in collaboration with

Ultimate Password Manager

Submitted By:

Bishal Aryal

Coventry ID: 11780434

Submitted To:

Nirajan Jha

Module Title: ST4061CEM Programming and Algorithms 1

Softwarica College of IT & E-Commerce

Coventry University

September 28, 2021

Assignment:

Individual Project

Softwarica College in collaboration with
Coventry University
Assessment Submission and Declaration Form
PLEASE COMPLETE SECTIONS IN BLOCK CAPITALS

Group work If group work ALL student names and IDs must be added below- on behalf of all members; Name..... ID..... Name..... ID..... Name..... ID..... Name..... ID..... Name..... ID.....	**Surname:** Aryal **First Name:** Bishal **Word Count:** 1484
Student number (ID): 210121	**Attempt:** FIRST ☑ RESIT ☐
Assignment Due Date: 28 September 2021	**Module Code:** ST4061
Programme Title: Ultimate Password Manager	
Module Title: Programming and Algorithm 1	
Name of Supervisor or Tutor (if applicable): Nirajan Jha	**Individual Work:** ☑ **Group Work:** ☐
Assessment Title and Type(ie essay, journal, CD, Dissertation)	

I have read the Softwarica College rules and regulations on the submission of academic work and in particular the sections concerning misconduct in assessment, including plagiarism, collusion and cheating. I certify that this assignment is the result of my own5 (or group) work and contains no unreferenced material from another source and does not contravene any part of the College's rules and regulations.

I acknowledge that in submitting this work I am declaring that I (or my group) are fit to be assessed and that a deferral may not be requested following hand in.

I confirm that an electronic version of the item to be assessed where appropriate) is available and will be made available to the College by the specified deadline via Moodle.

In respect of group assignments, the submission of this work is made on the basis that all group members are jointly and severally responsible for the work presented for assessment and that by handing in this item for assessment, all group members acknowledge and confirm the statements above and that ALL student names and ID numbers for the group are listed.

Student(s) Signature: Bishal	**College Stamp**

Acknowledgment

I would like to express my heartfelt appreciation to my teacher Nirajan Jha for providing me with the wonderful opportunity to work on this project titled "Ultimate Password Manager." This opportunity also enabled me to conduct extensive research on integrating Python with the sqlite3 database and I learned a lot about the Tkinter module for creating GUI applications using python.

Abstract

In this project, I created a password manager using python and sqlite3 database which has an easy user graphical interface that provides high security protection of user information. We have entered a lot of account information at the time of login or signup, but we can’t remember each and every detail. To protect our credentials, we can store information securely in a local database using this password manager, and the master key is used to open the locker of passwords. We can easily create unique or strong passwords with mixed symbols, add account information, copy accounts or passwords to the clipboard using this application, and update the old passwords. We can share passwords between multiple devices using this password manager, but we should use the same database file and the master key to access the password manager.

Table of Contents

List of Figures

Words Count:

Introduction – Conclusion: 1484

1. Introduction:

Python is a general-purpose programming language that is interpreted at a high degree of abstraction, and it is so adaptable. We can use it for web development, desktop applications, hacking tools, and many more. *(Introduction To Python,* 2021)

I made an Ultimate Password Manager application with a Graphical User Interface for my assignment using the tkinter module in Python and a sqlite3 database for storage. It has random password generation functionality and storage of information like website name, email id, and password. We can easily update the password in the database, remove the details and copy the account or password to the clipboard using the ultimate password manager.

2. Problem Statement:

Compared with other advanced online password managers, this application is simple and has fewer features. Once the master key is lost, we can't recover it. Md5 encryption is used to hash the master password, but the other account details, like email and password, are shown in plain text after viewing the database file. I will improve the security by adding the feature of protecting database files using passwords and adding forget password functionality in future updates.

