National University of Computer & Emerging Sciences
Karachi Campus

Secure Password Manager (The VAULT)

Project Report
Secure Software Design
Section: BCY-7A
Instructor name: Ms. Abeer Gauhar

Group Members:
(22K-4746) - Jahanzeb Khairi
(22K-4754) - Syed Muhammad Murtaza Rizvi
(22K-4690) - Muhammad Yahya Khan

1. Introduction

1.1. Problem Statement

Current password storage methods in small-scale systems often lack fundamental security measures. Sensitive data is frequently vulnerable to:

- **Unauthorized Access:** Due to lack of encryption.
- **Rainbow Table Attacks:** Due to unsalted hashing.
- **Brute-Force Attacks:** Due to weak password policies.

1.2. Proposed Solution

The Secure Password Manager provides an offline, encrypted vault. It ensures that even if the local database file is stolen, the attacker cannot retrieve plaintext passwords or easily crack the hashes due to the implementation of a "Pepper" (a secret key stored separately from the database).

2. System Analysis and Design

2.1. Key Features

1. **User Authentication:** Secure login using hashed credentials.
2. **Password Vault:** Add, View, Update, and Delete credentials securely.
3. **Password Strength Checker:** Real-time validation of password complexity (Length, Case, Symbols).
4. **Cryptographic Security:** Usage of Salts and Peppers to fortify hashes.

2.2. Security Architecture

- **Hashing Algorithm:** SHA-256 (via System.Security.Cryptography).
- **Salting:** Random data added to passwords before hashing to ensure unique hashes for identical passwords.
- **Peppering:** A system-wide secret key (pepper.txt) added to the password-salt combination. This defends against database breaches where the attacker does not have access to the file system/application directory.
- **Storage:** Encrypted file storage for the vault data using AES-256-CBC (Advanced Encryption Standard with a 256-bit key in Cipher Block Chaining mode).

3. Implementation Details

3.1. Technology Stack

- **Language:** C++ / CLI (Common Language Infrastructure).
- **Framework:** .NET Framework (Windows Forms Application).
- **IDE:** Microsoft Visual Studio.
- **Libraries:** `System::Security::Cryptography`, `System::IO`, `System::Windows::Forms`.

3.2. Code Structure

The Pepper Mechanism:

The application retrieves a secret key from a local file named pepper.txt. If this file is missing, the security operations halt to prevent compromise.

Code Snippet (from MyForm.h):

```
String^ GetPepper() {
    String^ path = "pepper.txt";
    if (!File::Exists(path)) {
        // Error handling if pepper is missing
        return "";
    }
    return File::ReadAllText(path)->Trim();
}
```

Actual Pepper Content: `mySuperSecretPepper!@#2025`

Password Strength Logic:

The system enforces strong password policies by analyzing the user's input against specific criteria (Length > 8, Lowercase, Uppercase, Digits, Symbols).

Code Snippet (from MyForm.h):

```
int score = 0;
if (password->Length >= 8) score++;
if (hasLower) score++;
if (hasUpper) score++;
if (hasDigit) score++;
if (hasSymbol) score++;

if (score <= 2) strength = "WEAK";
else if (score == 3 || score == 4) strength = "AVERAGE";
else strength = "STRONG";
```

Main Entry Point:

The application utilizes a "MyForm" class as the GUI container, initializing visual styles and running the main event loop.

