ChilPass Design Document

Version 1.2

Jonathan Cho and Hans Wilter

1. Introduction

1.1 Purpose

The purpose of this password manager application is to encrypt and store user passwords in select database files. The user can then access and decrypt the password by entering the master password associated with that database file. The master password will not be stored in clear text and on the user entering the password, will be hashed and compared to a stored hashed password.

1.2 Document Scope

The scope of this design document applies to the entirety of the password management application.

1.3 Design and Development Tools

LucidChart
Trello
Discord
Google Drive
Visual Studio 2019
Github

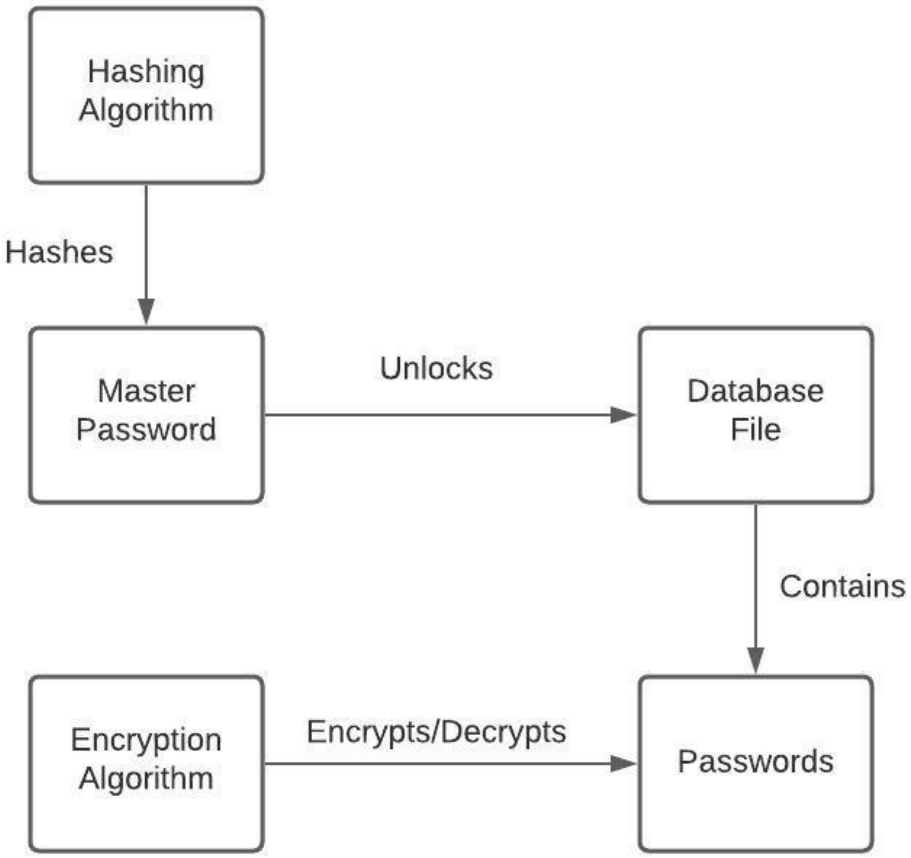
1.4 Logistics

The ChilPass password manager acts as an interface for authenticating and authorizing access to select database files associated with the ChilPass application. In practice, when creating a password file, ChilPass creates two files. One file is a database file that stores encrypted entries that include information like the title and password for each entry. The second file is a text file, which stores the name of the associated database file and the hashed master password associated with that file.