3. System Requirements:

Python 3

Operating system: Window, Mac, Linux

Module Used: tkinter, hashlib, sqlite3, random

4. Screenshot of Application Interface:

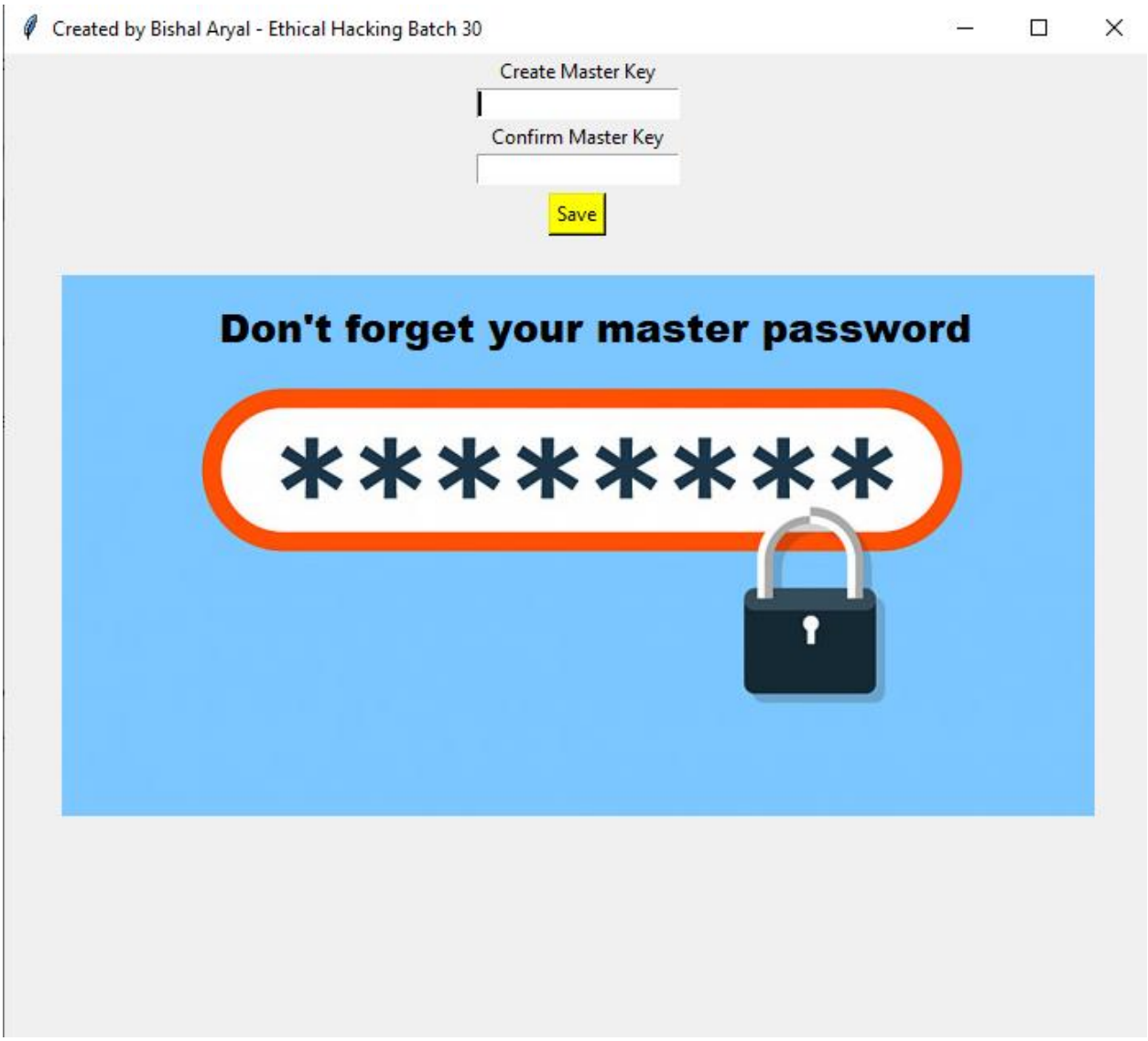

Fig1: Creating Master Password

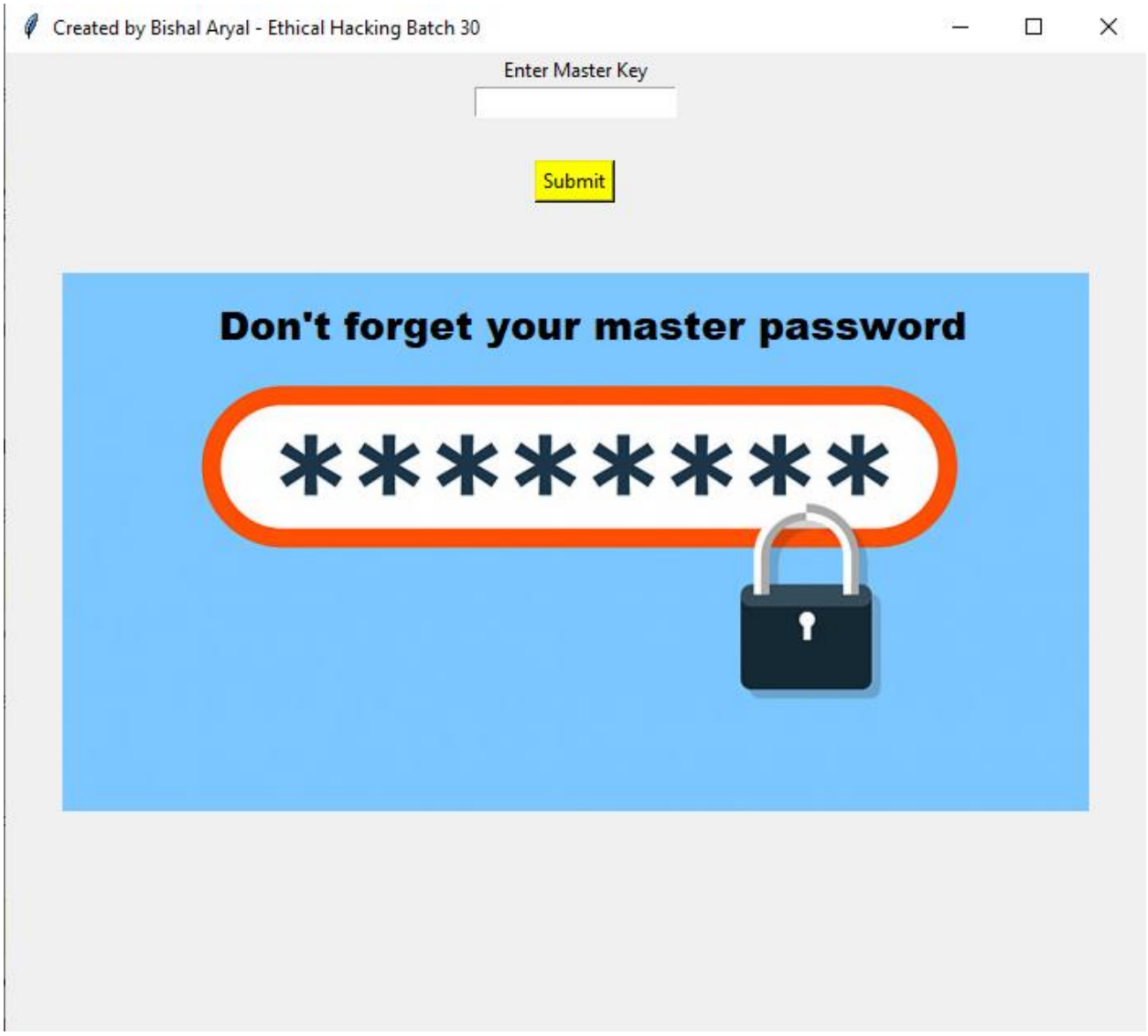

Fig2: Login page for password locker

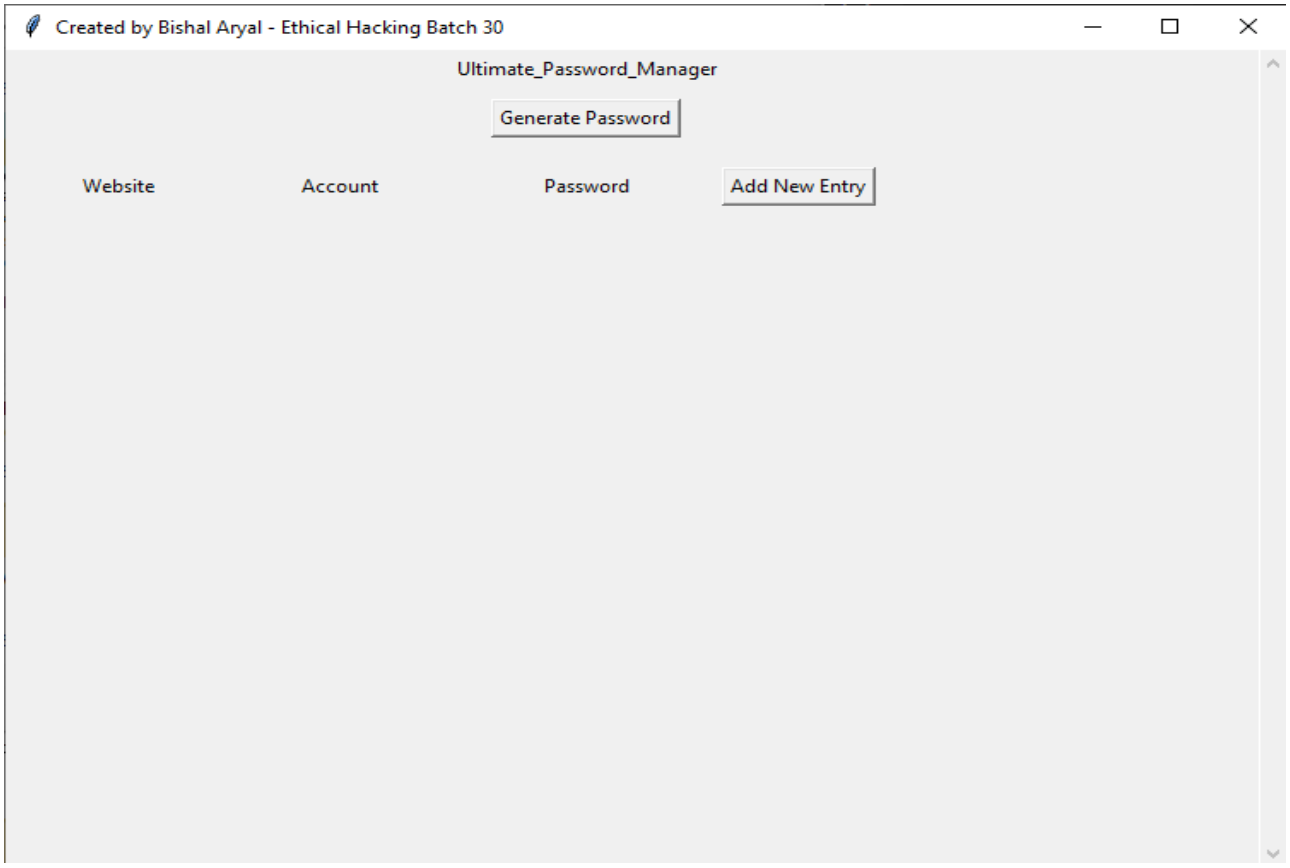

Fig3: Dashboard of Ultimate Password Manager

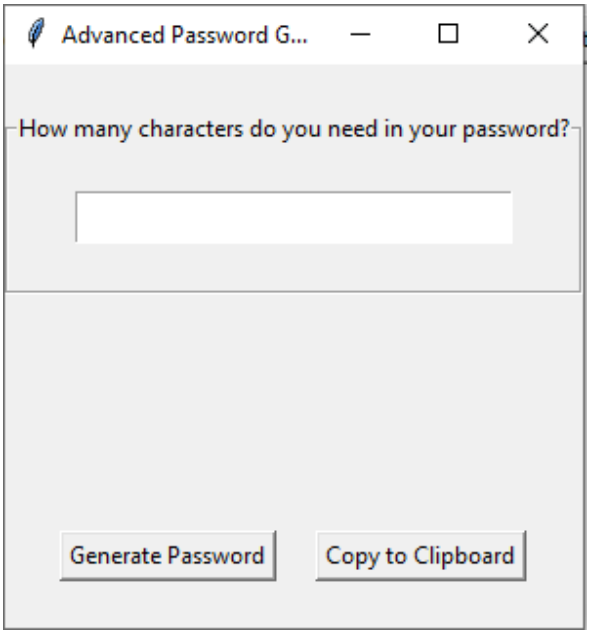

Fig4: Advanced Password Generator

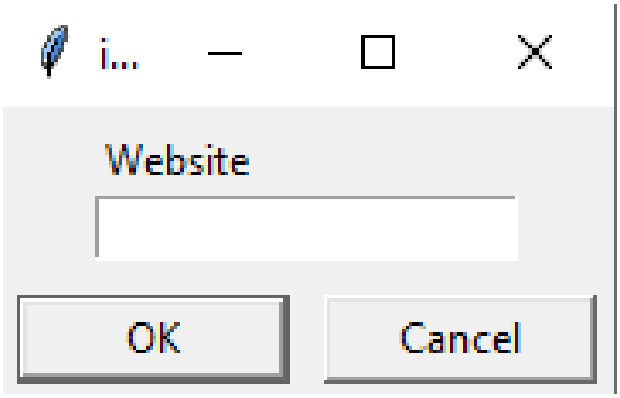

Fig5: Inserting Website Information

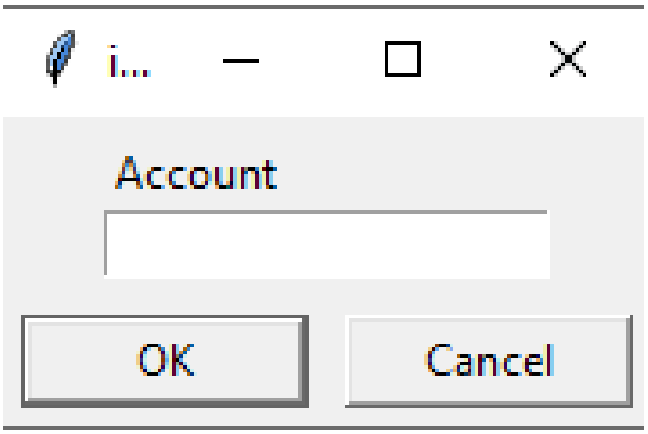

Fig6: Inserting Account Information

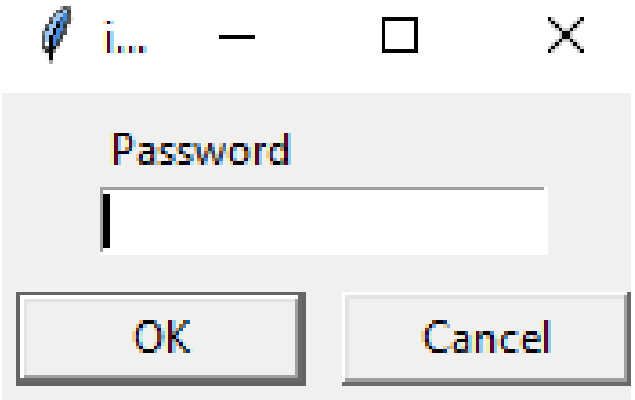

Fig7: Inserting Password Information

5. Detail Explanation of Code:

```
import hashlib
import sqlite3
from functools import partial
from tkinter import *
from tkinter import simpledialog
from tkinter import ttk
from random import randint
import tkinter
```

First, I imported hashlib for hashing the password, Sqlite3 for the database to store the password, and Tkinter for the graphical user interface, buttons, and login forms. For the random password generation functionality, I imported randint from the random module.

```
# Initiate Display Window


display = Tk()
display.update()

display.title("Ultimate_Password_Manager")
display.title("Created by Bishal Aryal - Ethical Hacking Batch 30 ")
```