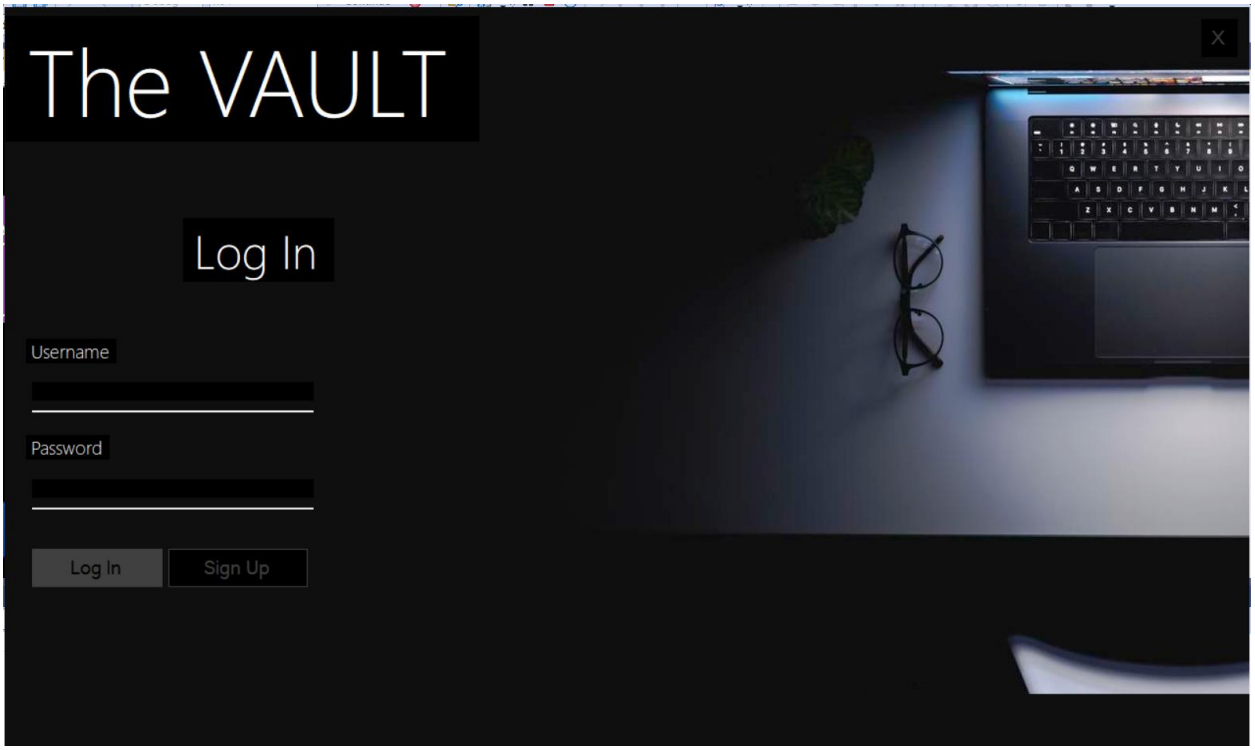
Code Snippet (from MyForm.cpp):

```
[STAThread]
void main() {
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);
    ssd::MyForm form;
    Application::Run(% form);
}
```

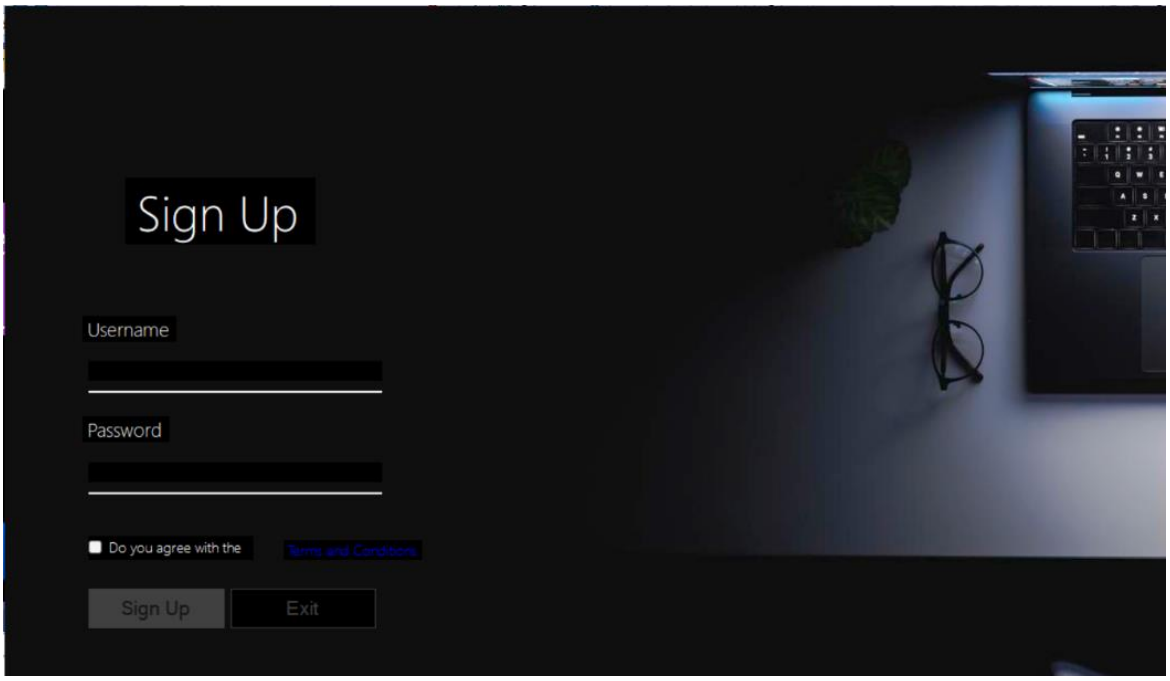
4. Results and GUI Walkthrough

The project successfully implements a Graphical User Interface (GUI) for ease of use. Below are the descriptions of the primary application screens (based on provided screenshots):

- **Login:** The entry point where users must authenticate.

