National Public School, Rajajinagar

Password Manager

Tarun K S
Mayur G
Class XII A
Roll No.

Table of Contents

Certificate

Department Of Computer Science

This is to certify that this project work entitled *Password Manager* is an original work carried out by Tarun K S and Mayur G during the year 2020-2021. No part of this work has been submitted elsewhere partially or fully. Any material reproduced in this project has been properly acknowledged.

The project has been approved as it satisfies academic requirement in respect of project work prescribed by CBSE.

Student Names : &

Reg No: &
Internal examiner: _____

External examiner: _____

Date : _____

Examination Center: National Public School, Rajajinagar, BLR

Acknowledgement

We sincerely thank CBSE for conducting this project and our school for giving us this opportunity. We are grateful to our parents for their support, our principal Ms. Malathy N and all our teachers for their help. We greatly appreciate the efforts of our computer teachers, Ms.Bhargavi, Ms.Poornima, Ms.Tanushree in guiding and helping the students while ensuring a fun and enriching experience.

About Python

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Designed by: Guido van Rossum

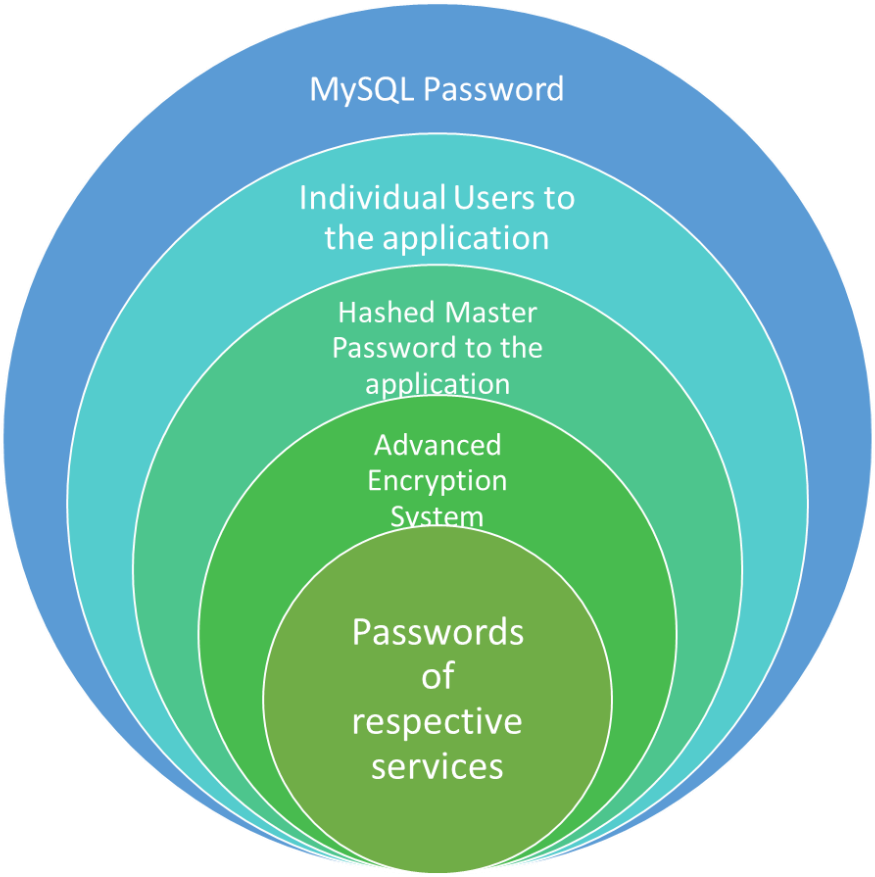
Developer: Python Software Foundation

DETAILS OF PROJECT

INTRODUCTION:

This application is developed to help all the privacy focused users of the internet and its services. A Password Manager or your 1-stop to access all the passwords that trying to remember turns into a real burden for all. It will be hosted locally on every user's machine using the MySQL Database. Each password entered into the application will be encrypted using Advanced Encryption System (AES). Each user is required to remember only 1 MASTER PASSWORD (which is hashed using MD5 Hashing technique to ensure maximum security) to enter into the application.

This application has been equipped with 4 layers of protection so as to keep all your credentials private.



BEGINNER'S GUIDE:

1. Run the main_code.py file in your preferred IDLE or IDE.
2. The Start Screen displays 3 buttons: Start console, End Console, Delete User.
3. Click on Start Console.
4. The Main Screen displays 3 buttons: New User, About Us, Existing User
5. Click on New User and fill in the detailed (**PLEASE REMEMBER YOUR MASTER PASSWORD**)
6. Close the current window and return to the Main Screen. Click on Existing User and sign in with the credentials that you created.
7. The window will prompt you to add passwords, click on Add passwords button. Enter the service name, E-mail id connected to the service and its password. Click on the Save Button.
8. Returning to the previous window, you will have a table displaying the service name. Click on it and click on Show Passwords to view its password and Email id in the next window.
9. If you want to edit the records in the application Click on Edit Password. This will display a new window with two buttons: Update Existing Password and Delete Existing Password.
10. Updating the Password will require you to confirm the old password, email id and the new password.
11. Deleting Existing Password will require you to confirm the service name and the email id.
12. You can sign out of the application in the previous window.
13. Deleting the user will trash out all his/her records on the machine so please confirm twice before attempting to.
14. For more information, Click on the About Us Button.

Working:

FRONTEND:

All graphics used in the frontend were custom designed using Adobe Photoshop. (Buttons)
The Main Windows are designed with the help of the GUI (Graphical User Interface) Library which comes preinstalled with Python - Tkinter

The Images are Placed onto the Labels by first opening and reading them into a image object with the help of Python's Image Processing library -PIL (Pillow)

Message Boxes (Confirm, Showing error, showing info) is from tkinter.messagebox module

Few special Elements not available in Tkinter (Listbox used to show services and table used to show all the details) is taken from Tkinter.ttk

The Tkinter.ttk is a module which provides access to the Tk themed widget set, introduced in Tk 8.5

Multiple Windows are assigned to buttons. Function Toplevel() is used for this. On press of a button the previous windows is moved and a new one with the data is presented

The GUI features 2 special buttons, one to copy password to computer's clipboard to make it easy to sign in and another to clear the clipboard after usage to ensure no one can steal the passwords

BACKEND:
Using MySQL Connector all the data input by tkinter is fed into the MySQL database. Users must ensure valid credentials are entered to get the application to connect with the database.
The master password is stored after MD5 Hashing while the individual passwords are encrypted using AES and stored. Users but make sure they remember their master password

Table structure:
A master table is used to store the name and master password of each user of the application.
Every user of the application gets his/her own table to store the name of the service, Email-Id, password and the encryption key.

How is the encryption of the passwords done?
Ans. Using a key value (the user's master password) the passwords entered are encrypted and decrypted using Advanced Encryption system. Without the key the passwords inside each user's table are just random strings.

Modules Used:

1. Tkinter
2. Tkinter.ttk
3. Tkinter.messagebox
4. Hashlib
5. Re
6. Pyperclip
7. Secure
8. PIL
9. MySQL Connector

SOURCE CODE

```python
import tkinter as tk
import tkinter.ttk as ttk
from tkinter import messagebox
from tkinter import *
from PIL import ImageTk,Image
import mysql.connector as mysql
from secure import retrieve,save,modify,delete
import re
import pyperclip
import hashlib



# saving master password for new user
def add_master(name,password):
    salt = 'LTC'
    passwd = password[0:1] + salt + password[1:]
    res=hashlib.md5(bytes(passwd,'utf-8')).hexdigest()

    mydb = mysql.connect(user='root',passwd='root',host='localhost',database='master_password')
    if mydb.is_connected:
        mycur = mydb.cursor()
        try:
            query = 'INSERT INTO password(name,password) VALUES(%s,%s)'
            values = (name,res)
            mycur.execute(query,values)
            make_table(name)
            mydb.commit()
        except Exception as e:
            mydb.rollback()
            messagebox.showerror('Error','User Already Exists, Please sign in')

    mydb.close()

# making table for new user
def make_table(name):
    mydb = mysql.connect(user='root',passwd='root',host='localhost',database='master_password')
    if mydb.is_connected:
        mycur = mydb.cursor()
        try:
            mycur.execute('CREATE TABLE {} (email varchar(225),password BLOB, service varchar(225),tkey varchar(225));'.format(name))
            mydb.commit()
        except Exception as e:
            messagebox.showerror('Error','User could not be added Try later')
            mydb.rollback()
    mydb.close()




# getting all master passwords
def get_master():
    mydb = mysql.connect(user='root',passwd='root',host='localhost',database='master_password')
    if mydb.is_connected:
        mycur = mydb.cursor()
        mycur.execute('SELECT * from password')
        passwords = mycur.fetchall()
    mydb.close()
```

```
59
60      return passwords
61
62  # clearing entries
63  def forget_newuser():
64      name_entry.delete(0,END)
65      master_entry.delete(0,END)
66      masterconfi_entry.delete(0,END)
67
68  # clear screen
69  def clear():
70      item_list = exist_win.winfo_children()
71      for widget in item_list:
72          widget.destroy()
73
74  # getting existing users passwords
75  def get_exist_pass(username):
76      mydb = mysql.connect(user='root',passwd='root',host='localhost',database='
    master_password')
77      if mydb.is_connected:
78          mycur = mydb.cursor()
79          mycur.execute('select * from {}'.format(username))
80          exist_passwords = mycur.fetchall()
81      mydb.close()
82
83      return exist_passwords
84
85  # getting passwords based on service
86
87  def get_pass_by_service():
88      selected_service = service_listbox.get(tk.ANCHOR)
89      passwords_by_service = retrieve.get_passwd(username,selected_service)
90
91      return passwords_by_service
92
93
94
95
96  # displaying existing password for that service
97  def display():
98      global trv
99      passwords_by_services = get_pass_by_service()
100     clear()
101     style = ttk.Style(exist_win)
102
103     # setting ttk theme to "clam" which support the fieldbackground option
104     style.theme_use("clam")
105
106     ttk.Style().configure("Treeview", background="black",foreground="white",
    fieldbackground="black")
107     trv = ttk.Treeview(exist_win)
108     # setting columns
109     trv['columns'] = ('Email','Password','Service')
110
111     # formatting columns
112     # ! #0 is phantom column given by treeview
113     trv.column('#0',width=0,minwidth=0)
114     trv.column('Email',anchor=W,width=120)
115     trv.column('Password',anchor=CENTER,width=120)
116     trv.column('Service',anchor=W,width=120)
117
118
```

```
119     trv.heading('#0',text='')
120     trv.heading('Email',text='Email')
121     trv.heading('Password',text='Password')
122     trv.heading('Service',text='Service')
123 
124     trv.pack()
125 
126     for row in passwords_by_services:
127         trv.insert(parent='',index='end',values=row)
128 
129     # ? calling copying function
130     copy_button = ttk.Button(exist_win,text='Copy Password',command=select)
131     copy_button.pack(pady=10)
132 
133 
134     # ! Using lambda to do 2 fuctions with 1 command
135     go_back_button = ttk.Button(exist_win,text='Go back',command=lambda :[
    clear(), show_details()])
136     go_back_button.pack(pady=10)
137 
138 
139 
140     clear_button = ttk.Button(exist_win,text='Clear ClipBoard',command=
    clear_clipboard)
141     clear_button.pack(pady=10)
142 
143 def clear_clipboard():
144     pyperclip.copy(' ')
145     messagebox.showinfo('Success','Clipboard Has been Cleared!')
146 
147  # Copying the password to computers clipboard
148 def select():
149     global trv
150     try:
151         curItem = trv.focus()
152         value = trv.item(curItem)['values'][1]
153         print(curItem,value)
154         messagebox.showinfo('WARNING','!!!COPYING PASSWORD WILL SAVE A COPY TO
     THE CLIPBOARD,REMEMBER TO CLEAR THE CLIPBOARD ONCE PASSWORD IS PASTED!!!')
155         pyperclip.copy(value)
156         messagebox.showinfo('Success','Password Copied to Clipboard')
157     except:
158         messagebox.showerror('Error','Please Select A Record to Copy!')
159 
160 
161 
162 
163 # adding new passwords for existing users
164 def dis_pass():
165     service_name = (service_entry.get())
166     username_add = (username_entry.get())
167     service_passwd = (masterpwd_entry.get())
168     regex = r"^[a-z0-9]+[\._]?[a-z0-9]+[@]\w+[.]\w+$"
169     if(re.search(regex,username_add)):
170         save.store_passwd(username,service_passwd,masterpassword,username_add,
    service_name)
171         service_entry.delete(0,END)
172         username_entry.delete(0,END)
173         masterpwd_entry.delete(0,END)
174 
175     else:
176         messagebox.showerror('Invalid Email Entered','Invalid Email Entered!
```

```
176 Please check')
177         new_add_win.destroy()
178         add_new_passwd()
179 
180     # ! refreshing screen as a password is added
181     exit_button_add = ttk.Button(new_add_win, text='Close', command = lambda
     :[new_add_win.destroy(),clear(),show_details()])
182     exit_button_add.grid(row=5, column=0, columnspan=2, pady=10, padx=20,ipadx
    =125)
183 
184 
185 
186 
187 # window for adding new passwords for existing users
188 def add_new_passwd():
189 
190     global new_add_win, username_entry, masterpwd_entry ,service_entry,
    sumbit_button
191     new_add_win = tk.Toplevel()
192     new_add_win.title("Store your password")
193     new_add_win.geometry('400x400+500+300')
194     new_add_win.resizable('false', 'false')
195     new_add_win.configure(bg='black')
196 
197     # labels
198     service_label = tk.Label(new_add_win, text='SERVICE NAME:',bg='black',fg=
    '#39FF14',font=('helvetica',14))
199     service_label.grid(row=0, column=0, pady=20)
200 
201     username_label = tk.Label(new_add_win, text='EMAIL :',bg='black',fg='#
    39FF14',font=('helvetica',14))
202     username_label.grid(row=1, column=0, pady=20)
203 
204     dum_label1 = tk.Label(new_add_win, text='')
205     dum_label1.grid(row=2, column=0, pady=20)
206 
207     masterpwd_label = tk.Label(new_add_win, text='PASSWORD :',bg='black',fg='#
    39FF14',font=('helvetica',14))
208     masterpwd_label.grid(row=2, column=0, pady=20)
209 
210     # entry
211     service_entry = tk.Entry(new_add_win)
212     service_entry.grid(row=0, column=1, pady=20)
213     username_entry = tk.Entry(new_add_win)
214     username_entry.grid(row=1, column=1, pady=20)
215     masterpwd_entry = tk.Entry(new_add_win,show='*')
216     masterpwd_entry.grid(row=2, column=1, pady=20)
217 
218     # dummy label
219     dum_label1 = tk.Label(new_add_win, text='',bg='black',fg='black')
220     dum_label1.grid(row=3, column=1, pady=20)
221 
222     # sumbit button
223     sumbit_button = ttk.Button(new_add_win, text='ADD RECORD', command=
    dis_pass)
224     sumbit_button.grid(row=4, column=0, columnspan=2, pady=20, padx=30,ipadx=
    125)
225 
226     # focusing on entry box
227     service_entry.focus()
228 
229 def actually_modify_password():
```

```
230     email_add = modify_username_entry.get()
231     modify_service_name = modify_service_entry.get()
232     modify_oldpassword = modify_masterpwd_entry.get()
233     modify_newpwd = modify_newpwd_entry.get()
234     modify.update(username,modify_service_name,email_add,modify_oldpassword,
    modify_newpwd)
235 
236     modify_username_entry.delete(0,END)
237     modify_service_entry.delete(0,END)
238     modify_masterpwd_entry.delete(0,END)
239     modify_newpwd_entry.delete(0,END)
240 
241     modify_sumbit_button.config(text='Close',command=modify_win.destroy)
242 
243 
244 def actually_delete_password():
245     del_service_name = delete_service_entry.get()
246     del_email = delete_username_entry.get()
247     delete.delete_rec(username,del_service_name,del_email)