Let’s initiate the window, naming it "display" and adding the title to the display window using the "display.title" command as shown in the above code.

```
display.mainloop()
```

The "display.mainloop()" command is added in the last of the code to run the program.

```
def firstScreen():
    display.geometry("700x600")

    lbl = Label(display, text="Create Master Key")
    lbl.config(anchor=CENTER)
    lbl.pack()

    txt = Entry(display, width=20, show="*")
    txt.pack()
    txt.focus()

    lbl1 = Label(display, text="Confirm Master Key")
    lbl1.config(anchor=CENTER)
    lbl1.pack()

    txt1 = Entry(display, width=20, show="*")
    txt1.pack()

```

To create the registration page for the password manager as shown in figure 1, I created a function named "firstScreen()", then I put a geometry of 700x600 size for the first screen. Next, the thing is I want to do is to add a label widget as "lb1","lb2" and entry as "txt" and "txt1" then text "Create Master Key" inside "lbl" is displayed on the screen with text entry where the user creates their master password and then another text "Confirm Master Key" inside "lbl1" is shown to confirm the same master key is entered.

```
#   Designing master key display window
bg = PhotoImage(file='locker.png')
my_label = Label(display, image=bg)
my_label.place(x=0, y=0, relwidth=1, relheight=1)

```

After creating a form to create a master key, I added an image "locker.png" as background "bg" inside the label widget.

```python
# Let's write the database Code where I am using sqlite db to store password locally
with sqlite3.connect("password_locker.db") as db:
    cursor = db.cursor()

cursor.execute("""
CREATE TABLE IF NOT EXISTS masterpassword(
id INTEGER PRIMARY KEY,
password TEXT NOT NULL);
""")

cursor.execute("""
CREATE TABLE IF NOT EXISTS vault(
id INTEGER PRIMARY KEY,
website TEXT NOT NULL,
account TEXT NOT NULL,
password TEXT NOT NULL);
""")
```

Let’s create a database to store the account information named "password_locker.db" where ".db" is the extension of the database file. I am using sqlite3 for the database to store information in local storage where cursor control the database as "db.cursor()" and initiate the database. Now, create the tables named "masterpassword" and "vault" inside the database if it does not exist, using the "cursor.execute" command to store the master key and account information respectively..

```python
    def saveMasterkey():
        if txt.get() == txt1.get():
            hashedPassword = hashedPass(txt.get().encode('utf-8'))

            insert_password = """INSERT INTO masterpassword(password)
            VALUES(?) """
            cursor.execute(insert_password, [hashedPassword])
            db.commit()
            lockerScreen()

        else:
            lbl.config(text="Please Re-enter Passwords don't match")