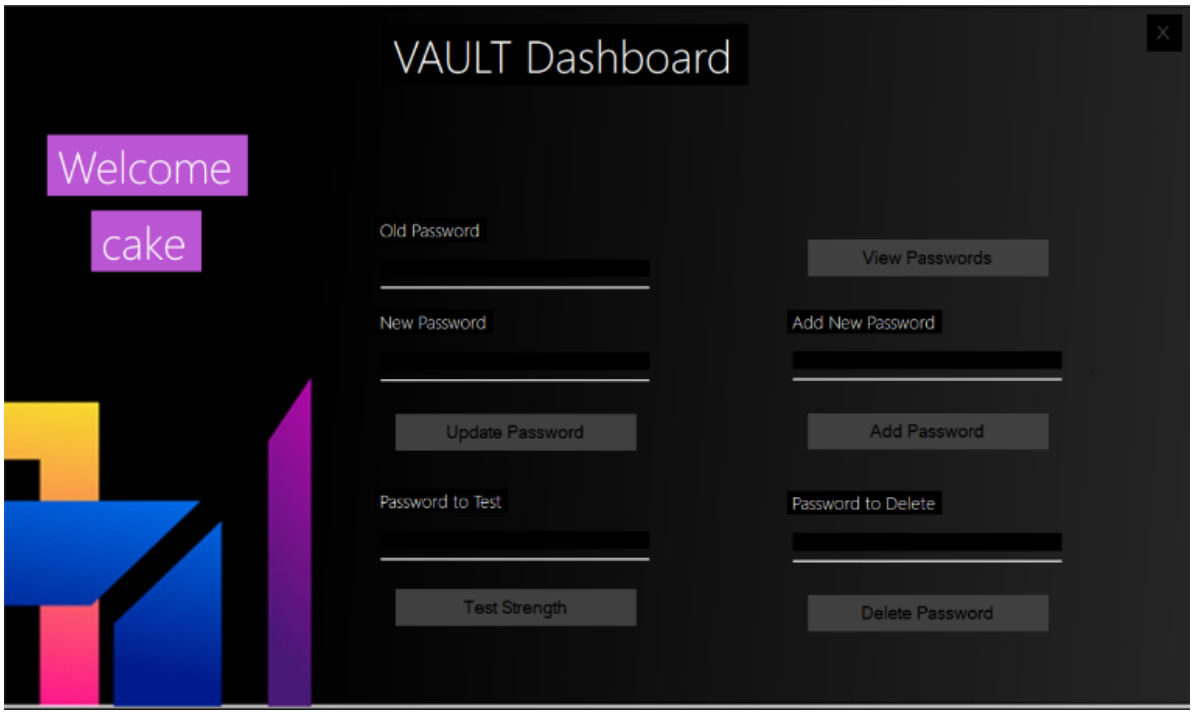


- **Sign Up:** The interface allowing users to sign up with new credentials.

