COMPUTER SCIENCE PROJECT

**PASSWORD MANAGER**



**INDEX**

|  |  |
| --- | --- |
| **S.L**  **NO.** | **TOPIC** |
| **1)** | **CERTIFICATE** |
| **2)** | **ACKNOWLEDGE** |
| **3)** | **PACKAGE/MODULE USED IN PROJECT** |
| **4)** | **SOURCE CODE** |
| **5)** | **OUTPUT** |
| **6)** | **LIMITATIONS** |
| **7)** | **REQIREMENTS** |
| **8)** | **BIBLIOGRAPHY** |

**PACKAGE / MODULE USED IN PROJECT**

**1. Registration:**

User registration is the process of allowing individuals to create accounts within your password manager application. It's a critical step for ensuring that users have a secure and personalized experience.

* **User Input Validation**: When a user registers, validate the input data for correctness and security. Ensure that the chosen username is unique and that the password meets security criteria (length, complexity, etc.).
* **Database Interaction**: Upon successful validation, store the user's registration information securely in a MySQL database. This typically includes the username and a securely hashed password.
* **Security Measures**: Implement security measures like salting and hashing to protect user credentials. Never store plain-text passwords in the database. Instead, store the hash of the password, making it computationally infeasible to reverse-engineer the original password.
* **User Feedback**: Provide feedback to the user during the registration process. Inform them if the chosen username is already taken or if their password doesn't meet the required criteria.

**2. Core Operations:**

Core operations in a password manager are the functionalities that enable users to securely manage their passwords.

* **Password Storage**: When a user adds a new password entry, encrypt or hash the password before storing it in the database. This ensures that even if the database is compromised, the actual passwords remain secure.
* **Password Retrieval**: Allow users to retrieve their stored passwords when needed. Decrypt or hash the stored password and present it to the user with proper authentication.
* **Password Updates**: Enable users to update or change their stored passwords. When a password is changed, ensure that it's securely replaced in the database.
* **Security Measures**: Continuously monitor and improve security practices to protect stored passwords. Implement features like automatic password expiration and reminders for users to update their passwords periodically.

**3. Profile Management:**

Profile management allows users to customize their experience within the password manager and manage their personal information.

**Key Concepts**:

* **Password Reset**: Implement a secure password reset mechanism for users who forget their passwords.
* **Preferences**: Allow users to customize their password manager preferences, such as specifying a default password length or enforcing specific password policies.
* **Account Deactivation/Deletion**: Provide options for users to deactivate or delete their accounts if they choose to discontinue using the password manager. Implement these features securely to protect user data.

**Source code**

import mysql.connector

db = mysql.connector.connect(

host="localhost",

user="root",

passwd="kasim108",

database="passwd",

auth\_plugin='mysql\_native\_password'

)

print("PLEASE MAKE SURE THAT YOU HAVE RUN \"setup.py\" BEFORE!!!!!")

print(db)

print("Successfully connected to the database")

mycursor = db.cursor()

def mainmenu():

print(" ")

print("WELCOME TO PASSWORD MANAGER")

print("MAIN MENU:")

print(" ")

print("Enter \"See all\" to show all the passwords.")

print("Enter \"New\" to enter a new password.")

print("Enter \"Delete\" to delete a password.")

print("Enter \"Search\" to search for any password.")

print("Enter \"Reset\" to reset the whole database and to remove all the passwords.")

print(" ")

user\_input = input("Please enter option from the above given menu: ")

if user\_input == "See all":

see\_allpasswds()

elif user\_input == "New":

add\_newpasswd()

elif user\_input == "Delete":

delete\_passwd()

elif user\_input == "Search":

search()

elif user\_input == "Reset":

reset\_database()

else:

print("ERROR: The option you have chosen is incorrect")

print("Please choose the option from the given menu!!!")

print(" ")

print("Enter \"Back\" to go back to the menu.")

userInput = input("Enter option: ")

if userInput == "Back":

menu()

else:

print("ERROR:Wrong input please rerun the program")

def see\_allpasswds():

print(" ")

mycursor.execute("SELECT \* FROM passwords")

results = mycursor.fetchall()

for x in results:

print(x)

print("All the data in the database.")

print(" ")

print("Enter \"Back\" to go back to the menu.")

userInput = input("Enter option: ")

if userInput == "Back":

menu()

else:

print("ERROR:Wrong input please rerun the program.")

def add\_newpasswd():

print(" ")

print("Add data accordingly as given.")

websites = input("Enter Website: ")

usernames = input("Enter Username or an Email\_Id: ")

passwords = input("Enter the password: ")

dates = input("Enter the date when password created/uploaded in yyyy/mm/dd format: ")

sql = "INSERT INTO passwords (Website, Username, Password, Date) VALUES (%s, %s, %s, %s)"

value = (websites, usernames, passwords, dates)

mycursor.execute(sql, value)

db.commit()

print("Your password has been inserted into the database!!!")

print(" ")

print("Do you want to enter more passwords? ")

print("Enter \"YES\" if you want to add.")

print("Enter \"NO\" to go back to mainmenu.")