```

As in the above screenshot, I created a function named "saveMasterkey()" which is the command to save the master key. If the password entered in both "txt" and "txt1" variables is equal, then the password is hashed using the md5 hashing, but if it is not, then the user must re-enter the password. In the above code, I gave "hashedPassword" as a string variable where it uses a function named "hashedPass()" to encrypt the master password and then store the encrypted value. Then, I created a command variable "insert_password" to insert the master key into the database inside the "masterpassword" table, and then I used "cursor.execute" to execute the command.

```python
#   Login Page of Ultimate Password Manager

def loginPage():
    display.geometry("700x600")

    lbl = Label(display, text="Enter Master Key")
    lbl.config(anchor=CENTER)
    lbl.pack()

    txt = Entry(display, width=20, show="*")
    txt.pack()
    txt.focus()

    lbl1 = Label(display)
    lbl1.pack()
```

Here, I created a simple login page using the same method as for the registration page. But I added two commands, "getMasterKey" and "checkPswd" inside the login page.

```
def getMasterKey():
    checkhashedpassword = hashedPass(txt.get().encode("utf-8"))
    cursor.execute("SELECT * FROM masterpassword WHERE id = 1 AND password = ?", [checkhashedpassword])

    return cursor.fetchall()

def checkPswd():
    password = getMasterKey()

    if password:
        lockerScreen()

    else:
        txt.delete(0, 'end')
        lbl1.config(text="Sorry Wrong Password")
```

As in the above screenshot of code, the "getMasterKey()" function will check the hash of the master key entered by the user on the login page against the hash of the password stored on the "masterpassword" table inside the database. Then another function, "checkPswd()" will check to match the hashed value of the entered password and the master password.

6. Buttons and its function:

I have created various buttons or features inside this project and now I am going to explain the code used to create those buttons and it's working with some test results.

1st button:

```
btn = Button(display, text="Save",bg= 'yellow', command=saveMasterkey)
btn.pack(pady=5)
```

After filling out the form, the user must save the master password. I created a button using the Button widget with text "Save" and gave it a background yellow colour and linked it with the command "saveMasterkey", which saves the password in the table master password inside the database.

2nd button:

```
btn = Button(display, text="Submit", bg='yellow', command=checkPswd)
btn.pack(pady=5)
```

As shown in the above code, I created a submit button as shown in figure 2, and linked it with the "checkPswd()" function, which checks the password and gives access to the authorised user who has the correct login credentials.

3rd button:

```
#   Ultimate password manager functionalities #

def passGen():
    # Password Generating display.
    display = Tk()

    display.title("Advanced Password Generator")

    myPassword = chr(randint(33,126))
```

This button is a functionality of the password manager which opens a window to create a random password. I gave a title to the password generator and stored the integer value in the "myPassword" variable where the “chr” function will convert the random integer into the ASCII character.

4th button:

```
def newRandom():
    passEntry.delete(0, END)
    pwLength = int(noEntry.get())

    pswd = ""

    for x in range(pwLength):
        pswd += chr(randint(33, 126))

    passEntry.insert(0, pswd)
```

When we enter the password length, click the "generate password" button where "pwLength" is an integer value. I created a "newRandom()" function and assigned the empty variable the name "pswd". As in the above code, the loop continues until the password length, where "randint" generates a random integer that will be added to “pswd” until the loop ends.

5th button:

```
def clipper():
    display.clipboard_clear()
    display.clipboard_append(passEntry.get())
    tkinter.messagebox.showinfo("Random Password","Copied Successfully.")

def copyAccount(input):
    display.clipboard_clear()
    display.clipboard_append(input)
    tkinter.messagebox.showinfo("Account","Account Copied Successfully.")

def copyPassword(input):
    display.clipboard_clear()
    display.clipboard_append(input)
    tkinter.messagebox.showinfo("Password","Password Copied Successfully.")
```

In my project, I created three buttons that use copy to clipboard functionality to copy the account and passwords, making it easier for users to login in any application.I made three buttons for clipping the information: "clipper," "copyAccount," and "copyPassword," and I used "display.clipboard_append()" to store it in the clipboard.

6th button:

```
def lockerScreen():
    for widget in display.winfo_children():
        widget.destroy()

    def newEntry():
        firstText = "Website"
        secondText = "Account"
        thirdText = "Password"

        website = popUp(firstText)
        account = popUp(secondText)
        password = popUp(thirdText)

        insert_fields = """INSERT INTO vault(website, account, password)
        VALUES(?, ?, ?)"""

        cursor.execute(insert_fields, (website, account, password))
        db.commit()
        lockerScreen()
```

Inside the "lockerScreen()" function, I created another function named "newEntry" for adding account information. I used the "popUp" function while inserting the account details, which will make data entry easier. I created a command variable "insert_fields" to insert the website, account, password to the database inside the "vault" table, then I used "cursor.execute" to execute the command. I used the "db.commit" function to save the information in the database file.

7th button:

```
def updatePass(input):
    update = "Type new password"
    password = popUp(update)
    update_password = "UPDATE vault SET password = ? WHERE id = ?"
    cursor.execute(update_password, (password, input,))
    db.commit()
    tkinter.messagebox.showinfo("Password","Password Updated Successfully.")
    lockerScreen()
```

To update the password in the database when a user changes their account password in the future, I added the update password functionality. I created a command variable "update_password" to update the password inside the vault table in the "updatePass()" function as shown in the above code. After entering the new password, "cursor.execute()" will execute the command variable to add the password, and the "db.commit()" function will save the password in the database file.