2. System Overview

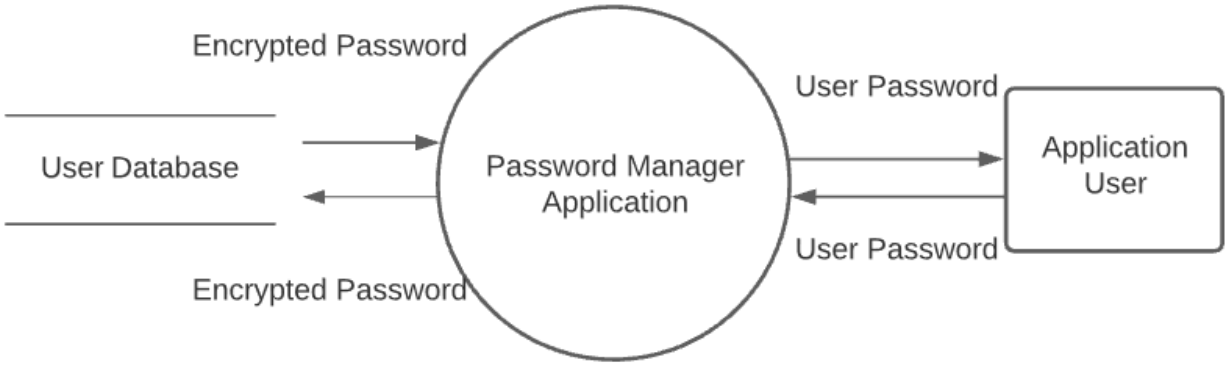
2.1 Project Flow Chart

This diagram shows the flow of processes that occur when accessing or adding new passwords to a given database file.



2.2 Data-flow Diagram

Content Diagram (Level 0) -



The level 0 data flow diagram showcases the flow of data between the user, the password manager application, and the database files.

Level 1 Diagram -

Level 2 Diagram -

3. System Requirements

3.1 Functional Requirements

List of Functional Requirements

- As a user, I want to be able to store passwords with easy access
- As a user, I want to be able to remove password entries
- As a user, I want to be able to change my passwords accordingly
- As a user, I want to be able to access multiple database files

Potential Functional Requirements

- As a user, I want to be able to auto-generate random strong passwords
- As a user, I want to be able to have an autofill option for my applications

3.2 Non-functional Requirements

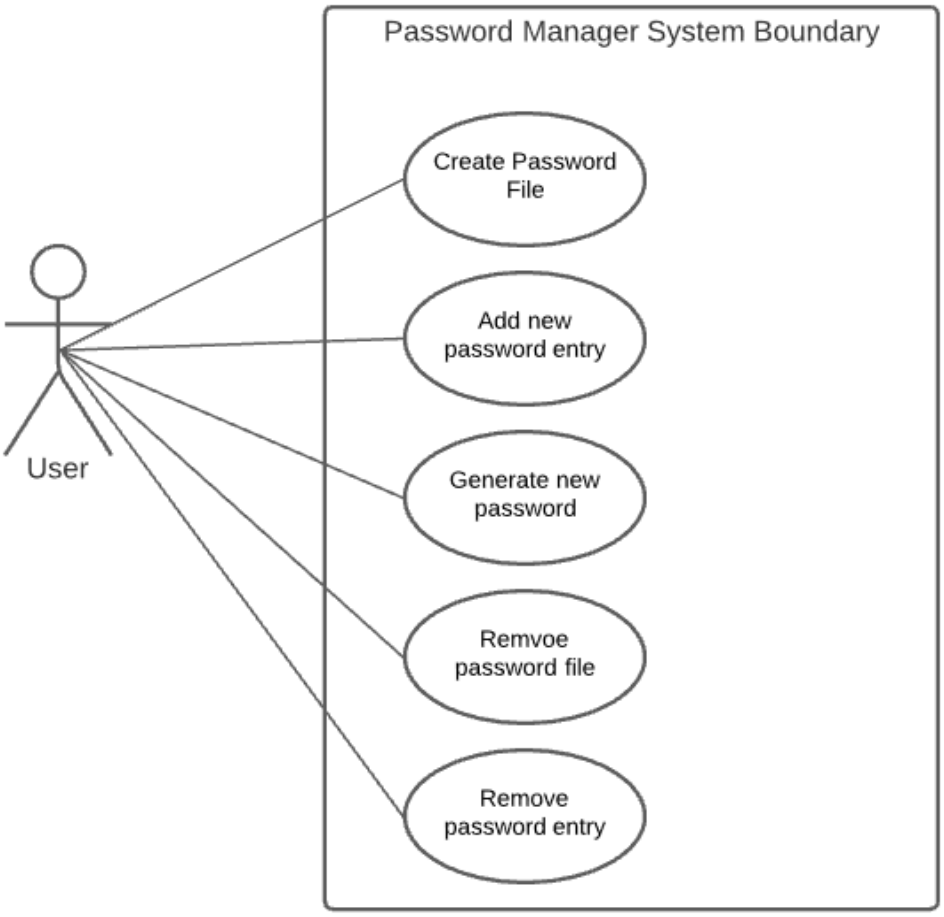
List of Non-functional Requirements

- As a user, I want the application to be available 24/7

- As a user, I want the application to be safe and secure

3.3 Use Case Diagram

The Use Case Diagram showcases how the user interacts with the password manager application. It includes all of the use cases with which the user can and will interact with the system.