userInput = input("Enter option: ")

if userInput == "YES":

add\_newpasswd()

elif userInput == "NO":

menu()

else:

print("ERROR:Wrong Input")

print("Enter \"Back\" to go back to the menu.")

userInput = input("Enter option: ")

if userInput == "Back":

menu()

else:

print("ERROR:Wrong input please rerun the program")

def delete\_passwd():

print("So you want to delete a password?")

userInput = input("Enter \"YES\" OR \"NO\": ")

if userInput == "YES":

delete = input("Enter the website you want to delete the password for: ")

sql = "DELETE FROM passwords WHERE website = %s"

val = (delete,)

mycursor.execute(sql, val)

db.commit()

print("The data saved in the website you choose has been deleted from database.")

print(" ")

print("Do you want to delete again? ")

print("Enter \"YES\" to delete again.")

print("Enter \"NO\" to go back to the mainmenu")

user\_input = input("Enter option: ")

if user\_input == "YES":

delete\_passwd()

elif user\_input == "NO":

menu()

else:

print("ERROR:Wrong Input")

print("Enter \"Back\" to go back to the menu.")

userInput = input("Enter option: ")

if userInput == "Back":

menu()

else:

print("ERROR:Wrong input please rerun the program")

elif userInput == "NO":

print("Enter \"Back\" to go back to the menu.")

userInput = input("Enter option: ")

if userInput == "Back":

menu()

else:

print("ERROR:Wrong input please rerun the program")

else:

print("Error:Wrong Input")

print("Enter \"Back\" to go back to the menu.")

userInput = input("Enter option: ")

if userInput == "Back":

menu()

else:

print("ERROR:Wrong input please rerun the program")

def search():

print(" ")

print("You can now choose how you want to search the database")

print("Enter \"Website\" if you want to search for a website")

print("Enter \"Username\" if you want to search for a username")

print("Enter \"Password\" if you want to search for a password")

searchInput = input("Enter your option: ")

if searchInput == "Website":

website\_search()

elif searchInput == "Username":

username\_search()

elif searchInput == "Password":

passwd\_search()

else:

print("ERROR:Wrong Input")

print("Enter \"Back\" to go back to the menu.")

userInput = input("Enter option: ")

if userInput == "Back":

menu()

else:

print("ERROR:Wrong input please rerun the program")

def website\_search():

print("You have choose to search the database using website name.")

searchWebsite = input("Enter website: ")

sql = "SELECT \* FROM passwords WHERE Website = %s"

val = (searchWebsite,)

mycursor.execute(sql, val)

sresult = mycursor.fetchall()

for x in sresult:

print(x)

print(" ")

print("Do you want to search database again.")

print("Enter \"YES\" to search again.")

print("Enter \"NO\" to go back to the menu")

userInput = input("Enter option: ")

if userInput == "YES":

search()

elif userInput == "NO":

menu()

else:

print("ERROR:Wrong Input")

print("Enter \"Back\" to go back to the menu.")

userInput = input("Enter option: ")

if userInput == "Back":

menu()

else:

print("ERROR:Wrong input please rerun the program")

def username\_search():

print("You have choose to search the database using username.")

searchUsername = input("Enter Username: ")

sql = "SELECT \* FROM passwords WHERE Username = %s"

val = (searchUsername,)

mycursor.execute(sql, val)

sresult = mycursor.fetchall()

for x in sresult:

print(x)

print(" ")

print("Do you want to search database again.")

print("Enter \"YES\" to search again.")

print("Enter \"NO\" to go back to the menu")

userInput = input("Enter option: ")

if userInput == "YES":

search()

elif userInput == "NO":

menu()

else:

print("ERROR:Wrong Input")

print("Enter \"Back\" to go back to the menu.")

userInput = input("Enter option: ")

if userInput == "Back":

menu()

else:

print("ERROR:Wrong input please rerun the program")

def passwd\_search():

print("You have choose to search the database using password.")

searchPasswd = input("Enter Password: ")

sql = "SELECT \* FROM passwords WHERE Password = %s"

val = (searchPasswd,)

mycursor.execute(sql, val)

sresult = mycursor.fetchall()

for x in sresult:

print(x)

print(" ")

print("Do you want to search database again.")

print("Enter \"YES\" to search again.")

print("Enter \"NO\" to go back to the menu")

userInput = input("Enter option: ")

if userInput == "YES":

search()

elif userInput == "NO":

menu()

else:

print("ERROR:Wrong Input")

print("Enter \"Back\" to go back to the menu.")

userInput = input("Enter option: ")

if userInput == "Back":

menu()

else:

print("ERROR:Wrong input please rerun the program")

def reset\_database():

print("Enter \"RESET\" if you want to reset database.")

print("Enter \"Back\" if you want to go back to menu.")

reset = input("Enter Option: ")

if reset == "RESET":

database\_reset()

elif reset == "BACK":

menu()

else:

print("ERROR:Wrong Input")

def database\_reset():

mycursor.execute("DELETE \* FROM passwords")

db.commit()

print("Your database has been completely erased.")

print(" ")

print("Enter \"Back\" to go back to the menu.")

userInput = input("Enter option: ")

if userInput == "Back":

menu()

else:

print("ERROR:Wrong input please rerun the program")

def app\_exit():

print("Thank You for using password manager!")