248     delete_service_entry.delete(0,END)
249     delete_username_entry.delete(0,END)
250 
251     delete_sumbit_button.config(text='Close',command=delete_win.destroy)
252 
253 # deleting password
254 def delete_password():
255     global delete_win, delete_service_entry,delete_username_entry,
    delete_sumbit_button
256 
257     delete_win = tk.Toplevel()
258     delete_win.title("Delete Password")
259     delete_win.geometry('400x400+500+300')
260     delete_win.resizable('false', 'false')
261     delete_win.configure(bg='black')
262 
263     # labels
264     delete_service_label = tk.Label(delete_win, text='SERVICE NAME:',bg='black
    ',fg='#39FF14',font=('helvetica',14))
265     delete_service_label.grid(row=0, column=0, pady=20)
266 
267     delete_username_label = tk.Label(delete_win, text='EMAIL :',bg='black',fg=
    '#39FF14',font=('helvetica',14))
268     delete_username_label.grid(row=1, column=0, pady=20)
269 
270     delete_dum_label1 = tk.Label(delete_win, text='',bg='black',fg='black')
271     delete_dum_label1.grid(row=2, column=0, pady=20)
272 
273 
274     # entry
275     delete_service_entry = tk.Entry(delete_win)
276     delete_service_entry.grid(row=0, column=1, pady=20)
277     delete_username_entry = tk.Entry(delete_win)
278     delete_username_entry.grid(row=1, column=1, pady=20)
279 
280 
281 
282     # sumbit button
283     delete_sumbit_button = ttk.Button(delete_win, text='DELETE RECORD',
    command=actually_delete_password)
284     delete_sumbit_button.grid(row=4, column=0, columnspan=2, pady=20, padx=30,
    ipadx=125)
285 
```

```
286     # focusing on entry box
287     delete_service_entry.focus()
288 
289 
290 
291 # modifying exisiting passwords
292 def modify_password():
293     global modify_win, modify_username_entry, modify_masterpwd_entry , 
    modify_service_entry,modify_sumbit_button,modify_newpwd_entry
294     modify_win = tk.Toplevel()
295     modify_win.title("Change Password")
296     modify_win.geometry('400x400+500+300')
297     modify_win.resizable('false', 'false')
298     modify_win.configure(bg='black')
299 
300     # labels
301     modify_service_label = tk.Label(modify_win, text='SERVICE NAME:',bg='black
    ',fg='#39FF14',font=('helvetica',14))
302     modify_service_label.grid(row=0, column=0, pady=20)
303 
304     modify_username_label = tk.Label(modify_win, text='EMAIL :',bg='black',fg=
    '#39FF14',font=('helvetica',14))
305     modify_username_label.grid(row=1, column=0, pady=20)
306 
307     modify_dum_label1 = tk.Label(modify_win, text='',bg='black',fg='black')
308     modify_dum_label1.grid(row=2, column=0, pady=20)
309 
310     modify_masterpwd_label = tk.Label(modify_win, text='OLD PASSWORD :',bg='
    black',fg='#39FF14',font=('helvetica',14))
311     modify_masterpwd_label.grid(row=2, column=0, pady=20)
312 
313     modify_newpwd_label = tk.Label(modify_win, text='NEW PASSWORD :',bg='black
    ',fg='#39FF14',font=('helvetica',14))
314     modify_newpwd_label.grid(row=3, column=0, pady=20)
315 
316     # entry
317     modify_service_entry = tk.Entry(modify_win)
318     modify_service_entry.grid(row=0, column=1, pady=20)
319     modify_username_entry = tk.Entry(modify_win)
320     modify_username_entry.grid(row=1, column=1, pady=20)
321     modify_masterpwd_entry = tk.Entry(modify_win,show='*')
322     modify_masterpwd_entry.grid(row=2, column=1, pady=20)
323     modify_newpwd_entry = tk.Entry(modify_win,show='*')
324     modify_newpwd_entry.grid(row=3, column=1, pady=20)
325 
326 
327     # sumbit button
328     modify_sumbit_button = ttk.Button(modify_win, text='CHANGE PASSWORD',
    command=actually_modify_password)
329     modify_sumbit_button.grid(row=4, column=0, columnspan=2, pady=20,padx=10,
    ipadx=125)
330 
331     # focusing on entry box
332     modify_service_entry.focus()
333 
334 
335 
336 # options in edit password
337 def edit_existing_passwd():
338     global modify_button_image1,delete_pass_image,exit_button_image_1
339     clear()
340     modify_button_image1 = tk.PhotoImage(file='/home/kali/Downloads/
```

```
Project_Password_Manager_12A/Main program/uppass.png',master=exist_win)
    modify_pass_button = tk.Button(exist_win,image=modify_button_image1,
command= modify_password)
    modify_pass_button.pack(pady=20)

    delete_pass_image = tk.PhotoImage(file='/home/kali/Downloads/
Project_Password_Manager_12A/Main program/deletepass.png',master=exist_win)
    delete_pass_button = tk.Button(exist_win, image=delete_pass_image, command
= delete_password)
    delete_pass_button.pack(pady=20)

    exit_button_image_1 = tk.PhotoImage(file='/home/kali/Downloads/
Project_Password_Manager_12A/Main program/CLOSE.png',master=exist_win)
    exit_button_add = tk.Button(exist_win, image=exit_button_image_1, command
 = lambda :[clear(),show_details()])
    exit_button_add.pack(pady=20)



# showing services
def show_details():
    global service_listbox,service_image,add_image,modify_image,log_out_image
    exist_win.title("Password Manager")
    total_details = get_exist_pass(username)
    if total_details:
        select_label = tk.Label(exist_win,text='Select Service to Show
Password ',bg='black',fg='#39FF14',font=('helvetica',14))
        select_label.pack()
    else:
        select_label = tk.Label(exist_win,text='Please add Passwords!',bg='
black',fg='#39FF14',font=('helvetica',14))
        select_label.pack()
    service_list = []
    for services in total_details:
        if services[2] not in service_list:
            service_list.append(services[2])
    service_listbox = tk.Listbox(exist_win,background="black", fg="white",
selectbackground="blue",highlightcolor="Red")
    service_listbox.pack(pady=5)
    service_listbox.insert(tk.END,*service_list)

    service_image = tk.PhotoImage(file='/home/kali/Downloads/
Project_Password_Manager_12A/Main program/showpasswords.png',master=exist_win)
    service_button = tk.Button(exist_win,image=service_image,command=display)
    service_button.pack(pady=5)

    add_image = tk.PhotoImage(file='/home/kali/Downloads/
Project_Password_Manager_12A/Main program/addpass.png',master=exist_win)
    add_button = tk.Button(exist_win,image=add_image,command=add_new_passwd)
    add_button.pack(pady=5,expand=TRUE)

    modify_image = tk.PhotoImage(file='/home/kali/Downloads/
Project_Password_Manager_12A/Main program/CHANGE.png',master=exist_win)
    modify_button = tk.Button(exist_win,image=modify_image,command=
edit_existing_passwd)
    modify_button.pack(pady=5,expand=TRUE)

    log_out_image = tk.PhotoImage(file='/home/kali/Downloads/
Project_Password_Manager_12A/Main program/logout.png',master=exist_win)
    log_out_button = tk.Button(exist_win,image=log_out_image,command=exist_win
.destroy)
    log_out_button.pack(pady=5,expand=TRUE)
```

```
388
389
390
391
392
393
394 # checking master passsword
395 def checkpwd():
396     global username,masterpassword
397     username = username_entry.get()
398     masterpassword = masterpwd_entry.get()
399     salt = 'LTC'
400     passwd = masterpassword[0:1] + salt + masterpassword[1:]
401     res=hashlib.md5(bytes(passwd,'utf-8')).hexdigest()
402
403     chktup=(username,res)
404     for k in get_master():
405         if k == chktup:
406             messagebox.showinfo('Success!','Signed in Successfully')
407             clear()
408             show_details()
409             break
410     else:
411         messagebox.showerror('Failed!','Master Password and Username dont
    match')
412         masterpwd_entry.delete(0,tk.END)
413
414
415
416
417
418 def exist_user():
419     global exist_win,username_entry,masterpwd_entry
420     exist_win = tk.Toplevel()
421     exist_win.title("Sign In")
422     exist_win.geometry('400x400+500+300')
423     exist_win.resizable('false','false')
424     exist_win.configure(bg='black')
425
426     # labels
427     username_label = tk.Label(exist_win,text='USERNAME :',bg='black',fg='#
    39FF14',font=('helvetica',14))
428     username_label.grid(row=0,column=0,pady=20)
429
430     dum_label1 = tk.Label(exist_win,text='',bg='black',fg='green',font=('
    helvetica',14))
431     dum_label1.grid(row=1,column=0,pady=20)
432
433     masterpwd_label = tk.Label(exist_win,text='MASTER PASSWORD :',bg='black',
    fg='#39FF14',font=('helvetica',14))
434     masterpwd_label.grid(row=2,column=0,pady=20)
435
436     # entry
437     username_entry=tk.Entry(exist_win)
438     username_entry.grid(row=0,column=1,pady=20)
439     masterpwd_entry=tk.Entry(exist_win,show="*")
440     masterpwd_entry.grid(row=2,column=1,pady=20)
441
442     # dummy label
443     dum_label1 = tk.Label(exist_win,text='',bg='black',fg='white',font=('
    helvetica',10))
444     dum_label1.grid(row=3,column=1,pady=20)
```

```
445
446     # sumbit button
447     #sign_in_image = tk.PhotoImage(file='signin.png')
448     sumbit_button = ttk.Button(exist_win,text='SIGN IN',command=checkpwd)
449     sumbit_button.grid(row=5,column=0,columnspan=2,pady=10,padx=15,ipadx=150)
450
451     # focusing on entry box
