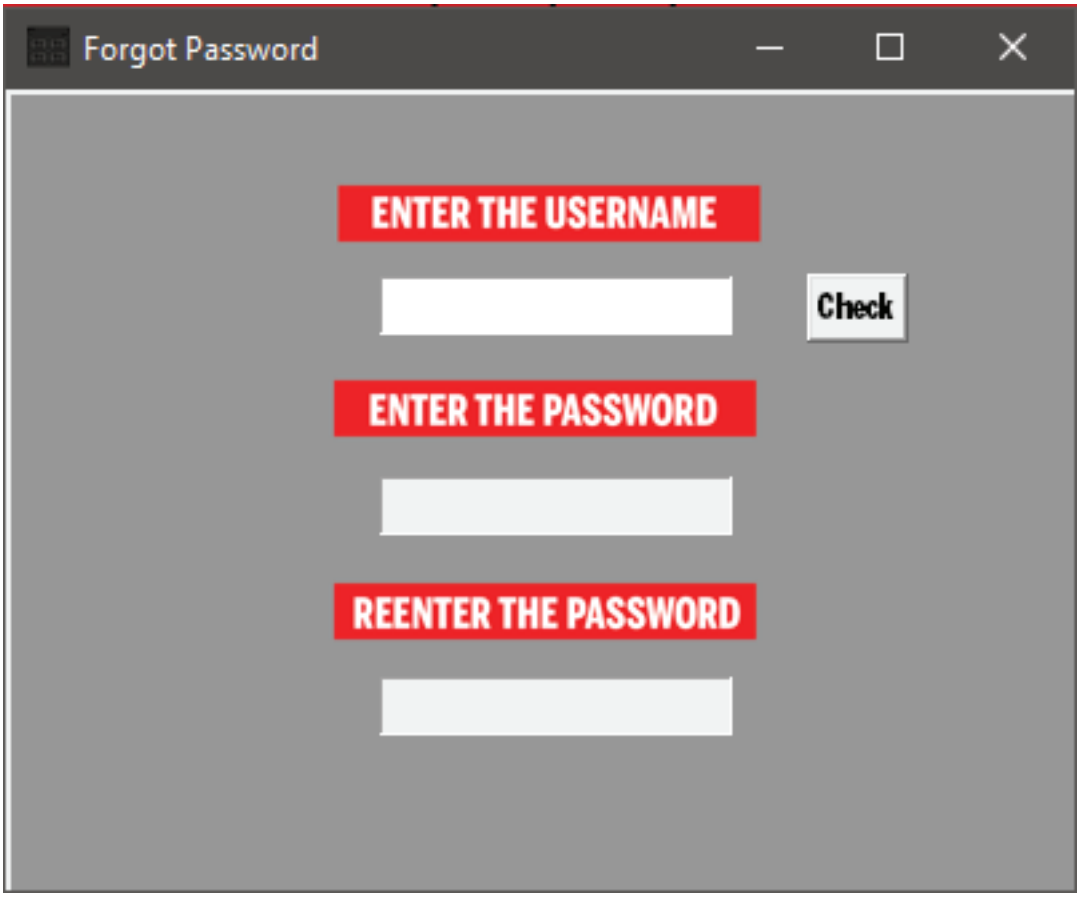

SecureVault-SignUp
ENTER A USERNAME
ENTER YOUR PASSWORD
REENTER YOUR PASSWORD
SIGN UP


SecureVault Password Manager
WELCOME!
IMPORT
EXPORT
ADD PASSWORD
REFRESH
UPDATE
DELETE
Site Name
Mail ID
Password


Add Password
ENTER NAME OF SITE
ENTER THE MAIL ID
ENTER THE PASSWORD
Generate Random Password
DONE


Update Details
SELECT SITE
ENTER THE NEW MAIL ID
ENTER THE PASSWORD
Generate Random Password
DONE


Delete Details
CHOOSE SITE DETAILS TO REMOVE
SELECT
DONE


Forgot Password
ENTER THE USERNAME
Check
ENTER THE PASSWORD
REENTER THE PASSWORD

FUTURE ENHANCEMENTS

I have tried my level best to incorporate most of the main features which all the famous Password Managers have. Yet there can be some improvements which can improve the application.

- The App can be integrated with browsers via browser extensions and therefore make it seamless for user to access his login details.
- The app can deployed to servers therefore making it easier for users to access their login details anywhere.
- The app can be optimised for multiple Operating systems such as Android, iOS, Linux, macOS making it easier to run the app on any platform.
- The App can be fitted with better images and transitions, making it more visually attractive.

BIBLIOGRAPHY

- GeeksForGeeks.com
- StackOverflow.com
- Youtube.com
- Tkinter Free Ebook: riptutorial.com
- Python Docs
- Computer Science With Python - Sumita Arora Class 11th and 12th
- JavaTPoint.com
- w3Schools.in
- GitHub.com