print("BYE SEE YOU SOON")

def menu():

print(" ")

print("MAIN MENU:")

print("Enter \"See all\" to show all the passwords.")

print("Enter \"New\" to enter a new password.")

print("Enter \"Delete\" to delete a password.")

print("Enter \"Search\" to search for any password.")

print("Enter \"Reset\" to reset the whole database and to remove all the passwords.")

print("Enter \"Exit\" to exit the application.")

print(" ")

user\_input = input("Please enter option from the above given menu: ")

if user\_input == "See all":

see\_allpasswds()

elif user\_input == "New":

add\_newpasswd()

elif user\_input == "Delete":

delete\_passwd()

elif user\_input == "Search":

search()

elif user\_input == "Reset":

reset\_database()

elif user\_input == "Exit":

app\_exit()

else:

print("ERROR: The option you have chosen is incorrect")

print("Please choose the option from the given menu!!!")

print(" ")

print("Enter \"Back\" to go back to the menu.")

userInput = input("Enter option: ")

if userInput == "Back":

menu()

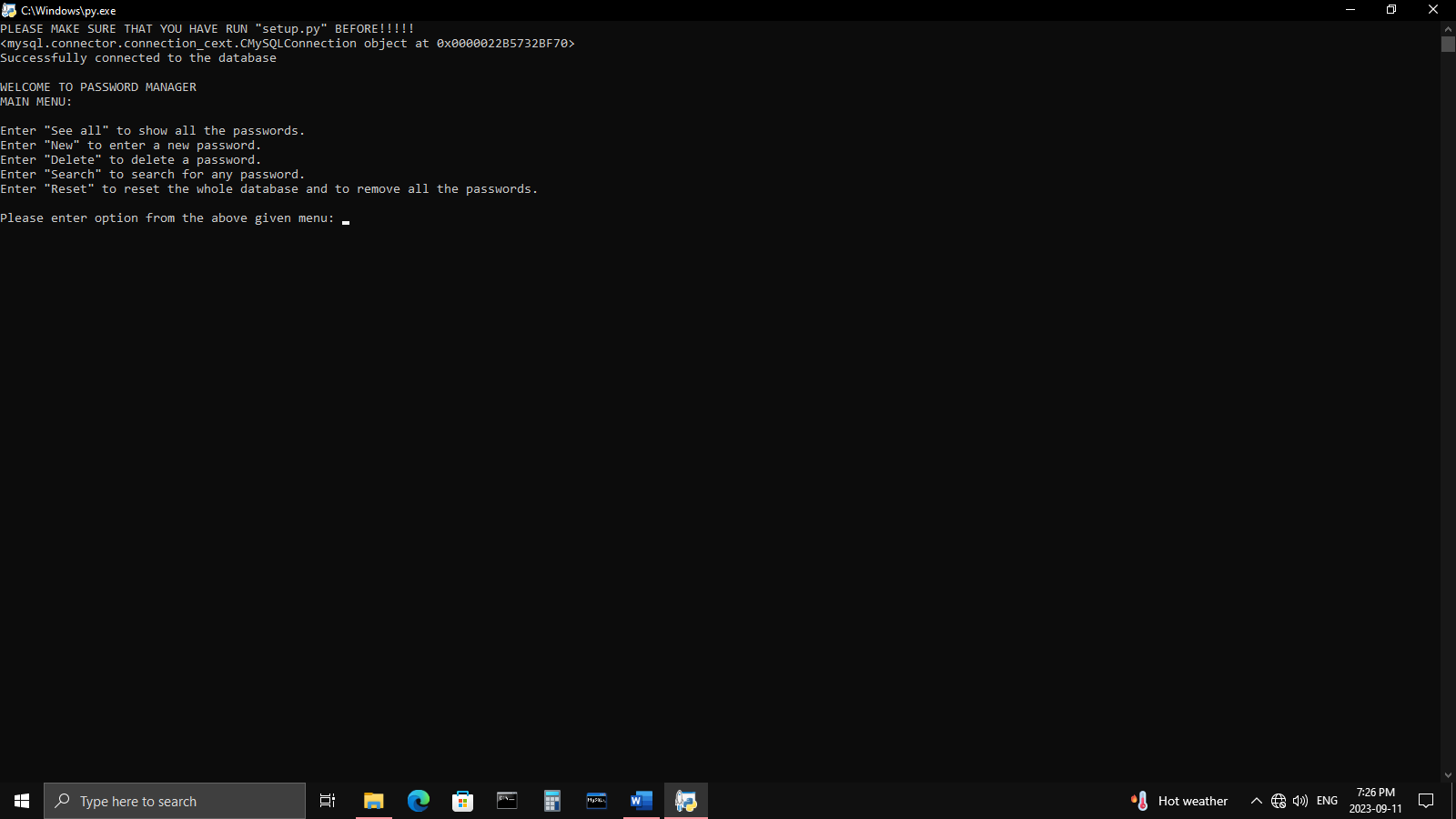
else:

print("ERROR:Wrong input please rerun the program")