8th button:

```
def deleteEntry(input):
    cursor.execute("DELETE FROM vault WHERE id = ?", (input,))
    db.commit()
    tkinter.messagebox.showinfo("Info","Deleted Successfully.")
    lockerScreen()
```

I created a "deleteEntry()" function to remove the row of specific id from the table. Instead of creating a command variable, I directly executed the "DELETE" command inside "cursor.execute()", which will delete the account information of a particular id from the "vault" table.

7. Button Testing:

I have created and tested multiple buttons while creating my project, "Ultimate Password Manager", and here are the results I got during the testing phase.

Button	Testing Field	Testing Steps	Expected Result	Output	Actual Result	Remarks
1st	"Save"	Click the "Save" button	Dashboard of password manager should be open.	Password manager is opened with its features.	As Expected	PASS
2nd	"Submit"	Click the "Submit" button	If correct master key is entered, application should be open.	Dashboard is opened after entering the correct master key.	As Expected	PASS
3rd	"Generate Password"	Click the "Generate Password" button	Another window of advanced password generator should be open.	Advanced Password Generator is opened with its functionality	As Expected	PASS
4th	"Generate Password"	Click the "Generate Password" button	After entering the desire characters, random password should generate.	Random password of entered length is generated.	As Expected	PASS
5th	"Copy Account, Password to Clipboard"	Click the "Copy to Clipboard, CopyAccount, CopyPass" buttons	Account info and password must be copied to clipboard.	There was email id and password copied in the clipboard.	As Expected	PASS
6th	"Add New Entry"	Click the "Add New Entry" button	Text box asking user to input website, account, and password should be open.	Pop up with simple dialog box having text entry for website, account, and password is opened	As Expected	PASS
7th	"Update"	Click the "Update" button	Text box asking user to update their password should be open.	Pop up with box window having text entry for updating password is open.	As Expected	PASS
8th	"Delete"	Click the "Delete" button	Specific row should be deleted from the password manager	Row containing account information got deleted.	As Expected	PASS

Fig8: Buttons testing and its results

8. Whole Code:

```python
import hashlib
import sqlite3
from functools import partial
from tkinter import *
from tkinter import simpledialog
from tkinter import ttk
from random import randint
import tkinter


```

```python
# Let's write the database Code where I am using sqlite db to store password locally
with sqlite3.connect("password_locker.db") as db:
    cursor = db.cursor()

cursor.execute("""
CREATE TABLE IF NOT EXISTS masterpassword(
id INTEGER PRIMARY KEY,
password TEXT NOT NULL);
""")

cursor.execute("""
CREATE TABLE IF NOT EXISTS vault(
id INTEGER PRIMARY KEY,
website TEXT NOT NULL,
account TEXT NOT NULL,
password TEXT NOT NULL);
""")

```

```
# Create PopUp window

def popUp(text):
    ans = simpledialog.askstring("input string", text)

    return ans

# Initiate Display Window


display = Tk()
display.update()

display.title("Ultimate_Password_Manager")
display.title("Created by Bishal Aryal - Ethical Hacking Batch 30 ")


def hashedPass(input):
    hash1 = hashlib.md5(input)
    hash1 = hash1.hexdigest()

    return hash1

#   Designing master key display window
bg = PhotoImage(file='locker.png')
my_label = Label(display, image=bg)
my_label.place(x=0, y=0, relwidth=1, relheight=1)

```

```python
def firstScreen():
    display.geometry("700x600")

    lbl = Label(display, text="Create Master Key")
    lbl.config(anchor=CENTER)
    lbl.pack()

    txt = Entry(display, width=20, show="*")
    txt.pack()
    txt.focus()

    lbl1 = Label(display, text="Confirm Master Key")
    lbl1.config(anchor=CENTER)
    lbl1.pack()

    txt1 = Entry(display, width=20, show="*")
    txt1.pack()



    def saveMasterkey():
        if txt.get() == txt1.get():
            hashedPassword = hashedPass(txt.get().encode('utf-8'))

            insert_password = """INSERT INTO masterpassword(password)
            VALUES(?) """
            cursor.execute(insert_password, [hashedPassword])
            db.commit()
            lockerScreen()
```

```python
        else:
            lbl.config(text="Please Re-enter Passwords don't match")

    btn = Button(display, text="Save",bg= 'yellow', command=saveMasterkey)
    btn.pack(pady=5)


```

```python
def loginPage():
    display.geometry("700x600")

    lbl = Label(display, text="Enter Master Key")
    lbl.config(anchor=CENTER)
    lbl.pack()

    txt = Entry(display, width=20, show="*")
    txt.pack()
    txt.focus()

    lbl1 = Label(display)
    lbl1.pack()

    def getMasterKey():
        checkhashedpassword = hashedPass(txt.get().encode("utf-8"))
        cursor.execute("SELECT * FROM masterpassword WHERE id = 1 AND password = ?", [checkhashedpassword])

        return cursor.fetchall()

    def checkPswd():
        password = getMasterKey()

        if password:
            lockerScreen()

        else:
            txt.delete(0, 'end')
            lbl1.config(text="Sorry Wrong Password")

    btn = Button(display, text="Submit", bg='yellow', command=checkPswd)
    btn.pack(pady=5)
```

```python
#   Ultimate password manager functionalities #

def passGen():
    # Password Generating display.
    display = Tk()

    display.title("Advanced Password Generator")

    myPassword = chr(randint(33,126))

    def newRandom():
        passEntry.delete(0, END)
        pwLength = int(noEntry.get())

        pswd = ""

        for x in range(pwLength):
            pswd += chr(randint(33, 126))

        passEntry.insert(0, pswd)

    def clipper():
        display.clipboard_clear()
        display.clipboard_append(passEntry.get())
        tkinter.messagebox.showinfo("Random Password","Copied Successfully.")


    # Label frame for asking user input
    lf = LabelFrame(display, text="How many characters do you need in your password?")
    lf.pack(pady=24)