3.4 Use Case 1

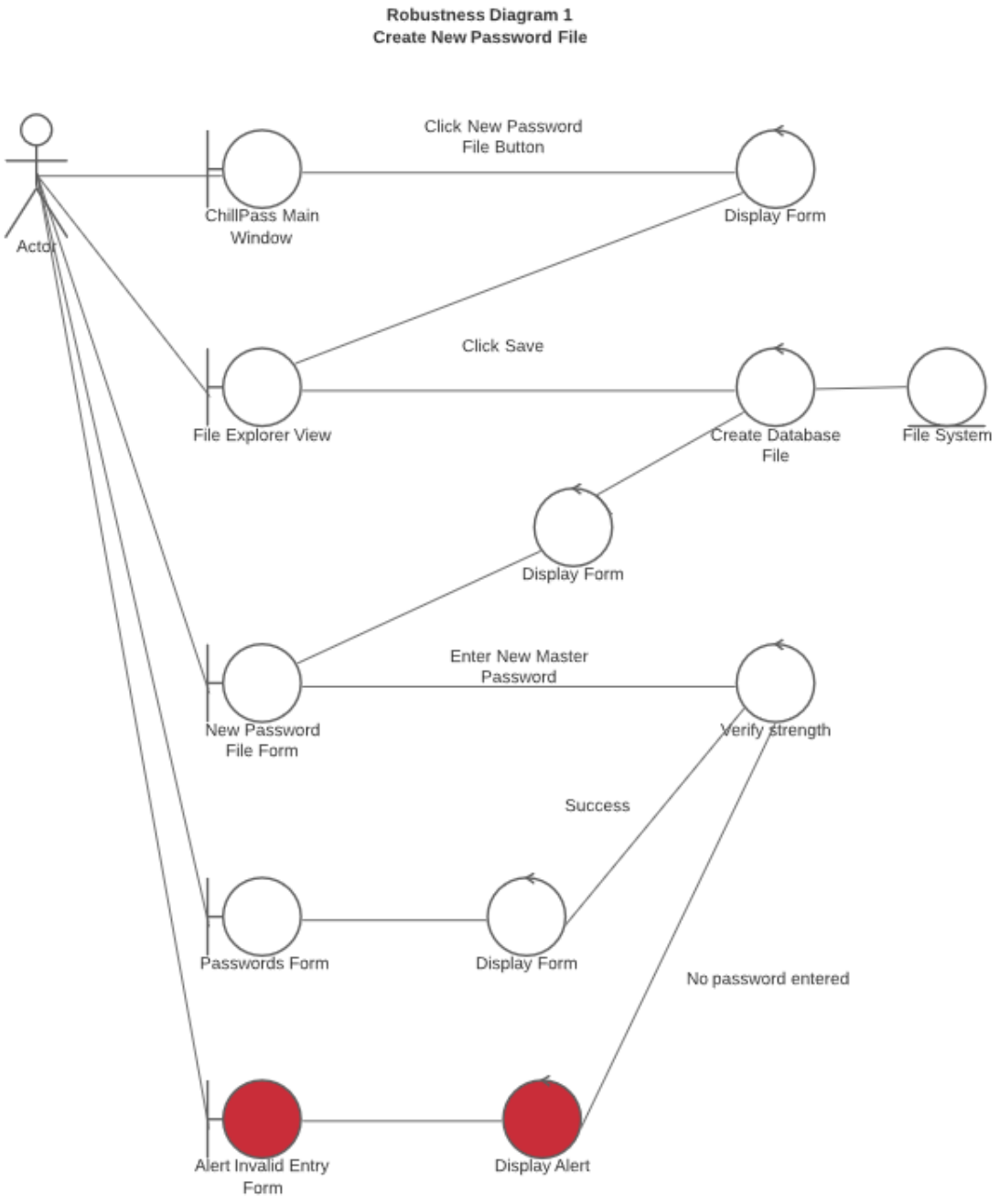
Use Case Scenario: Create a password file
Actor: User
Stakeholders and Needs:
User: Enters data
Software Engineer: Maintain application updates
Company: Have valid encryption and hashing
Preconditions: User has access to a computer with ChilPass installed
Postconditions: New password file, formatted for the database
Trigger: User initiates the creation of a password file
Basic Flow:

1. User interacts with ChilPass to initiate creation of password file
2. The application creates the file in the designated location
3. The application assigns the hashed master password that the user entered to the file

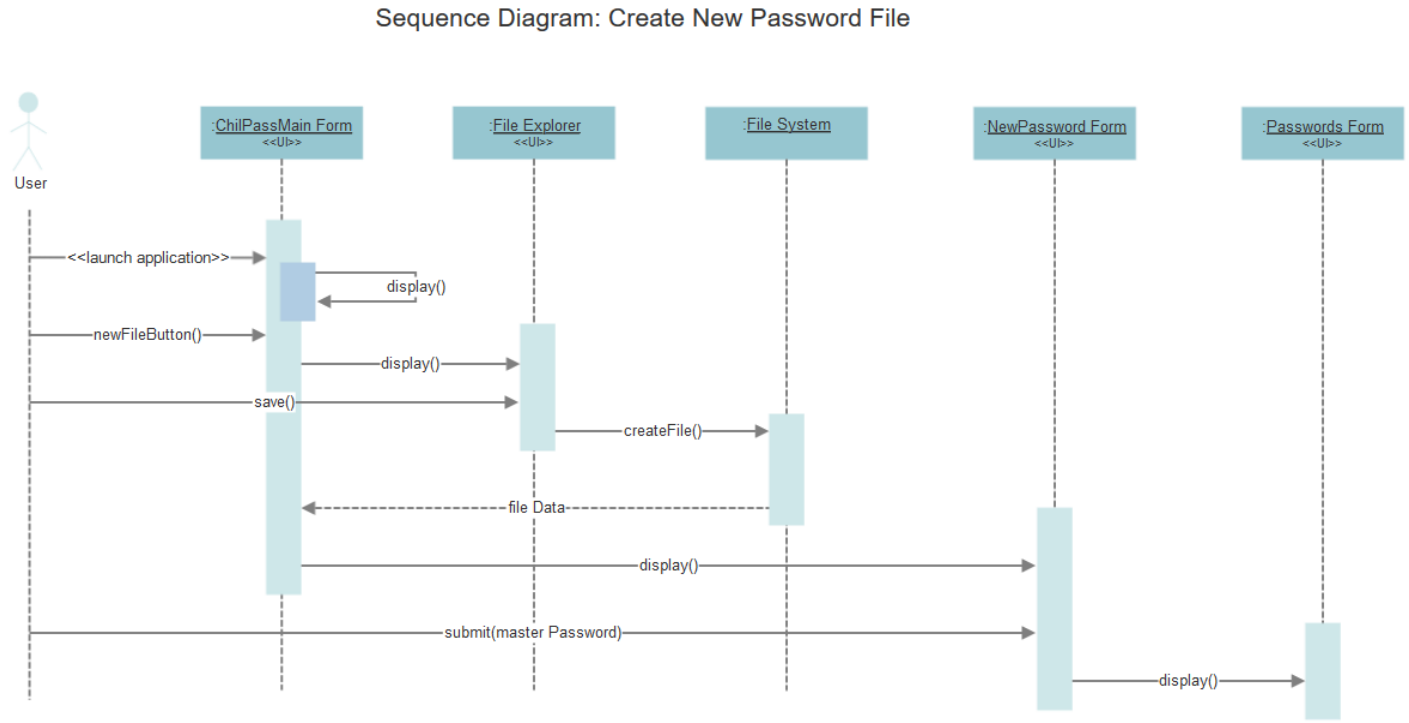
Extensions:

1. User does not enter a master password
    a. Show warning that no password was entered

3.4.1 Robustness Diagram 1



3.4.2 Sequence Diagram 1



3.5 Use Case 2

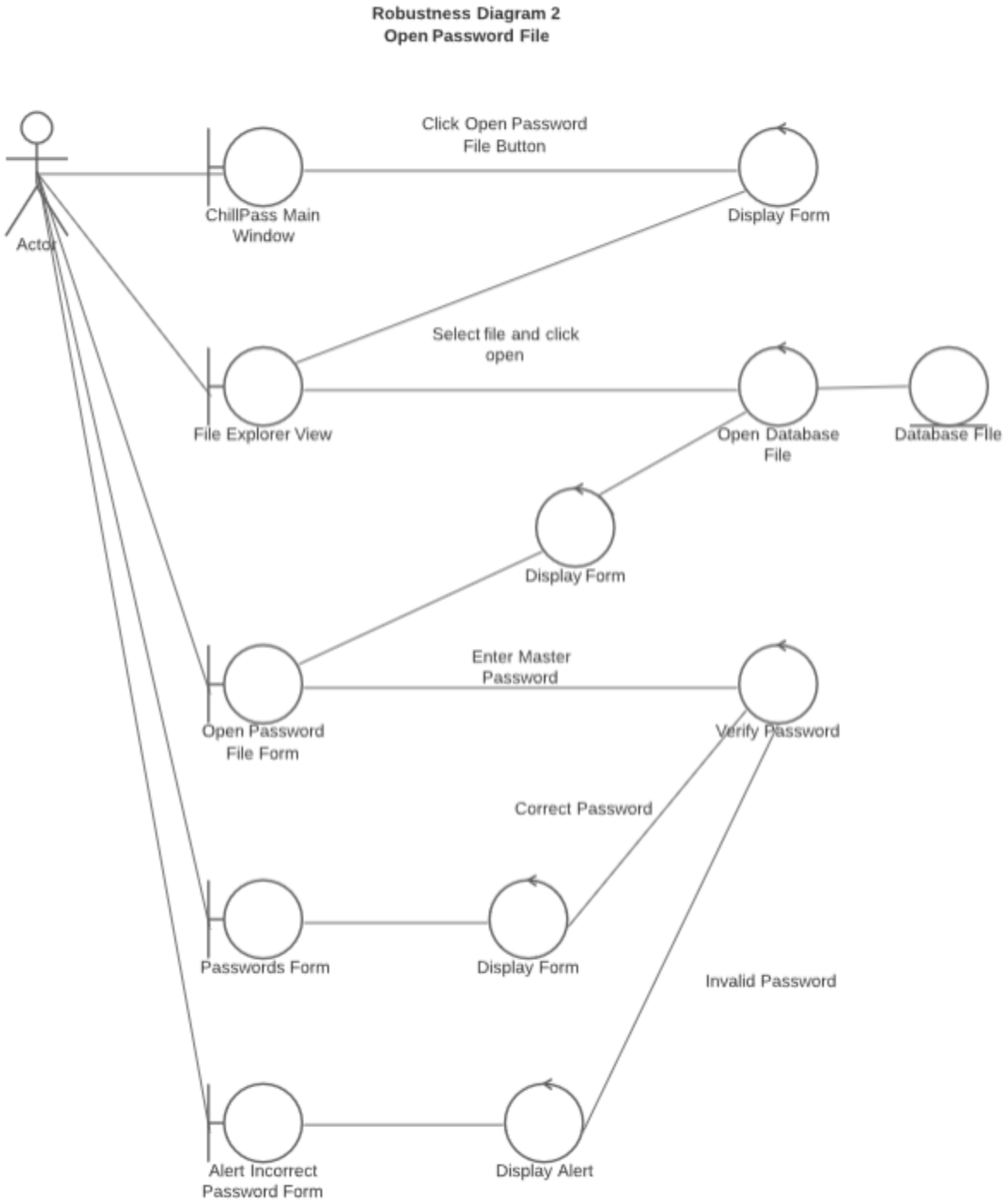
Use Case Scenario: Open a password file
Actor: User
Stakeholders and Needs:
User: Enters data
Software Engineer: Maintain application updates
Company: Have valid encryption and hashing
Preconditions: User has access to a computer with ChilPass installed
Postconditions: The password file is unencrypted and displayed for the user once authenticated.
Trigger: User attempts to access a Chilpass password file
Basic Flow:

1. User interacts with ChilPass to open an existing ChilPass password file
2. User enters the master password associated with the file
3. The application authenticates the password that the user entered and grants authorization to the password depending on the validity of the password

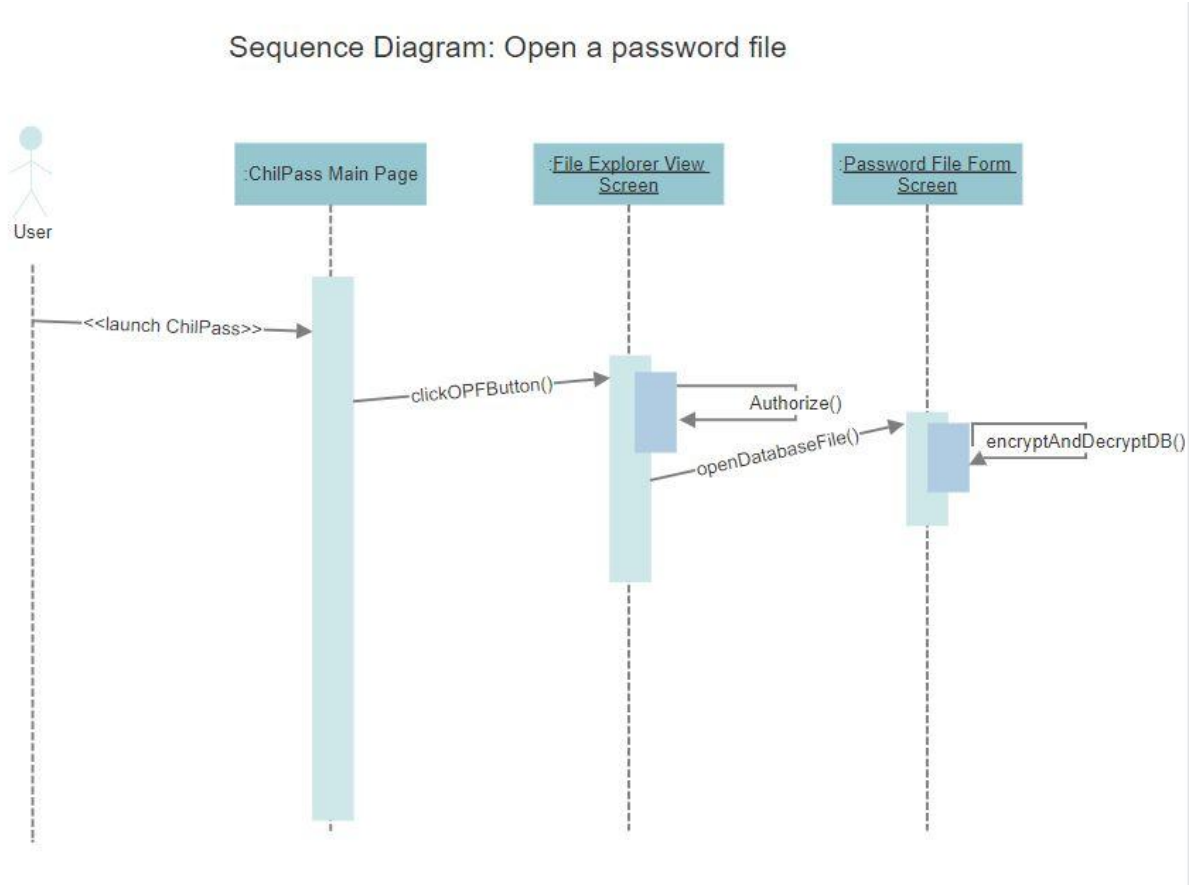
Extensions:

1. User does not enter a master password
    a. Show warning that no password was entered
2. User enters incorrect master password
    a. Show a warning that the password entered was incorrect

