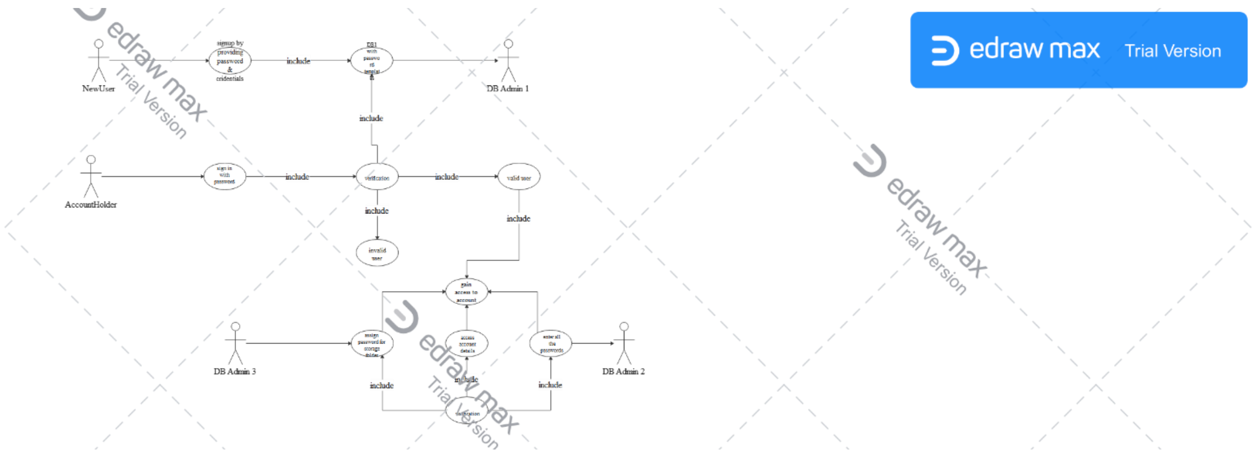
**Mvenhle Shikovha and Lesego Nzimande**

**15942 & 15941**

**Mobile technologies**

**Project Report**



# Password Vault

We will make an app that saves your passwords and usernames for you on a database, with hashing security.

This is an app whereby you can store your passwords to certain application. You don't have to remember your own passwords. You can just write down the application and the password will be given. At first you will need to make a password for the password vault but afterwards you can login to the password vault with the password you set in the beginning.

All your passwords will be in the databases but they will be encrypted.

## Requirements

- Latest version of Pycharm
- Sql lite3
- PythonShell

## Method of Development

The main goal was to create an application which could generate secured passwords and store them into a database and it is stored locally on your system so it's more secure.
If you are like me, you probably have logged in many websites so far. But rarely remember any password of them. And sometimes, the websites won't let you sign up until you add a 'super-strong password.' But as a matter of sorrow, you forgot that password the next day.

There is a simple solution to this gigantic problem. Use a password manager. But nowadays, maintaining privacy in this vast online world is quite impossible unless you don't use the internet at all. But you aren't going to that, are you?

You even may find your password in the dark web selling for a couple of dollars. So, using a third-party password manager doesn't sound safe to me. You should also be concerned about this.

So what is the solution? Create your own ;as programmers we determined that we could create an application that could satisfy our need for privacy and security by just spending time on a computer and creating the best application by ourselves.

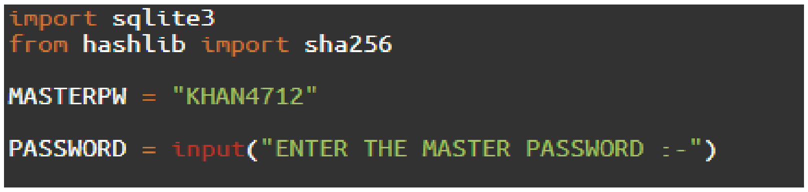
Step1: Install python
Installing Python was by far the easiest step

Step2: Making the environment
Create a folder in your device where you want to set up the environment. Once we made the folder we had to install virtualenv env on the command prompt to setup a virtual environment.