452     username_entry.focus()
453
454
455 def sumbit():
456     global exit_button_image,close_image
457     name = name_entry.get()
458     master = master_entry.get()
459     chkmaster = masterconfi_entry.get()
460     if name!= '' and master!='':
461         if chkmaster == master:
462             add_master(name,master)
463             messagebox.showinfo('Success',f'New User {name} Created! ')
464             forget_newuser()
465             message_label.config(text='Sign Into Existing User Please')
466         else:
467             messagebox.showerror('Failed','Master Password error! Try later')
468             forget_newuser()
469     else:
470         messagebox.showerror('Failed','Fields Are Empty! Please Fill in')
471         forget_newuser()
472
473     close_image = tk.PhotoImage(file='/home/kali/Downloads/
    Project_Password_Manager_12A/Main program/CLOSE.png',master=new_win)
474     sumbit_button.config(image=close_image,command=new_win.destroy)
475
476
477
478 def new_user():
479     global name_entry,master_entry,masterconfi_entry,sumbit_button,new_win,
    message_label,sumbit_user_image
480     new_win = tk.Toplevel()
481     new_win.title("New User")
482     new_win.geometry('400x400+500+300')
483     new_win.resizable('false','false')
484     new_win.configure(bg='black')
485
486     # labels
487     name_label = tk.Label(new_win,text='USERNAME :',bg='black',fg='#39FF14',
    font=('helvetica',10))
488     name_label.grid(row=0,column=0,pady=20,padx=20)
489     master_label = tk.Label(new_win,text='MASTER PASSWORD :',bg='black',fg='#
    39FF14',font=('helvetica',10))
490     master_label.grid(row=1,column=0,pady=20,padx=20)
491     confi_label = tk.Label(new_win,text='CONFIRM MASTER PASSWORD :',bg='black'
    ,fg='#39FF14',font=('helvetica',10))
492     confi_label.grid(row=2,column=0,pady=20,padx=20)
493
494     # entry
495     name_entry=tk.Entry(new_win)
496     name_entry.grid(row=0,column=1,pady=20)
497     master_entry=tk.Entry(new_win,show="*")
498     master_entry.grid(row=1,column=1,pady=20)
499     masterconfi_entry=tk.Entry(new_win,show="*")
500     masterconfi_entry.grid(row=2,column=1,pady=20)
501
```

```
502     # dummy label for grid
503     #dum_label1 = tk.Label(new_win,text='')
504     #dum_label1.grid(row=3,column=0,pady=20)
505     message_label = tk.Label(new_win,text='',bg='black',fg='white',font=('
    helvetica',10))
506     message_label.grid(row=4,column=0,pady=20)
507
508
509     # sumbit button
510     sumbit_user_image = tk.PhotoImage(file='/home/kali/Downloads/
    Project_Password_Manager_12A/Main program/SUBMIT.png',master=new_win)
511     sumbit_button = tk.Button(new_win,image=sumbit_user_image,command=sumbit)
512     sumbit_button.grid(row=6,column=1,pady=10,padx=20)
513
514     # focusing on entry box
515     name_entry.focus()
516
517 def open_info():
518     messagebox.showinfo('Information','''This is a GUI Password Manager, That
    uses Encryption to store passwords localy on your machine ensuring maximum
    privacy and giving you the option of having multiple users. A make in india
    initative.''')
519
520
521
522 def about_us():
523     global about_Img
524     about_win = tk.Toplevel()
525     about_win.title("ABOUT US")
526     about_win.geometry('400x400+500+300')
527     about_win.resizable('false','false')
528     about_win.configure(bg='black')
529
530     about_Img = ImageTk.PhotoImage(file="/home/kali/Downloads/
    Project_Password_Manager_12A/Main program/about_final.png",master=about_win)
531     about_img_button = tk.Button(about_win, image=about_Img,command=open_info)
532     about_img_button.pack()
533
534
535
536 def open():
537     global Img,win,about_image,new_user_image,exist_user_image
538     root.geometry('500x500+50+50')
539     win = tk.Toplevel()
540     win.title("PASSWORD MANAGER")
541     win.geometry('400x400+500+300')
542     win.resizable('false','false')
543     win.configure(bg='black')
544
545     #picture
546     Img = ImageTk.PhotoImage(file="/home/kali/Downloads/
    Project_Password_Manager_12A/Main program/LOGODARKSMALL1.png",master=win)
547     img_label = tk.Label(win, image=Img)
548     img_label.pack()
549
550     #main_buttons
551     new_user_image = tk.PhotoImage(file='/home/kali/Downloads/
    Project_Password_Manager_12A/Main program/new.png',master=win)
552     new_button = tk.Button(win,image=new_user_image,command=new_user)
553     new_button.pack(side=LEFT)
554
555     exist_user_image = tk.PhotoImage(file='/home/kali/Downloads/
```

```
Project_Password_Manager_12A/Main program/exist2.png',master=win)
    exist_button = tk.Button(win,image=exist_user_image,command=exist_user)
    exist_button.pack(side=RIGHT)

    #about us
    about_image = tk.PhotoImage(file='/home/kali/Downloads/
Project_Password_Manager_12A/Main program/about us.png',master=win)
    about_button = tk.Button(win,image=about_image,command=about_us)
    about_button.pack(side=LEFT,padx=40)

def delete_user_from_database(username_to_delete):
    db = mysql.connect(user='root',passwd='root',host='localhost',database='
master_password')
    cur = db.cursor()
    statement = 'DROP table {}'.format(username_to_delete)
    state2 = '''DELETE from password
    WHERE name = "{}" '''.format(username_to_delete)
    try:
        cur.execute(statement)
        cur.execute(state2)
        db.commit()
        messagebox.showinfo('Success',f'User {username_to_delete} successfully
 deleted!')
    except Exception as error:
        db.rollback()
        messagebox.showerror('Failed',f"Couldn't Delete User Data : {error}")


def check_for_del():
    user_data = get_master()
    del_username = del_username_entry.get()
    del_passwd = del_masterpwd_entry.get()
    data = (del_username,del_passwd)
    if data in user_data:
        response = messagebox.askokcancel('User Found',f'{del_username} found
, Proceed to Delete?')
        if response == 1:
            delete_user_from_database(del_username)
        else:
            messagebox.showinfo('Failed','User not deleted')
    else:
        messagebox.showerror('Error','User Not found!')
    del_username_entry.delete(0,END)
    del_masterpwd_entry.delete(0,END)




def delete_screen():
    global del_username_entry,del_masterpwd_entry,del_win,
del_sumbit_button_image
    del_win = tk.Toplevel()
    del_win.title("Delete User")
    del_win.geometry('400x400+500+300')
    del_win.resizable('false','false')
    del_win.configure(bg='black')

    # labels
    del_label = tk.Label(del_win,text='USERNAME :',bg='black',fg='#39FF14',
font=('helvetica',12))
    del_label.grid(row=0,column=0,pady=20,padx=20)
```

```
611
612     del_dum_label1 = tk.Label(del_win,text='',bg='black',fg='#39FF14')
613     del_dum_label1.grid(row=1,column=0,pady=20)
614
615     del_masterpwd_label = tk.Label(del_win,text='MASTER PASSWORD :',bg='black'
    ,fg='#39FF14',font=('helvetica',12))
616     del_masterpwd_label.grid(row=2,column=0,pady=20,padx=20)
617
618     # entry
619     del_username_entry=tk.Entry(del_win)
620     del_username_entry.grid(row=0,column=1,pady=20)
621     del_masterpwd_entry=tk.Entry(del_win,show="*")
622     del_masterpwd_entry.grid(row=2,column=1,pady=20)
623
624     # dummy label
625     del_dum_label3 = tk.Label(del_win,text='',bg='black',fg='white')
626     del_dum_label3.grid(row=3,column=1,pady=20)
627
628
629     # sumbit button
630
631     del_sumbit_button_image = tk.PhotoImage(file='/home/kali/Downloads/
    Project_Password_Manager_12A/Main program/SUBMIT.png',master=del_win)
632     del_sumbit_button = tk.Button(del_win,image=del_sumbit_button_image,
    command=check_for_del,borderwidth=2)
633     del_sumbit_button.grid(row=4,column=1,pady=10,padx=50)
634
635     # focusing on entry box
636     del_username_entry.focus()
637
638
639 #main window
640 root = tk.Tk()
641 root.title("PASSWORD MANAGER")
642 root.geometry('500x500+500+150')
643 root.resizable('false','false')
644 root.configure(bg='black')
645 about_Img_1 = ImageTk.PhotoImage(file="/home/kali/Downloads/
    Project_Password_Manager_12A/Main program/about_final.png",master=root)
646 about_img_button = tk.Button(root, image=about_Img_1,command=open_info)
647 about_img_button.pack()
648
649 main_button_image = tk.PhotoImage(file='/home/kali/Downloads/
    Project_Password_Manager_12A/Main program/start.png',master=root)
650 main_button = tk.Button(root,image=main_button_image,command=open)
651 main_button.pack(side=LEFT)
652
653 exit_button_image = tk.PhotoImage(file='/home/kali/Downloads/
    Project_Password_Manager_12A/Main program/end.png',master=root)
654 exit_button = tk.Button(root,image=exit_button_image,command=root.quit)
655 exit_button.pack(side=LEFT,padx=90)
656
657 delete_button_image = tk.PhotoImage(file='/home/kali/Downloads/
    Project_Password_Manager_12A/Main program/delete.png',master=root)
658 delete_button = tk.Button(root,image=delete_button_image,command=delete_screen
    )
659 delete_button.pack(side=LEFT)
660
661 root.mainloop()
```

