

StrongPass

Password Generator



Secure. Customizable. Reliable.

TEAM MEMBERS

Ulyana Clarke

Marcell Placencia

Christian Colon

Zeyon Charles

Problem at Hand

The Password Problem

- Most built-in password generators (Apple, Google) lock users into specific ecosystems
- Users have little to no control over password composition or character types
- Weak or reused passwords remain the #1 cause of account compromise
- No easy way to check if a generated password has been previously breached
- Storing passwords securely remains a challenge for everyday users

#1

Weak passwords are the leading cause of breaches

80%

Of attacks involve stolen credentials

60%

Of users reuse passwords across accounts

Supporting Statistics

~80%

of breaches involve stolen credentials

Verizon DBIR 2025 / SpyCloud 2025

3.1B

passwords exposed in 2024 — a 125% increase from the prior year

SpyCloud Identity Exposure Report 2025

85%

of users reuse passwords across multiple sites

Bitwarden World Password Day Survey 2023

<1s

to brute force common passwords like '123456', 'password', and 'abc123'

Hive Systems Password Table 2024

Our Motivation



Promote Better Practices

Our motivation is to promote and educate users on strong password practices to reduce breaches and improve overall security.



Reduce Breaches

By automating the creation of complex, unique passwords, we directly reduce the risk of credential-based attacks.



A More Secure Digital World

Leading to a more secure digital environment for everyone — not just technical users, but everyday people too.

What Is Our Solution?

Our solution was to create a password manager that not only creates a secure password but also stores and manages other passwords for applications.

- Generates strong, customizable passwords
- Stores and manages passwords for any application
- Profile-specific password vaults
- Breach detection via HaveIBeenPwned API
- Strength meter for every generated password
- Save credentials to vault with one click
- Installed locally on your system for easy, secure access

TECH STACK

Python 3.13

Tkinter & CustomTkinter

SQL Server

Encryption via Cryptography.fernet

HaveIBeenPwned API

Hashlib (password hashing)

Pyodbc (database connector)

GitHub

What Is Its Value?



Automated Security

The value of our solution is that it automates the password creation process with a higher degree of security than users would achieve on their own.



No More Remembering

Users can have complex, unique passwords without needing to remember them — the built-in password vault handles it all.



Platform Freedom

Runs anywhere Python is installed. No browser required, no ecosystem lock-in, no cloud dependency.



Built to Grow

Password expiration, cross-check, and multi-user features are already in the roadmap. Strong Pass is a foundation, not a finished product.

Software Functionality



Create
Account

Customize
Password

Generate
& Evaluate

Breach
Check

Save to
Vault

Account Setup

Users create a profile with a master password. SHA-256 hashed credentials stored in SQL Server.

Customization

Specify number of letters, numbers, symbols, and include an optional keyword in the password.

Breach Detection

There will be a strength meter for the password which will also mention if it is found in a breach (cross-references with haveibeenpwned).

Vault Storage

Save credentials to profile-specific vault.

Importing/Exporting credentials

Examples



Strong Pass Generator

Your digital fortress

SECURE LOGIN

Create Master Account



Strong Pass Generator

Sign Out

@n6#yUIoxR9{v07Aq[eo

SECURE

Letters

Numbers

12

4

Symbols

Custom Keyword

4

GENERATE SECURE PASSWORD

Save

Vault

Secured Vault

Import JSONExport JSON

Apple | marcell

Copy

Edit

Delete

Google | marcellp

Copy

Edit

Delete

FIU | 6338011

Copy

Edit

Delete

Details

Service / Application Name

Apple

Username

marcel|

Password



Regenerate Password

SAVE CREDENTIALS

Code Snippets

```
def evaluate_strength(self, pwd):
    """Returns a score from 0.0 to 1.0 based on password complexity."""
    # Simple algorithm to give a rough "score" of how good the password is
    score = 0.0
    if len(pwd) >= 8: score += 0.2
    if len(pwd) >= 12: score += 0.3
    if any(c.islower() for c in pwd): score += 0.1
    if any(c.isupper() for c in pwd): score += 0.1
    if any(c.isdigit() for c in pwd): score += 0.15
    if any(c in "!@#$%^&*()-_+=[]{}|;:,<>?" for c in pwd): score += 0.15
    return min(score, 1.0)
```

Snippet 2

```
def save(self):
    app, user, pwd = self.in_app.get(), self.in_user.get(), self.in_pwd.get()
    if not app or not user: return messagebox.showwarning("Validation", "Required fields are missing.", parent=self)

    conn = get_db_connection()
    if conn:
        cur = conn.cursor()
        # Encrypt the password right before saving it to the database
        enc = encrypt_val(pwd)
        if self.mode == "save":
            # Save a brand new entry
            cur.execute("INSERT INTO Passwords (UserID, AppName, Username, EncryptedPassword) OUTPUT INSERTED.PasswordID VALUES (?, ?, ?, ?)", self.uid, app, user, enc)
            pid = cur.fetchone()[0]
            cur.execute("INSERT INTO AuditLogs (PasswordID, UserID, ActionType) VALUES (?, ?, ?)", pid, self.uid, 'Creation')
        else:
            # Update an existing entry
            cur.execute("UPDATE Passwords SET AppName=?, Username=?, EncryptedPassword=? WHERE PasswordID=?", app, user, enc, self.data['pid'])
    conn.commit(); conn.close()
    if self.on_close: self.on_close()
    self.destroy()
```



Thank You

StrongPass — Secure. Customizable. Built to Grow.

Marcell Placencia · Ulyana Clarke · Christian Colon · Zeyon Charles

Instructor: Prof. Masoud Sadjadi | FIU KFSCIS | Capstone II — Spring 2026