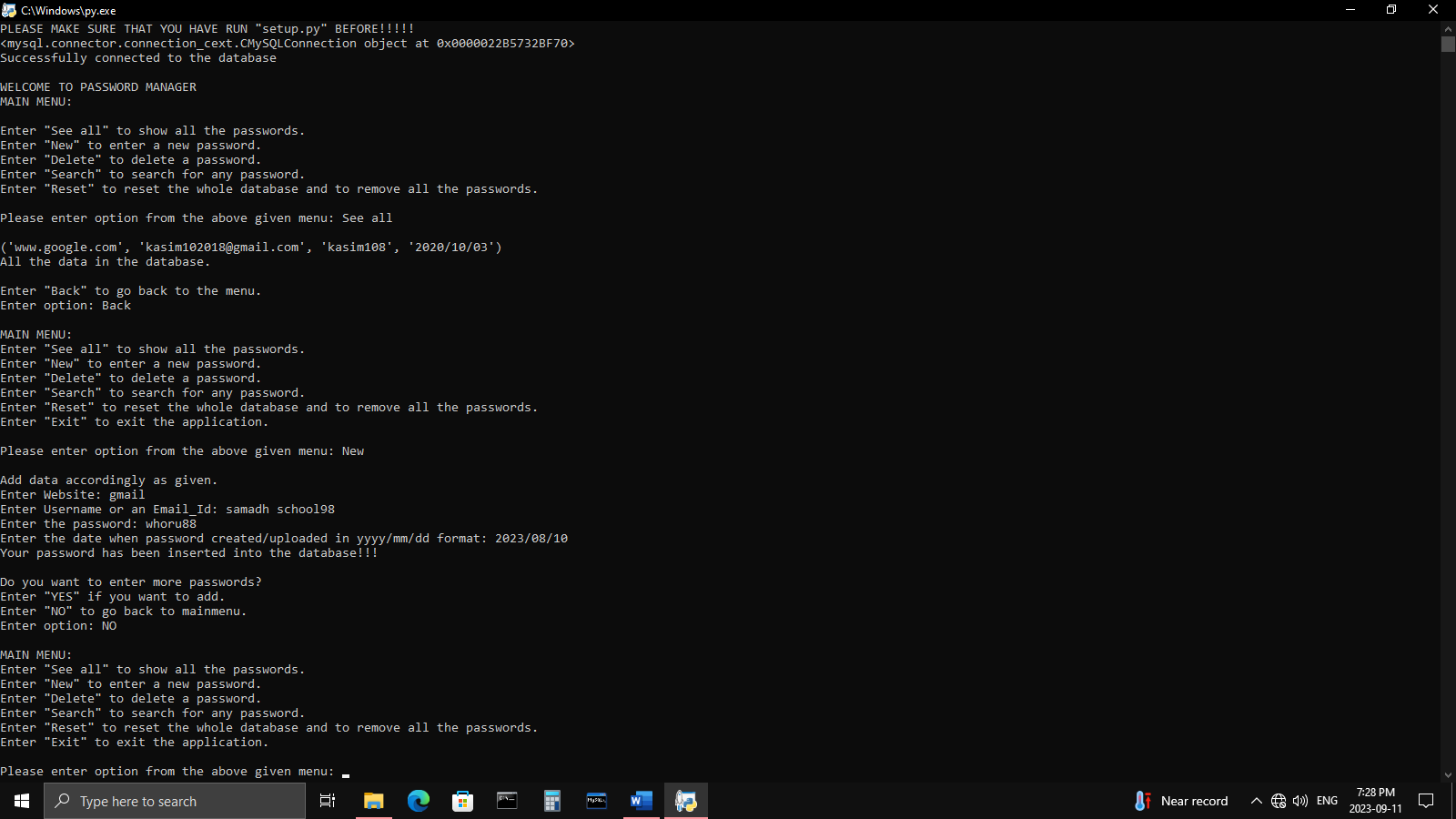
mainmenu()