SOURCE CODE

BACKEND – SECURE MODULE

init.py

delete.py

modify.py

retrieve.py

save.py

```
1
2
```

```python
'''
This function takes username,service name and email.It will check for the
parameters in the database and delete the record .
PLEASE CHANGE DATABASE NAME TARUN
'''

import mysql.connector
from tkinter import messagebox
import tkinter as tk

root = tk.Tk()
root.withdraw()

def delete_rec(un,serv,mail):
    try:
        check = mysql.connector.connect(user='root', passwd='root', host='
localhost', database='master_password')
        cursor = check.cursor()
        cursor.execute('select * from {}'.format(un))
        data = cursor.fetchall()
        lg = len(data)
        try:
            for record in data:
                if record[0] == mail and record[2] == serv:
                    break
            else:
                messagebox.showerror('Error','Record Not Found, Please Check!')

            a = 0
            for k in data:
                a += 1
                em = k[0]
                sn = k[2]
                if em == mail:
                    if serv == sn:
                        res = messagebox.askokcancel('Confirm', 'DELETE RECORD
?')
                        if res == 1:
                            cursor.execute('DELETE FROM {} where email="{}" and
 service="{}"'.format(un,em,serv))
                            check.commit()
                            messagebox.showinfo('Success', 'Delete Successful!'
)
                            break
                else:
                    if a == lg:
                        messagebox.showerror('ERROR',"Records don't match!")
                    else:
                        continue
        except:
            messagebox.showerror('Failed','Something Went wrong. Please try
again')
    except:
        messagebox.showerror('Failed','Connection problem ')
```

```
'''This function takes username service name,email id and the old password.It will check each of the parameters
mentioned before and then runs the update command to change the password.Error will pop if nay of the parameters
are not matching!
Tell me if you need me to change the errors to messagebox
PLEASE CHANGE DATABASE NAME TARUN'''

import mysql.connector
from tkinter import messagebox
import tkinter as tk

root = tk.Tk()
root.withdraw()

def update(un,serv,mail,opass,pw):
    try:
        check = mysql.connector.connect(user='root', passwd='root', host='localhost', database='master_password')
        cursor = check.cursor()
        cursor.execute('select * from {}'.format(un))
        data = cursor.fetchall()
        lg=len(data)
        try:
            for record in data:
                if record[0] == mail and record[2] == serv:
                    break
            else:
                messagebox.showerror('Error','Record Not Found, Please Check!')

            a=0
            for k in data:
                a+=1
                em = k[0]
                sn = k[2]
                tk = k[3]
                cursor.execute('select AES_DECRYPT(a.password,"{}") from {} a where a.email="{}" AND a.service="{}";'.format(tk,un,em,sn))
                res1 = cursor.fetchall()
                pd=res1[0][0]
                if pd == opass:
                    if em == mail:
                        if serv==sn:
                            res = messagebox.askokcancel('Confirm','Confirm Changes?')
                            if res == 1:
                                cursor.execute('update {} set password = AES_ENCRYPT("{}","{}") where email="{}";'.format(un, pw, tk, em))
                                check.commit()
                                messagebox.showinfo('Success','Changes Successful!')
                                break
                else:
                    if a == lg:
                        messagebox.showerror('ERROR',"Records don't match!")
                    else:
                        continue
        except:
            messagebox.showerror('Failed','Something Went Wrong Please try later')
    except:
        messagebox.showerror('Failed','Connection problem ')
```

```
'''
This function takes username and service.It will decrypt and provide the result
 in this form of all the passwords of that service
PLEASE CHANGE DATABASE NAME TARUN
'''

import mysql.connector
from tkinter import messagebox
import tkinter as tk

root = tk.Tk()
root.withdraw()

def get_passwd(un,sn):
    final = []
    try:
        check = mysql.connector.connect(user='root', passwd='root', host='localhost', database='master_password')
        cursor = check.cursor()
        try:
            cursor.execute('select * from {}'.format(un))
            data = cursor.fetchall()
            for k in data:
                tk=k[3]
                em=k[0]
                #sn=k[2]
                cursor.execute('select AES_DECRYPT(a.password,"{}") from {} a where a.email="{}" AND a.service="{}";'.format(tk,un,em,sn))
                res1 = cursor.fetchall()
                for response in res1:
                    if response:
                        for element in response:
                            line = f'{em}               {element}            {sn}'
                            final.append(line)
            return final
        except:
            messagebox.showerror('Failed','!!!!!')
    except:
        messagebox.showerror('Failed','Connection problem ')


#get_passwd('tarun','google')
```

```
1  '''
2  This function takes username,password,key(==master password),email,service name
3  It inserts space as passoword first and then updates it.PLEASE CHANGE DATABASE
   NAME TARUN
4
5  '''
6
7  import mysql.connector
8  from tkinter import messagebox
9  import tkinter as tk
10
11 root = tk.Tk()
12 root.withdraw()
13
14
15 def store_passwd(un,pw,tk,em,sn):
16     try:
17         check = mysql.connector.connect(user='root', passwd='root', host='
   localhost', database='master_password')
18         cursor = check.cursor()
19         df = ' '
20         try:
21             cursor.execute('INSERT INTO {} VALUES ("{}","{}","{}","{}");'.
   format(un,em,df,sn,tk))
22             cursor.execute('update {} set password = AES_ENCRYPT("{}","{}")
   where email="{}" and service="{}";'.format(un,pw,tk,em,sn))
23             check.commit()
24             messagebox.showinfo('Success','Password Added Successfully')
25         except:
26
27             messagebox.showerror('Failed','Password could not be saved!')
28     except:
29         messagebox.showerror('Failed','User not found ')
30
31
32
33 #store_passwd('tarun','ff','y','mail@gm','google')
34
```