```

```python
    # Creating Entry Box for number of characters
    noEntry = Entry(lf, font=("Helvetica", 15))
    noEntry.pack(pady=24, padx=24)

    # Now entry box for returned password.
    passEntry = Entry(display, text="", font=("Helvetica", 15), bd=0, bg="systembuttonface")
    passEntry.pack(pady=24)

    # Frame for buttons.
    myFrame = Frame(display)
    myFrame.pack(pady=24)

    # Create buttons
    genButton = Button(myFrame, text="Generate Password", command=newRandom)
    genButton.grid(row=0, column=0, padx=10)

    clipButton = Button(myFrame, text="Copy to Clipboard", command=clipper)
    clipButton.grid(row=0, column=1, padx=10)

```

```python
def lockerScreen():
    for widget in display.winfo_children():
        widget.destroy()

    def newEntry():
        firstText = "Website"
        secondText = "Account"
        thirdText = "Password"

        website = popUp(firstText)
        account = popUp(secondText)
        password = popUp(thirdText)

        insert_fields = """INSERT INTO vault(website, account, password)
        VALUES(?, ?, ?)"""

        cursor.execute(insert_fields, (website, account, password))
        db.commit()
        lockerScreen()

    def updatePass(input):
        update = "Type new password"
        password = popUp(update)
        update_password = "UPDATE vault SET password = ? WHERE id = ?"
        cursor.execute(update_password, (password, input,))
        db.commit()
        tkinter.messagebox.showinfo("Password","Password Updated Successfully.")
        lockerScreen()

```

```python
    def deleteEntry(input):
        cursor.execute("DELETE FROM vault WHERE id = ?", (input,))
        db.commit()
        tkinter.messagebox.showinfo("Info","Deleted Successfully.")
        lockerScreen()

    def copyAccount(input):
        display.clipboard_clear()
        display.clipboard_append(input)
        tkinter.messagebox.showinfo("Account","Account Copied Successfully.")

    def copyPassword(input):
        display.clipboard_clear()
        display.clipboard_append(input)
        tkinter.messagebox.showinfo("Password","Password Copied Successfully.")

```

```python
    display.geometry("750x550")
    main_frame = Frame(display)
    main_frame.pack(fill=BOTH, expand=1)

    my_canvas = Canvas(main_frame)
    my_canvas.pack(side=LEFT, fill=BOTH, expand=1)

    my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL, command=my_canvas.yview)
    my_scrollbar.pack(side=RIGHT, fill=Y)

    my_canvas.configure(yscrollcommand=my_scrollbar.set)
    my_canvas.bind('<Configure>', lambda e: my_canvas.configure(scrollregion=my_canvas.bbox("all")))

    second_frame = Frame(my_canvas)

    my_canvas.create_window((0, 0), window=second_frame, anchor="nw")

    lbl = Label(second_frame, text="Ultimate_Password_Manager")
    lbl.grid(column=2)

    btn2 = Button(second_frame, text="Generate Password", command=passGen)
    btn2.grid(column=2, pady=10)

    btn = Button(second_frame, text="Add New Entry", command=newEntry)
    btn.grid(column=4, pady=10)

    lbl = Label(second_frame, text="Website")
    lbl.grid(row=2, column=0, padx=40)
    lbl = Label(second_frame, text="Account")
    lbl.grid(row=2, column=1, padx=40)
    lbl = Label(second_frame, text="Password")
    lbl.grid(row=2, column=2, padx=40)
```

```python
    cursor.execute("SELECT * FROM vault")

#   Buttons Layout #

    if cursor.fetchall() is not None:
        i = 0
        while True:
            cursor.execute("SELECT * FROM vault")
            array = cursor.fetchall()

            lbl1 = Label(second_frame, text=(array[i][1]))
            lbl1.grid(column=0, row=i + 3)
            lbl2 = Label(second_frame, text=(array[i][2]))
            lbl2.grid(column=1, row=i + 3)
            lbl3 = Label(second_frame, text=(array[i][3]))
            lbl3.grid(column=2, row=i + 3)
            btn2 = Button(second_frame, text="Copy Account", command=partial(copyAccount, array[i][2]))
            btn2.grid(column=3, row=i + 3, pady=10)
            btn3 = Button(second_frame, text="Copy Password", command=partial(copyPassword, array[i][3]))
            btn3.grid(column=4, row=i + 3, pady=10)
            btn1 = Button(second_frame, text="Update", command=partial(updatePass, array[i][0]))
            btn1.grid(column=5, row=i + 3, pady=10)
            btn = Button(second_frame, text="Delete", command=partial(deleteEntry, array[i][0]))
            btn.grid(column=6, row=i + 3, pady=10)

            i = i + 1

            cursor.execute("SELECT * FROM vault")
            if len(cursor.fetchall()) <= i:
                break
```

```
cursor.execute("SELECT * FROM masterpassword")
if cursor.fetchall():
    loginPage()
else:
    firstScreen()

display.mainloop()
```

(Python Project - Random Password Generator & Password Manager, 2021)