**OUTPUT**

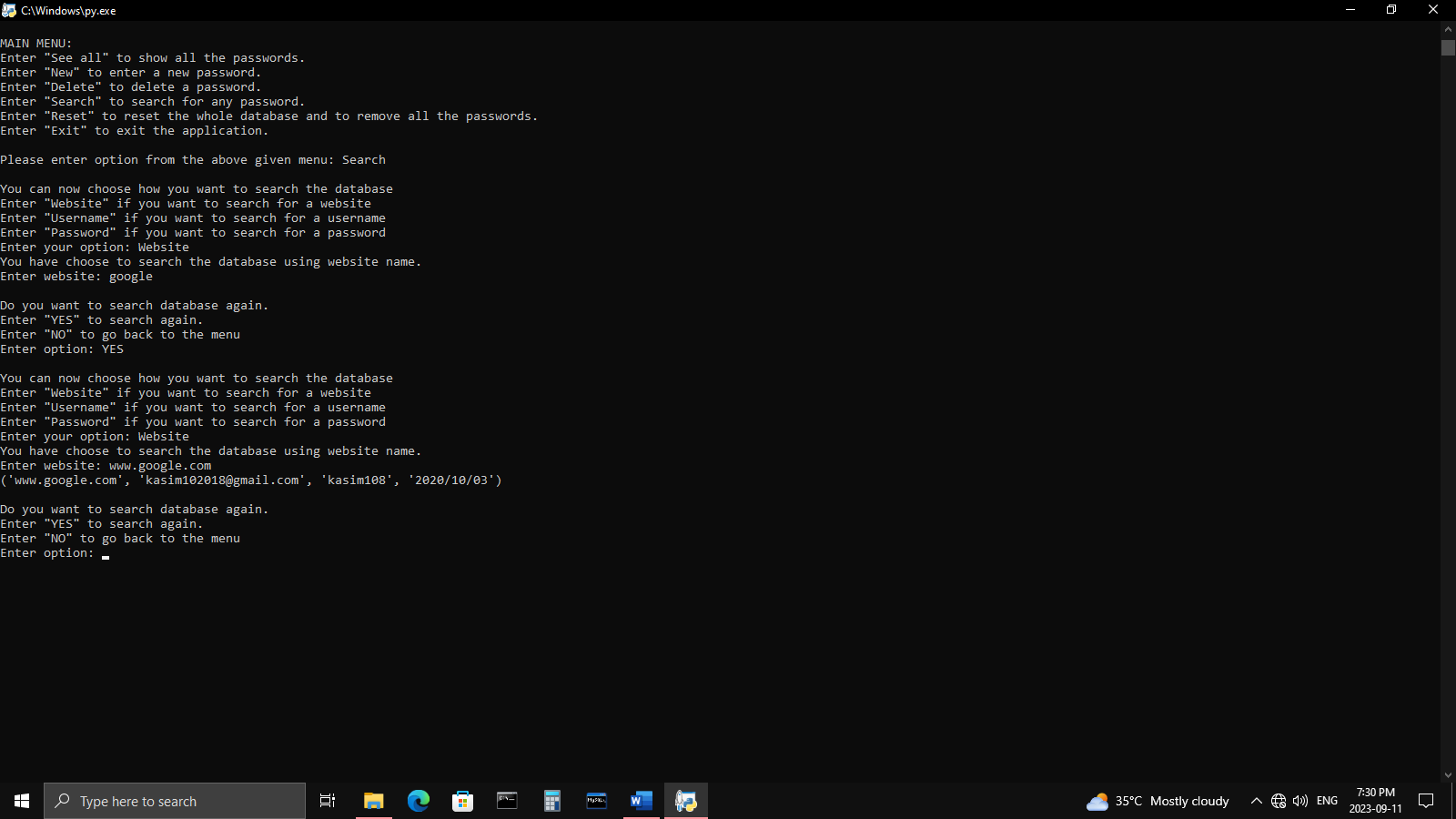
MAIN MENU



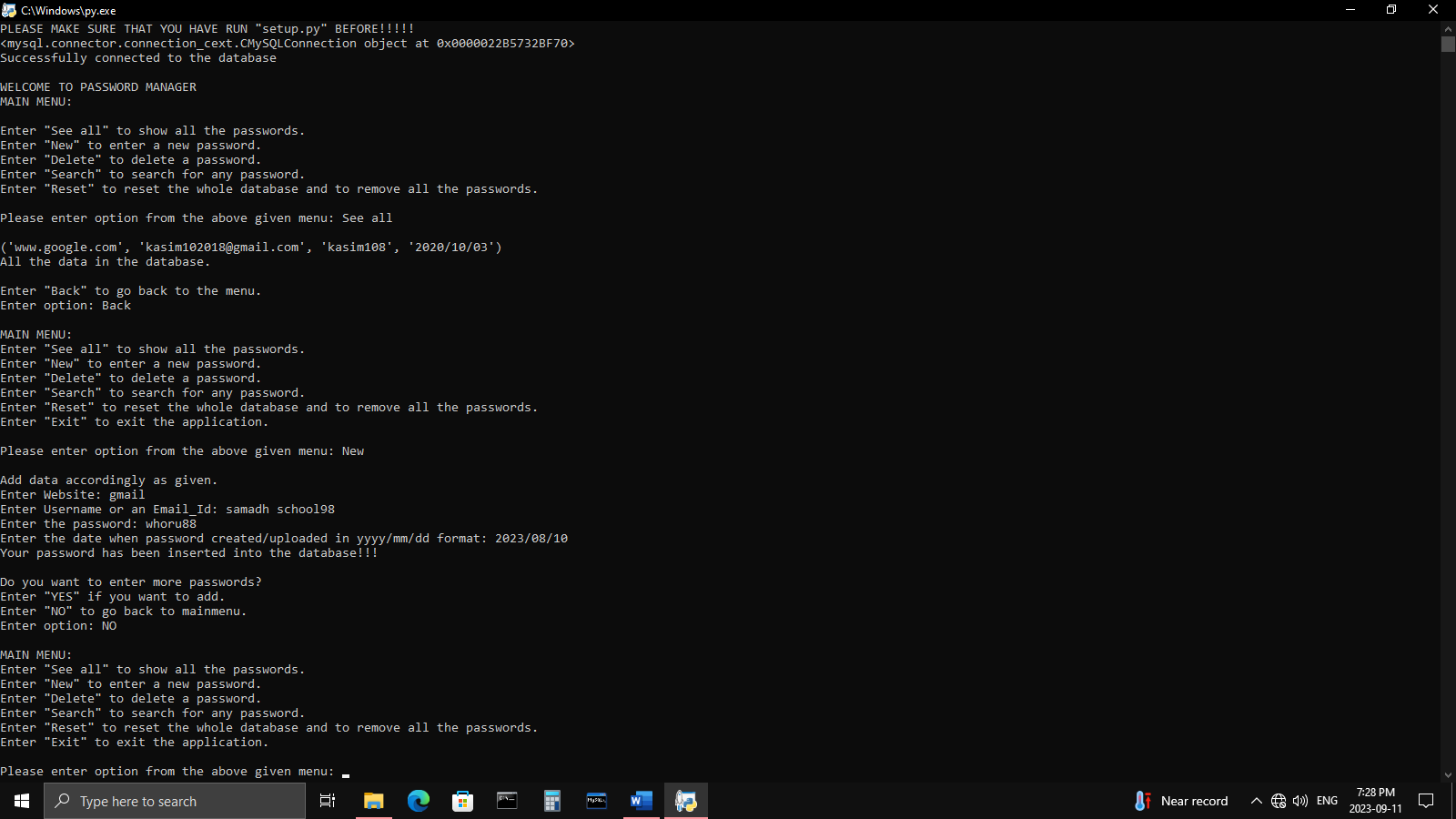