OUTPUT SCREENSHOTS

TKINTER FRONTEND

Figure 1 Open Screen of application

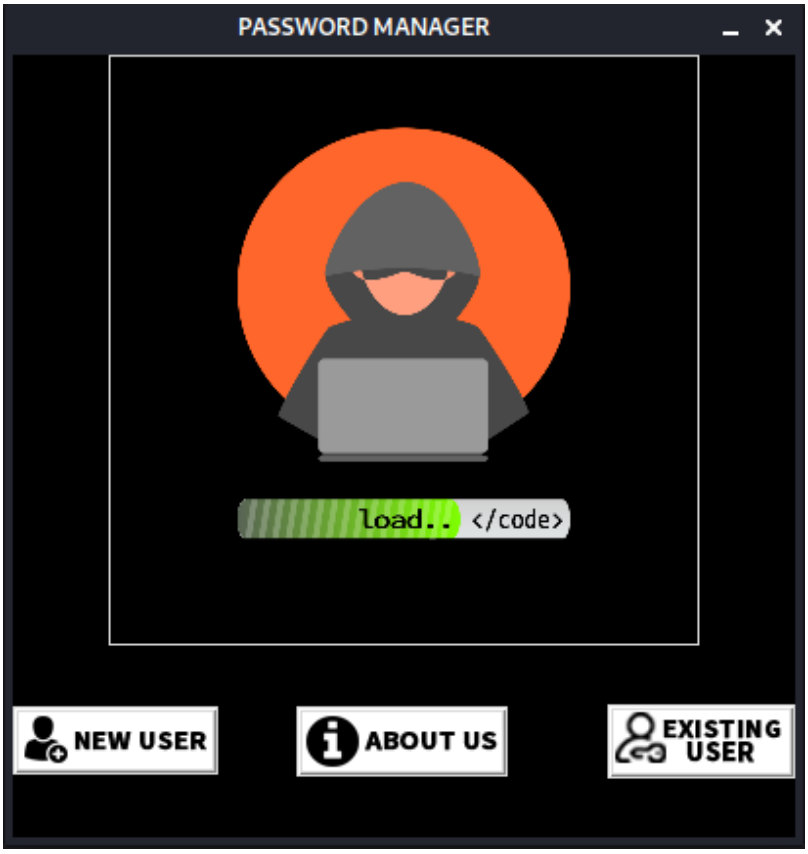


Figure 2 Main Screen of application

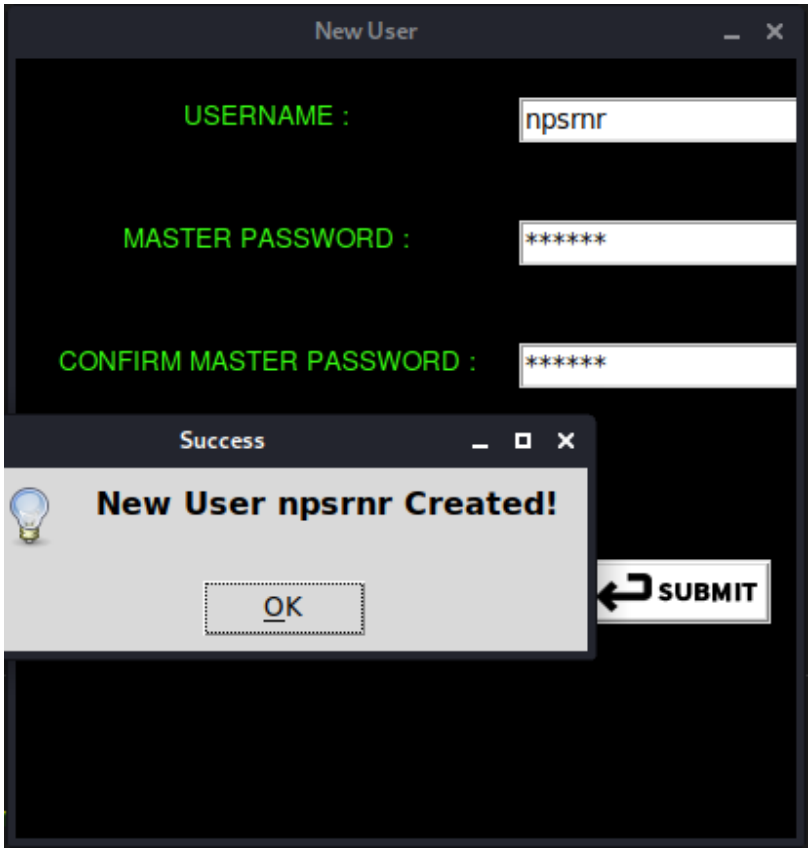


Figure 3 New User registration into application

Figure 4 About Us Screen

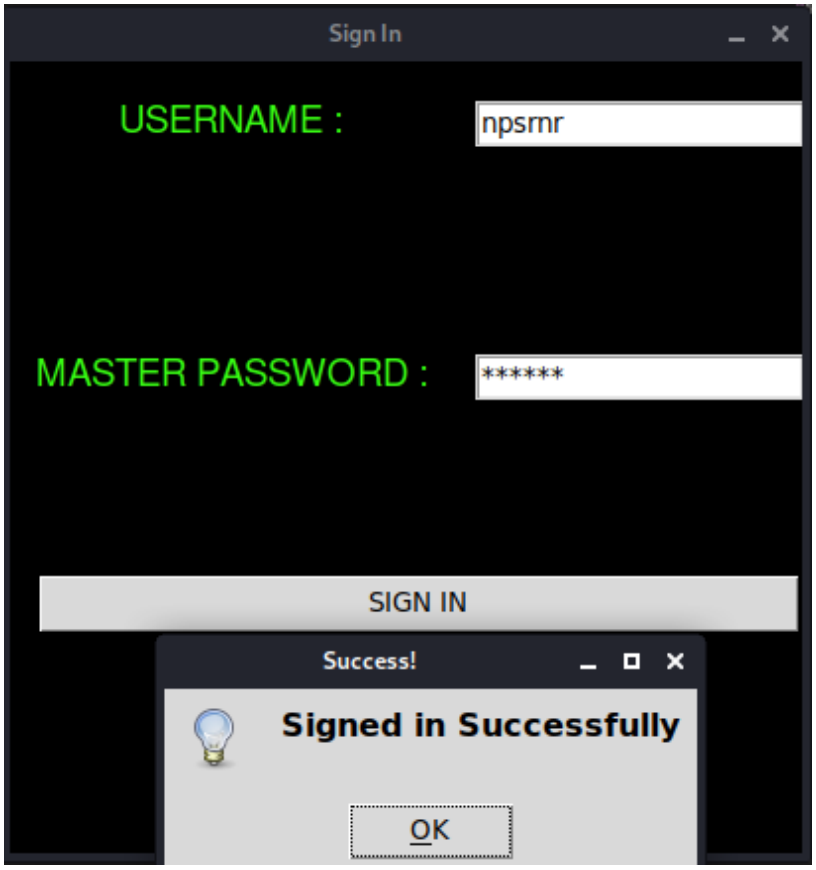


Figure 5 Existing User Sign In

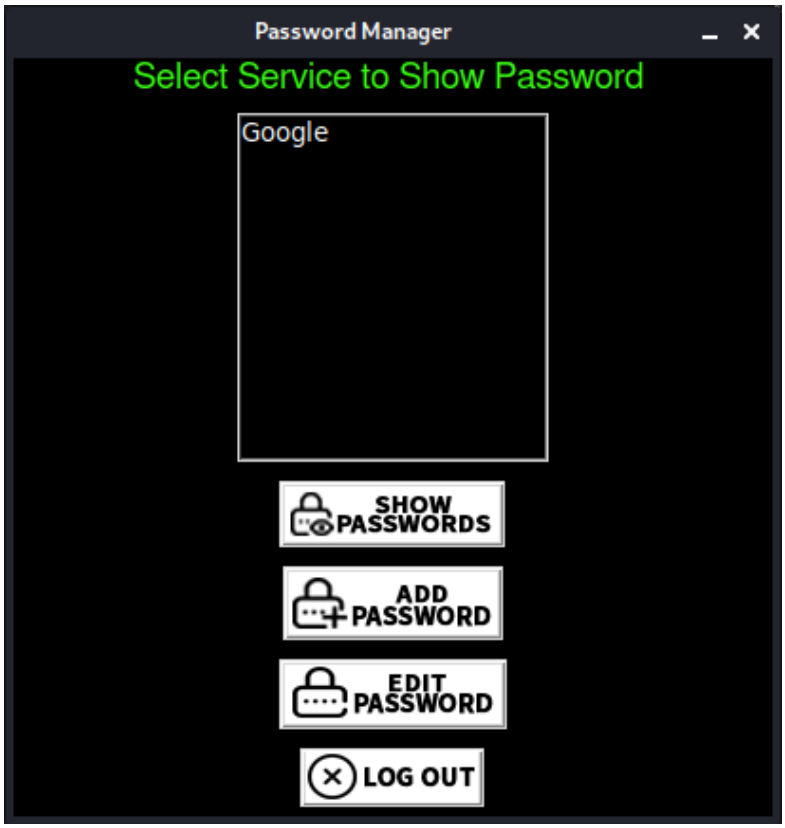


Figure 6 Existing User Open Screen

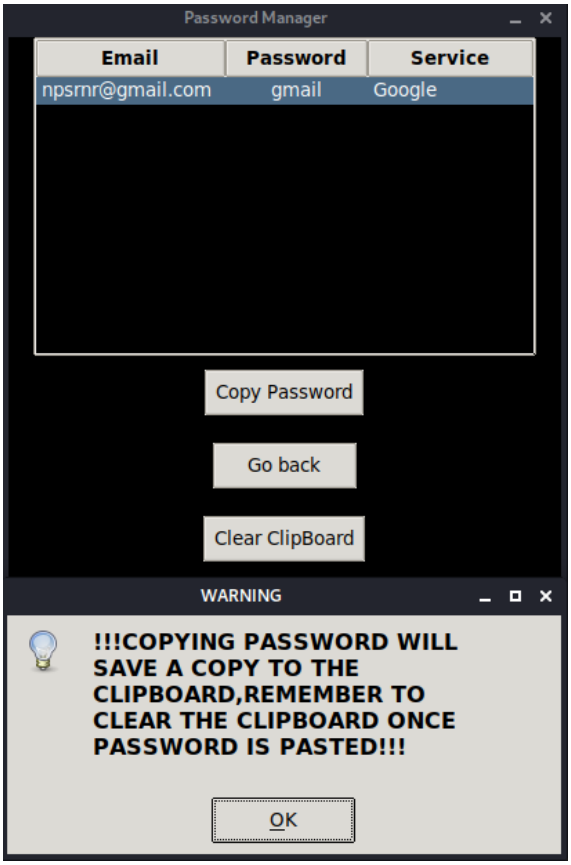


Figure 7 List of all the password being copied

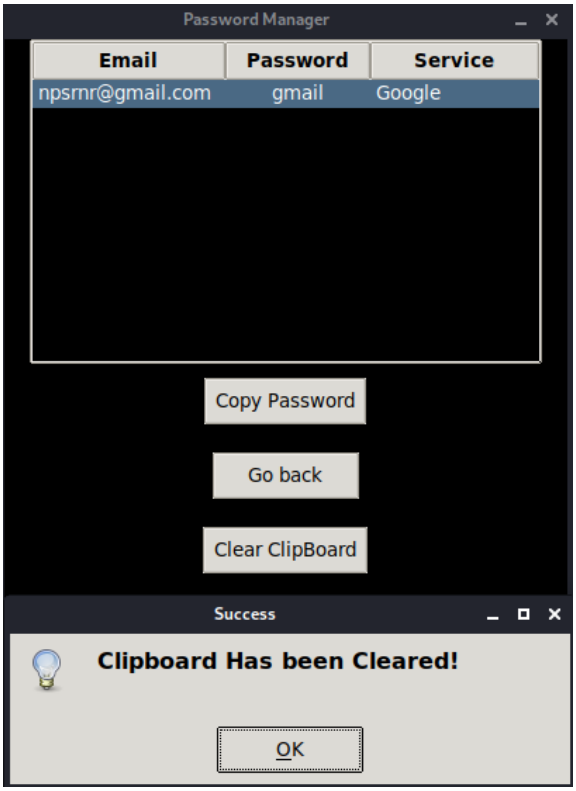


Figure 8 Clearing Clipboard after pasting password

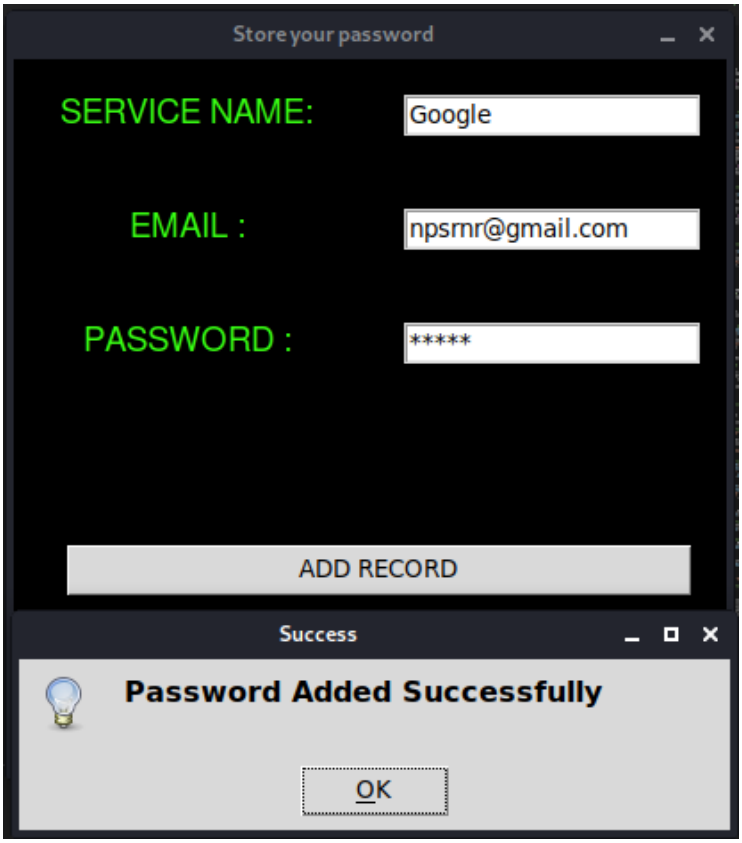


Figure 9 Storing new password

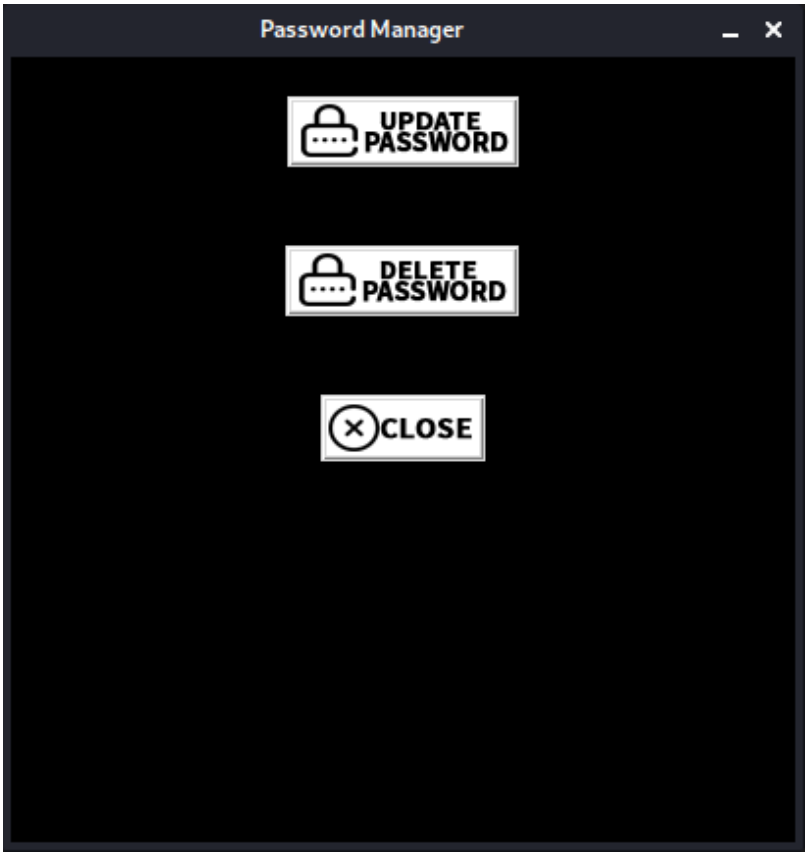


Figure 10 Edit existing password

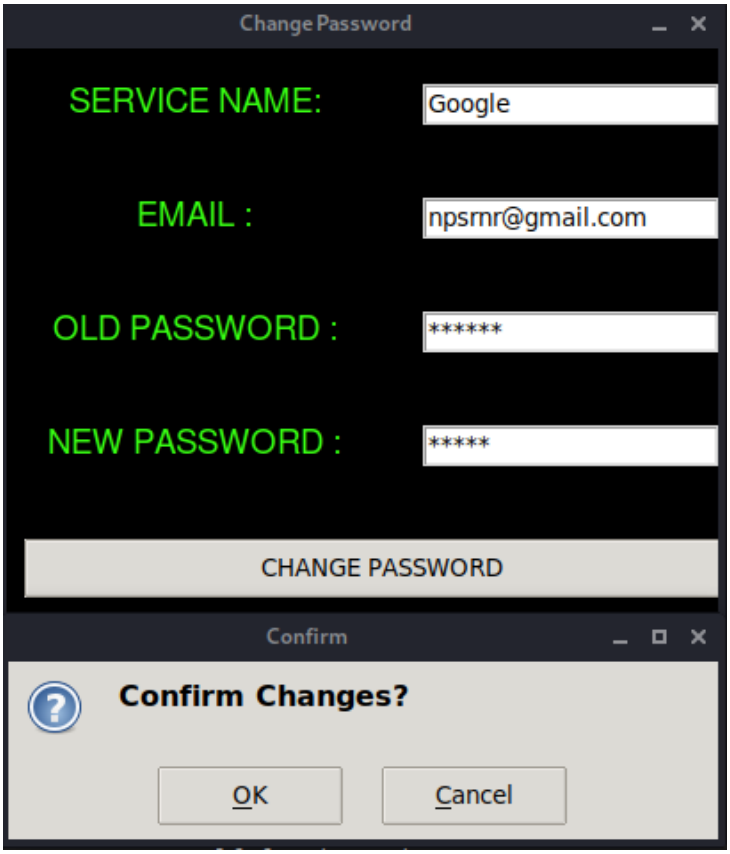