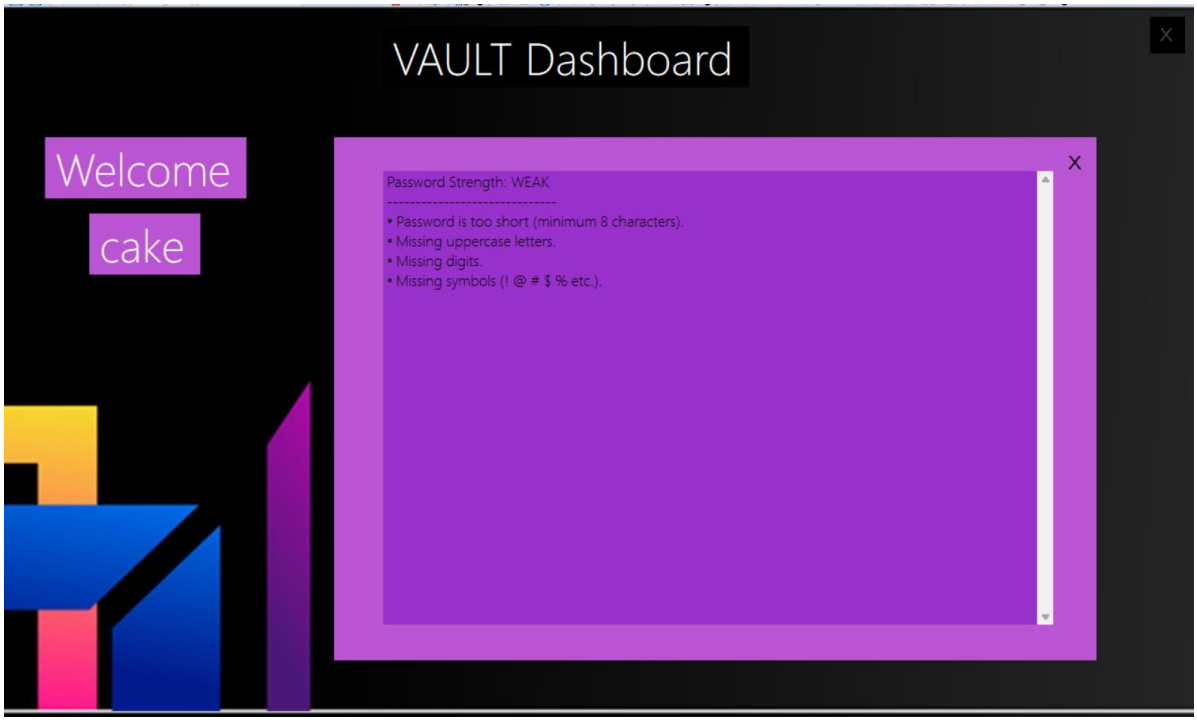


- **Terms and Conditions:** The interface allowing users to view the terms and conditions.

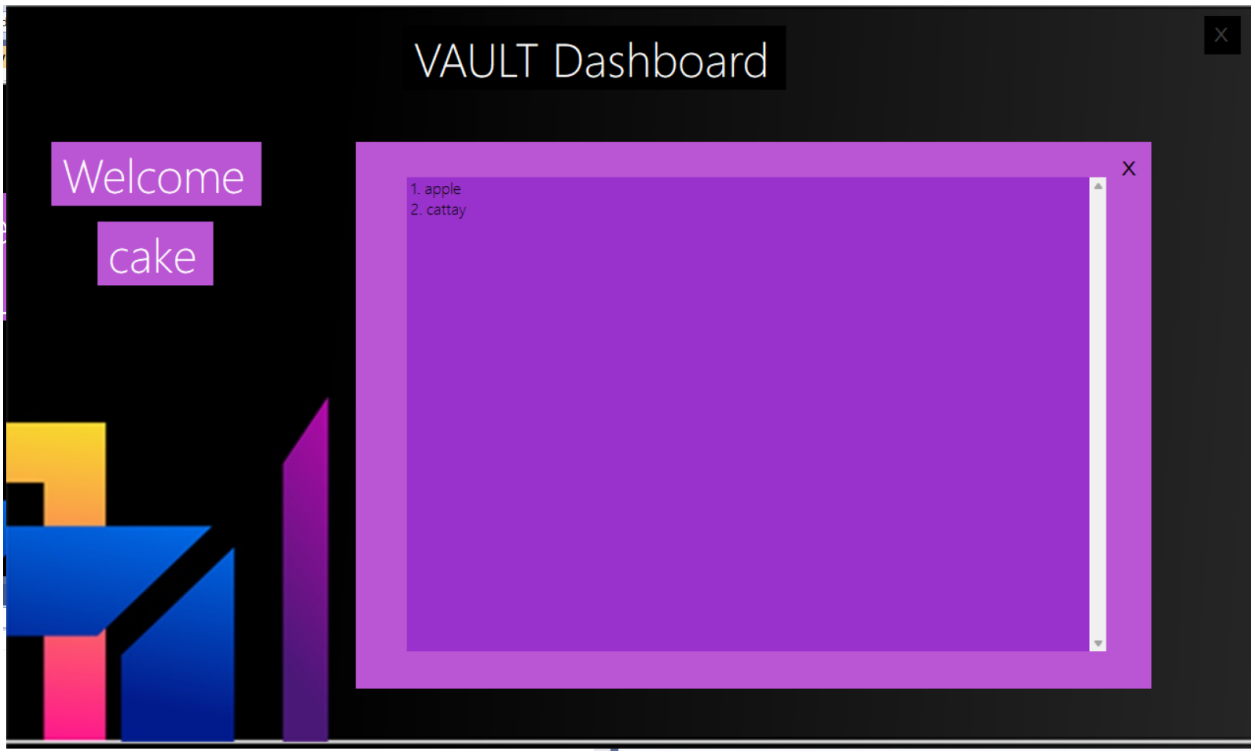
- **Dashboard:** The interface allowing users to add, update, delete, view and strength check passwords.



- **Security Analysis:** A view demonstrating the password strength checker in action or the viewing of encrypted entries.



- **Viewing Encrypted Passwords:** Visual appearance of decrypted passwords stored.



5. Conclusion

The Secure Password Manager fulfills the requirements of the Secure Software Design course. By moving from simple text storage to a cryptographically secure model using C++ and .NET libraries, we have mitigated common threats like brute-force and rainbow table attacks. The inclusion of the "Pepper" and AES-256-CBC encryption adds a layer of security that separates this project from standard academic exercises.