Step 3: Using SQL to create a Database
We created a small database on SPLlite3 with seven main accounts  and locked it with a Primary Key and a master encryption

Step 4: Creating a file in Python

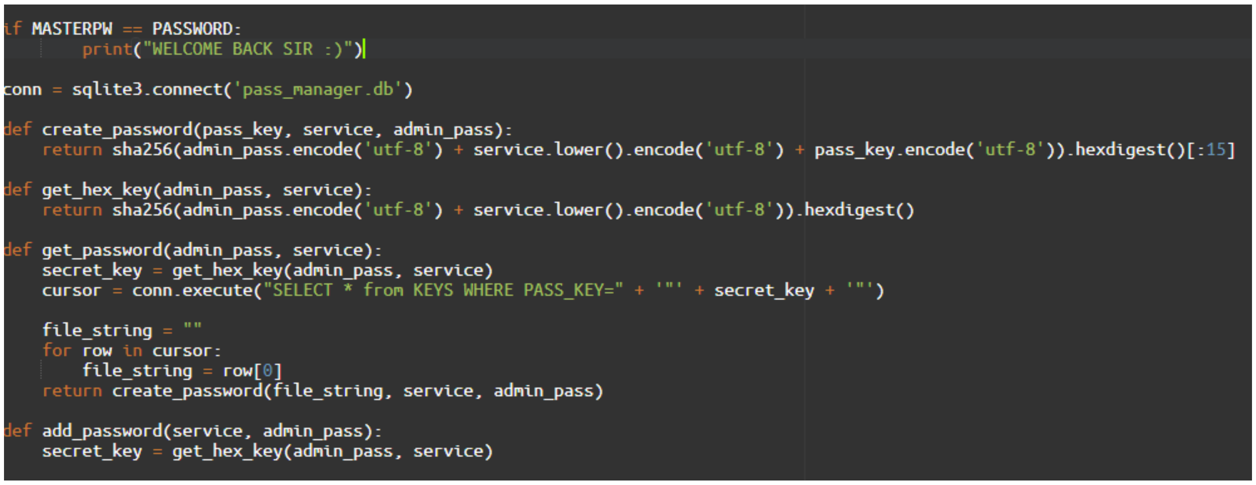
At first we are importing the already existing sqlite3 database into our Python file for the sake of simplicity the Sql database & the application share the same password.



```
import sqlite3
from hashlib import sha256

MASTERPW = "KHAN4712"

PASSWORD = input("ENTER THE MASTER PASSWORD :-")
```

Step 5: Writing Python functions

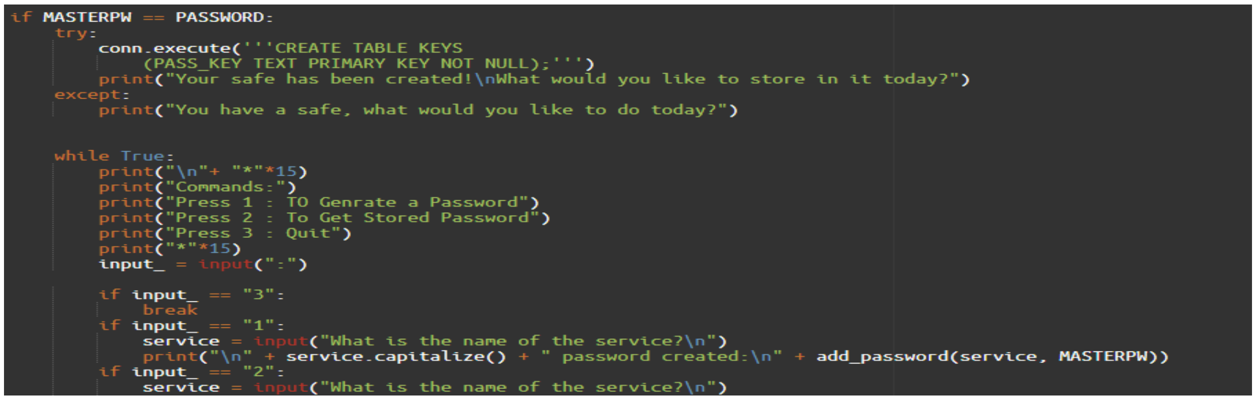
Making use of functional programming principles we intergrated the python scripts with our password database ; our scripts made use of pseudo random algorithms to generate random characters from anwhere between a-z & 0-9, as well as get password and create password functions for the final product.



```
if MASTERPW == PASSWORD:
        print("WELCOME BACK SIR :)")

conn = sqlite3.connect('pass_manager.db')

def create_password(pass_key, service, admin_pass):
    return sha256(admin_pass.encode('utf-8') + service.lower().encode('utf-8') + pass_key.encode('utf-8')).hexdigest()[:15]

def get_hex_key(admin_pass, service):
    return sha256(admin_pass.encode('utf-8') + service.lower().encode('utf-8')).hexdigest()

def get_password(admin_pass, service):
    secret_key = get_hex_key(admin_pass, service)
    cursor = conn.execute("SELECT * from KEYS WHERE PASS_KEY=" + '"' + secret_key + '"')

    file_string = ""
    for row in cursor:
        file_string = row[0]
    return create_password(file_string, service, admin_pass)

def add_password(service, admin_pass):
    secret_key = get_hex_key(admin_pass, service)
```