Figure 11 Change Passwords

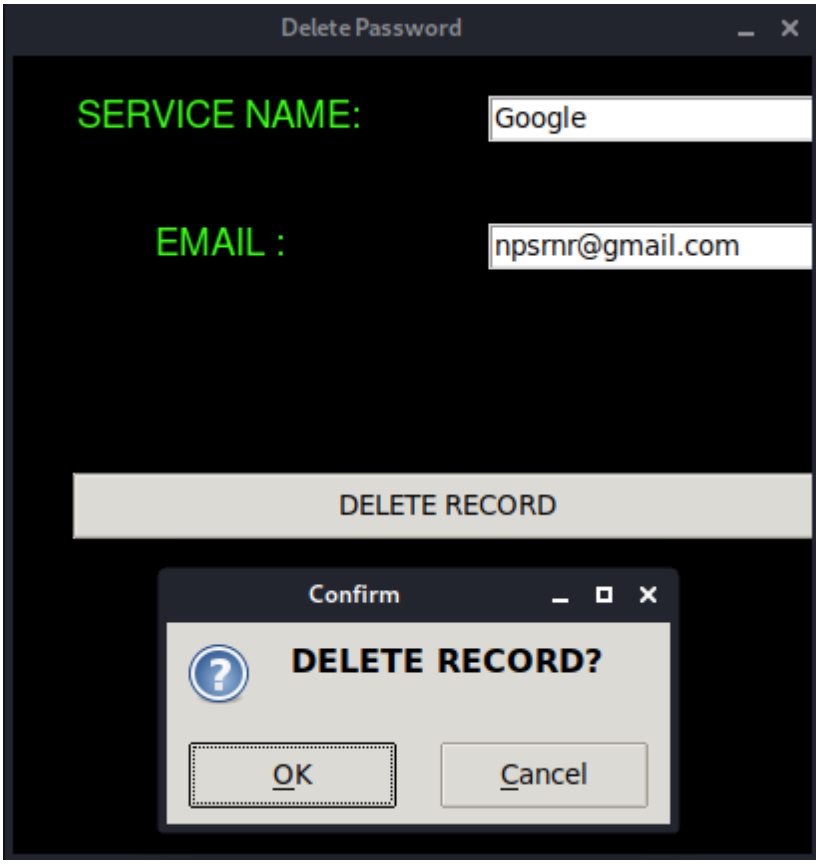


Figure 12 Delete password

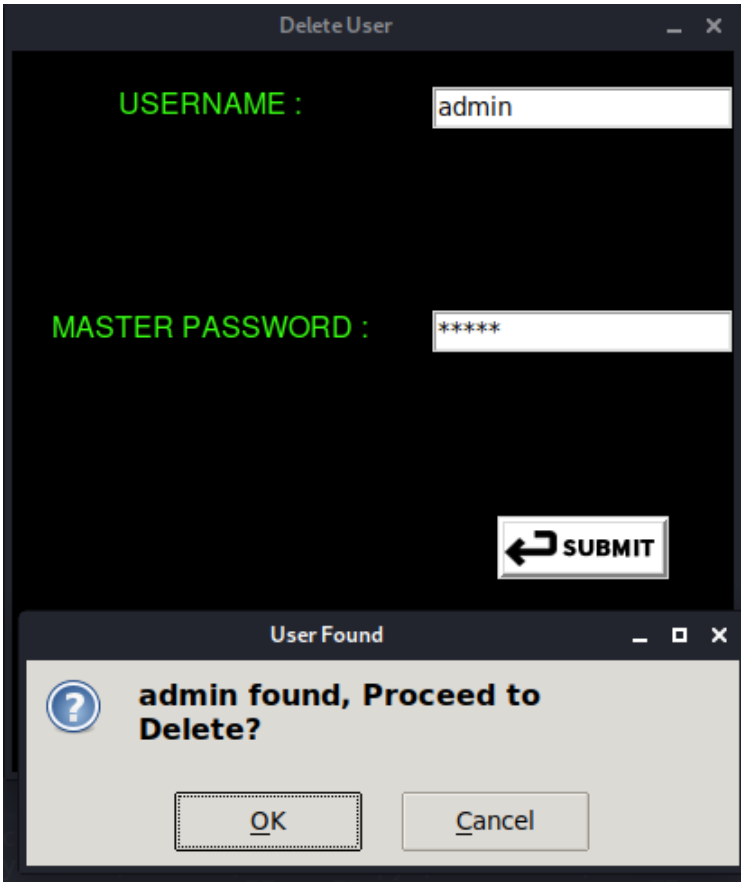


Figure 13 Delete existing User

OUTPUT SCREENSHOTS

MYSQL-BACKEND

```
mysql> show tables;
+-----------------------------+
| Tables_in_master_password |
+-----------------------------+
| npsrnr                      |
| password                    |
+-----------------------------+
2 rows in set (0.00 sec)
```

Figure 14 Tables present in the Database

```
mysql> select * from password;
+--------+----------------------------------+
| name   | password                         |
+--------+----------------------------------+
| npsrnr | 3d926232417d60aa17f5d4384cdd4c85 |
+--------+----------------------------------+
1 row in set (0.01 sec)
```

Figure 15 Table containing all the credentials of users of application

```
mysql> select * from npsrnr;
+------------------+------------------------------------+---------+--------+
| email            | password                           | service | tkey   |
+------------------+------------------------------------+---------+--------+
| npsrnr@gmail.com | 0×581BDAF1D894258BCC7E4739AB261989 | Google  | npsrnr |
+------------------+------------------------------------+---------+--------+
1 row in set (0.00 sec)
```

Figure 16 Table contains the passwords of a particular user

BIBLIOGRAPHY

1. https://www.w3resource.com/mysql/encryption-and-compression-functions/aes_encrypt%28%29.php
2. *"Computer Science with Python"* by Sumita Arora
3. www.codemy.com
4. https://medium.com/better-programming/how-to-hash-in-python-8bf181806141