9. Results and Discussion:

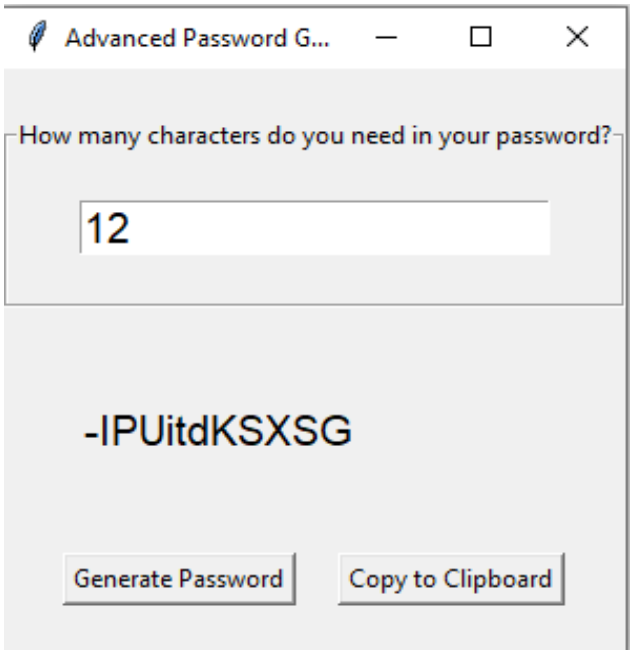

Fig9: Random password of 12 characters is generated

We can create a random password of desire length using password generation feature. As user input 12 character and then click the generate password, random password is displayed on the screen as shown in above figure. After this, user can copy this password to use in different web services and save this password in this password manager.

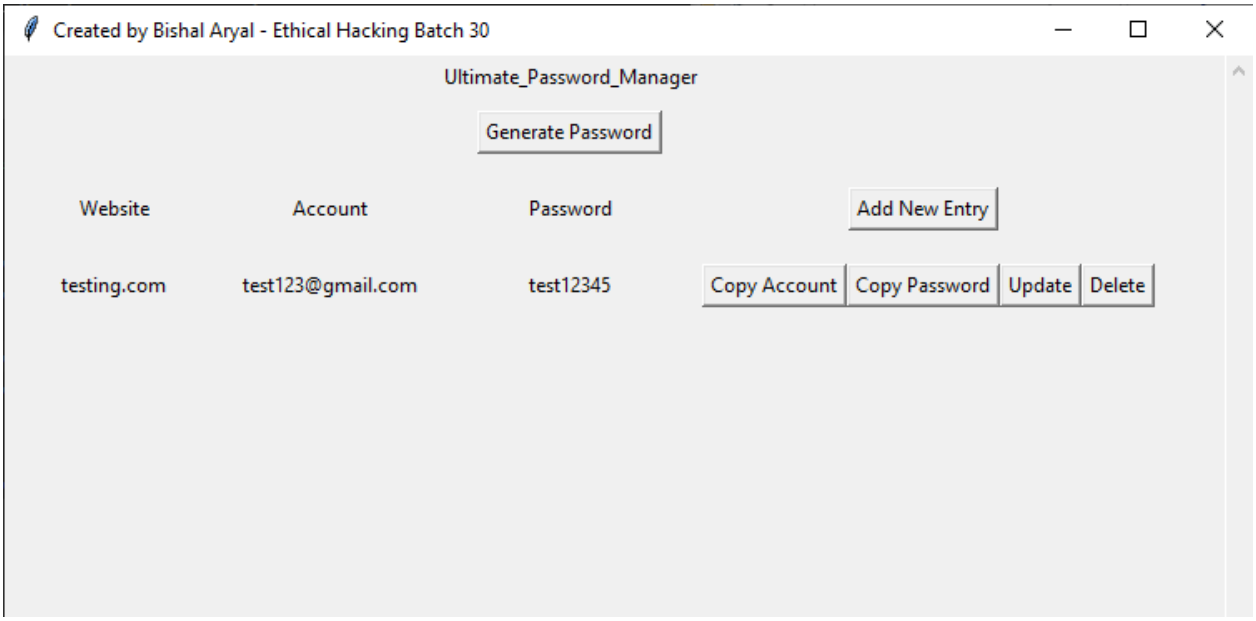

Fig10: Dashboard after new entry of account information

User will store their account information by clicking "Add New Entry" button then the text box is pop up asking user to insert website, account and password as shown in figure 5, 6, and 7 respectively. Then the information will stored in the database file and visible in the dashboard as shown in above figure.

YouTube Video Link of my project:

https://www.youtube.com/watch?v=wRP0toFyp_A

10. Conclusion:

Among the fastest-growing areas in computer science and the technology sector, cyber security has emerged as one of the most promising. The global economy has suffered enormous losses as a result of inadequate security. (Luevanos et al., 2017) Many cyber-attacks in the world occur because of the use of the most common and weak passwords. We can create a secure password and use random passwords in different web services and we just need to remember one master password to access this password manager. After creating this project, I learned how we can create a GUI application using tkinter in python and integrate it with sqlite3 database.

11. References:

Introduction To Python (2021) available from <https://www.w3schools.com/python/python_intro.asp> [24 September 2021]

Python Project - Random Password Generator & Password Manager (2021) available from <https://www.youtube.com/watch?v=PuW6vrcIPvc&t=382s> [24 September 2021]

Luevanos, C., Elizarraras, J., Hirschi, K. and Yeh, J. (2017) *Analysis On The Security And Use Of Password Managers* [online] available from <https://ieeexplore.ieee.org/document/8326801> [24 September 2021]