**Code:**

Data scheme design using SQL:

CREATE TABLE Users (

UserID INT PRIMARY KEY,

Username VARCHAR(50) UNIQUE,

EncryptedMasterPassword VARCHAR(256) -- Use a strong hashing algorithm

);

CREATE TABLE Passwords (

PasswordID INT PRIMARY KEY,

UserID INT,

ServiceName VARCHAR(50),

EncryptedPassword VARCHAR(256), -- Use strong encryption (AES, bcrypt, etc.)

FOREIGN KEY (UserID) REFERENCES Users(UserID)

);

Password Generation Function Using python:

import random

import string

def generate\_password(length=12, uppercase=True, digits=True, special\_chars=True):

characters = string.ascii\_lowercase

if uppercase:

characters += string.ascii\_uppercase

if digits:

characters += string.digits

if special\_chars:

characters += string.punctuation

return ''.join(random.choice(characters) for \_ in range(length))

Password Strength Checker using python:

def check\_password\_strength(password):

# Check length

if len(password) < 8:

return "Weak: Password is too short."

# Check for uppercase, lowercase, digits, and special characters

has\_uppercase = any(c.isupper() for c in password)

has\_lowercase = any(c.islower() for c in password)

has\_digit = any(c.isdigit() for c in password)

has\_special\_char = any(c in string.punctuation for c in password)

if not (has\_uppercase and has\_lowercase and has\_digit and has\_special\_char):

return "Weak: Password lacks complexity."

return "Strong: Password meets criteria."

# Example usage:

password = generate\_password()

strength\_result = check\_password\_strength(password)

print(f"Generated Password: {password}")

print(f"Password Strength: {strength\_result}")