View



search



Add



Database

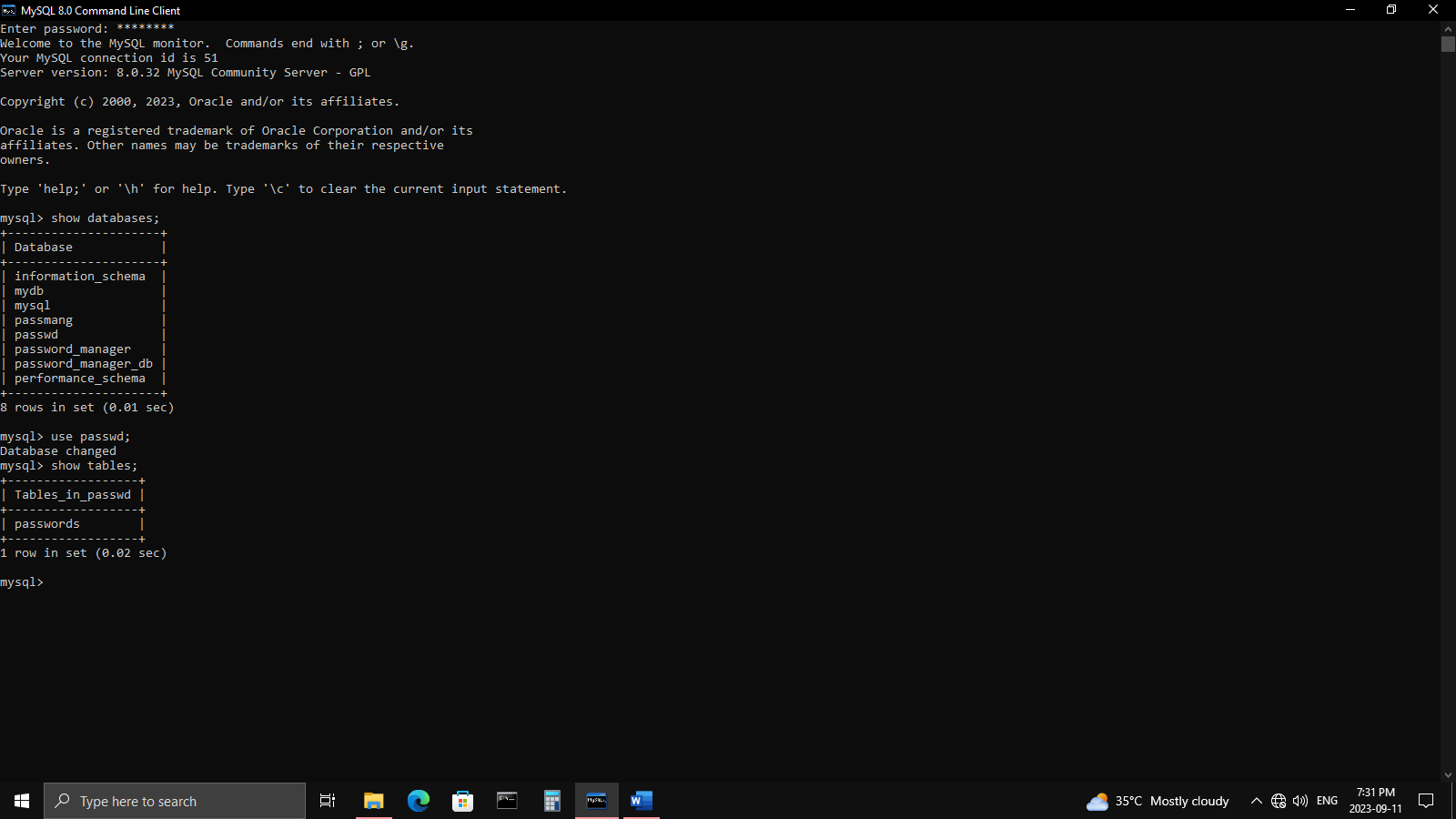