Step 6: Output the Password

At the end our main focus was the output now that the methods to store and return variabes have been made we have to focus on printing the necessary statements once our required values are displayed on screen.



```
if MASTERPW == PASSWORD:
    try:
        conn.execute('''CREATE TABLE KEYS
            (PASS_KEY TEXT PRIMARY KEY NOT NULL);''')
        print("Your safe has been created!\nWhat would you like to store in it today?")
    except:
        print("You have a safe, what would you like to do today?")


    while True:
        print("\n"+ "*"*15)
        print("Commands:")
        print("Press 1 : TO Genrate a Password")
        print("Press 2 : To Get Stored Password")
        print("Press 3 : Quit")
        print("*"*15)
        input_ = input(":")

        if input_ == "3":
            break
        if input_ == "1":
            service = input("What is the name of the service?\n")
            print("\n" + service.capitalize() + " password created:\n" + add_password(service, MASTERPW))
        if input_ == "2":
            service = input("What is the name of the service?\n")
```

Features:

- Login and Registration service
- Multiple  accounts can be created & stored in this application
- Password Generator (Mix, Alpha-Numercial, Numerical)
- Fully Offline
- Added Encryption to save Passwords

Why need Password Managers?

Every day we go to a new website and surf the Internet. While doing this we create lots of accounts and so far we also havefun forgotten many passwords and so some time's we needed to login again but we got stuck at the fogotten passwords page and finally due to this we have to reset that forgotten password, but it takes up too much of our time.

So to get rid from problem, this appilication has been created so you should not to get stuck while logging in and your work shouldn't stop due to this.

Security

It work's on python so it has great security features so that no one can view , change or remove your password without your permission. It has a master password which only you have to remember it ensures efficient use , so feel safe to use it. As it work's offline your data will not be hacked by hacker, your data is secure in your computer hard disc.

Advantages

The advantage of password-based access controls is that they are easily incorporated in most software using APIs available in many software products, they require no extensive computer/server modifications, and that users are already familiar with the use of passwords. While passwords can be fairly secure, the weakness is how users choose and manage them, by using:

- simple passwords – short in length, that use words found in dictionaries, or do not mix in different character types (numbers, punctuation, upper/lower case), or are otherwise easily guessable
- passwords others can find – on sticky notes on monitors, in a notepad by the computer, in a document on the computer, whiteboard reminders, smart device storage in clear text, etc.
- the same password – using the same password for multiple sites, never changing account passwords, etc.

- shared passwords – users telling others passwords, sending unencrypted emails with password information, contractors using same password for all their accounts, etc. administrative account logins where limited logins would suffice, or administrators who allow users with the same role to use the same password.

  These are the issues we wanted to address in our project.