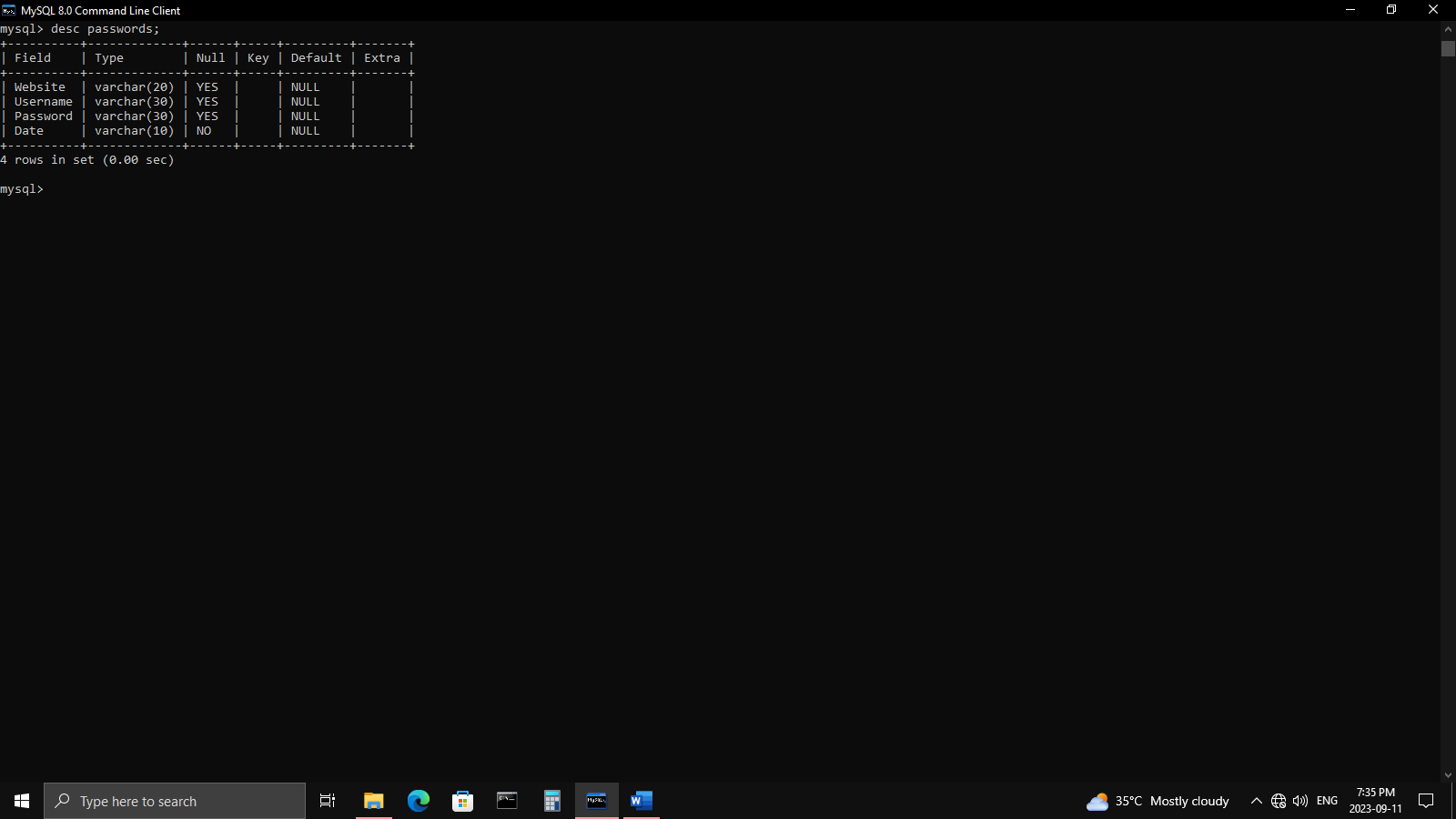
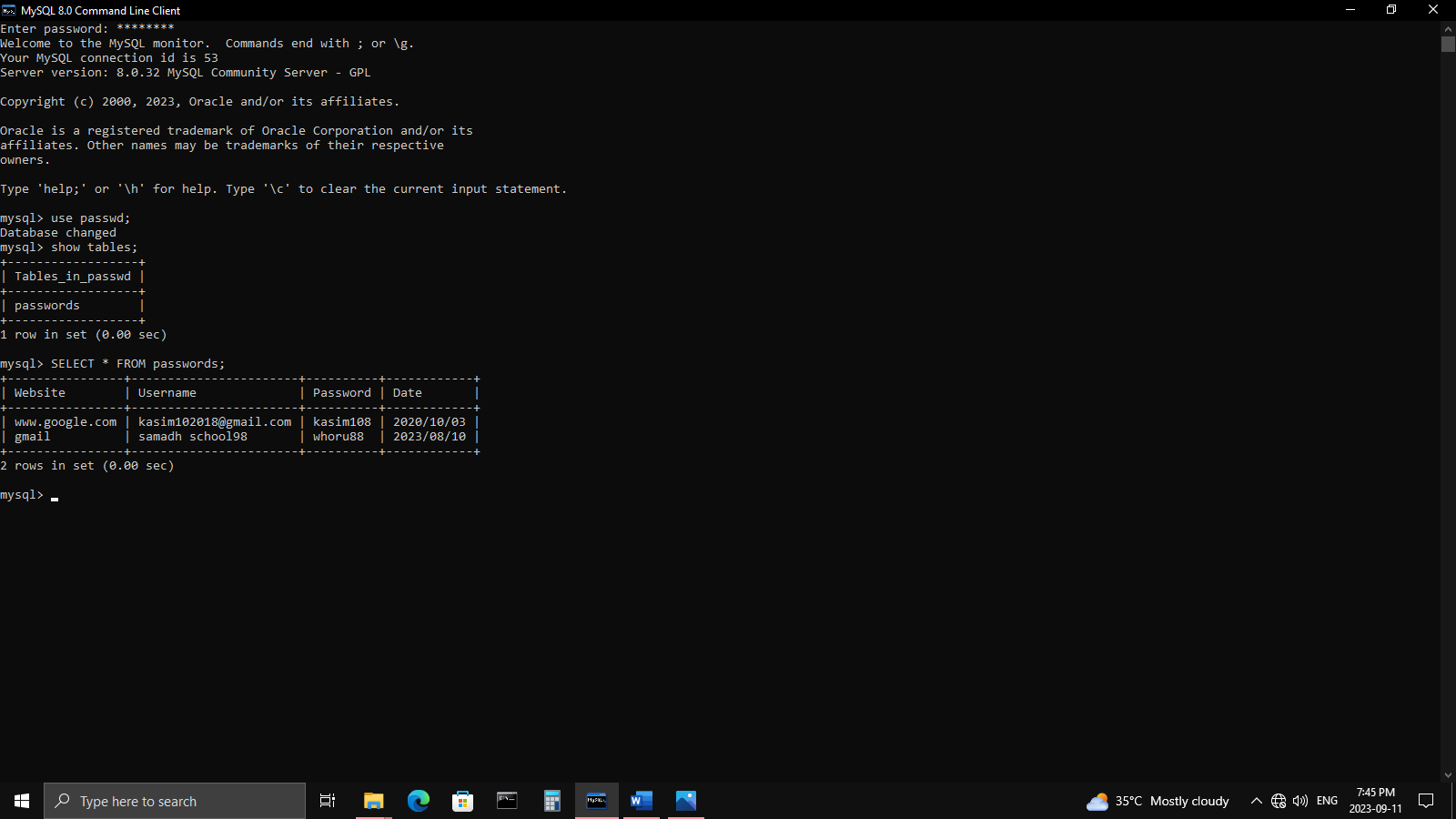


Table structure



Data stored



**LIMITATIONS**

**1. Devices and browser support:**

Different tools support different devices and software, so it’s important to choose a password manager that works well with your needs. Some managers do not work on every device, while others only work in specific web browsers. In these cases, support can be considered a drawback. However, this can usually be avoided entirely by choosing the right password management tool.

**2. A single point of failure:**

Password managers ask you to set a master password that is used to access any site or service that the tool manages. This master password is incredibly important, so it does create a single point of failure. If a user loses their master password or other identifying information, they could lose access to all of their passwords all at once. Likewise, if your master password fell into the wrong hands, it would allow a bad actor to access any account saved in the password manager.

**3. Some do not work with all websites:**

Again, this comes down to choosing the right tool for your use case. Some password managers do not work correctly on all websites. They may not recognize that a login is being requested, or they may not autofill a login form. In these cases, the user will either need to trigger the password manager manually or input their login and password without the help of the tool.

**4. False sense of security:**

The downfall of many security solutions is that they can lull users into a false sense of security. It’s not uncommon for any solution or tool to be viewed as a “silver bullet” or cure all for security. As such, the end users may feel less of a need to self-police their habits. For example, users may stop practicing phishing awareness because they feel their accounts are protected. While password managers are a beneficial addition to a security stack, clients and end users should be made aware that it doesn’t eliminate the need for other cybersecurity measures. Especially awareness training and vigilance.

**5. Poorly-protected managers:**

Password managers can be a security threat if they do not encrypt their data. Hackers know that compromising a password manager is like getting the keys to the castle. Because of this a strong encryption must be in place to prevent access to your saved passwords. The best password tools — such as Password Boss — use AES 256 bit encryption providing a high level of security.

Before we move into “pros”, it’s important to remember that not all password manager tools are the same. Some of these pros or cons of using a password manager will not apply to every option (such as in the example of unencrypted password tools.)

**PROJECT REQUIREMENTS:**

**SOFTWARE**

* MY SQL SERVER
* WINDOWS OS (7 AND +)

**HARDWARE**

* PROCESSOR: AMD OR INTEL
* RAM: 4GB (MINIMUM)
* STROAGE: 128GB (MINIMUM)

**BIBILIOGRAPHY**

* 1. COMPUTER SCIENCE CLASS 12: PREETI ARORA
  2. python-forum.io
  3. pythonworld.in
  4. github.com
  5. google.com