3.5.1 Robustness Diagram 2



3.5.2 Sequence Diagram 2



3.6 Use Case 3

Use Case Scenario: Add a new password entry
Actor: User
Stakeholders and Needs:
User: Enters data
Software Engineer: Maintain application updates
Company: Have valid encryption and hashing
Preconditions:

- User has access to a computer with ChilPass installed
- The user has selected an existing password file compatible with ChilPass

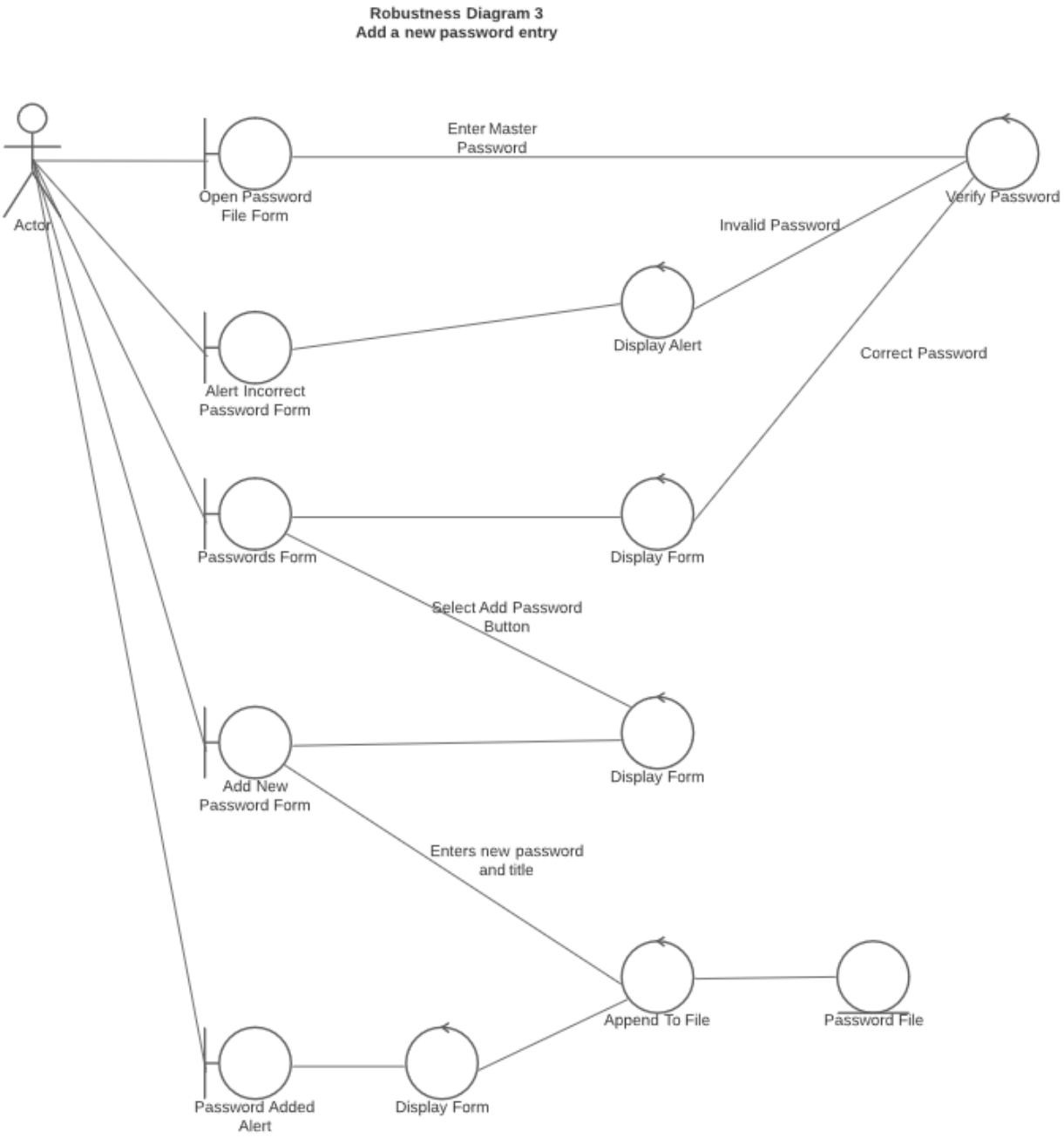
Postconditions: New password entry added to the database password file
Trigger: User initiates the creation of an entry in a selected password file
Basic Flow:

1. User interacts with ChilPass UI to initiate the addition of an entry in a given password file
2. The user enters the entry name and a password for the entry to be added
3. The application encrypts the password entered and adds the entry to the selected database password file

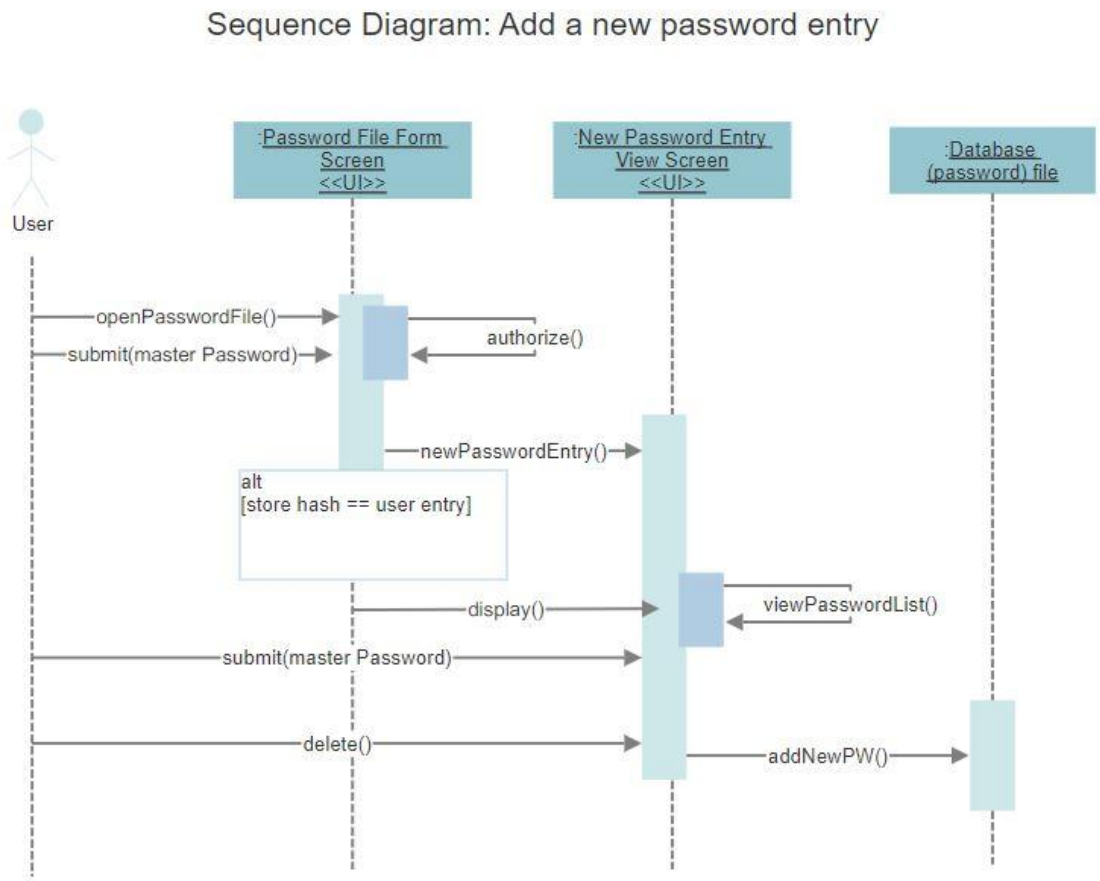
Extensions:

1. User does not enter a name for the entry
    a. Show warning that no name has been entered in the title field
2. User does not enter a password for the entry
    a. Show a warning that no password has been entered into the password field

3.6.1 Robustness Diagram 3



3.6.2 Sequence Diagram 3



3.7 Use Case 4

Use Case Scenario: Remove password entry
Actor: User
Stakeholders and Needs:
User: Enters data
Software Engineer: Maintain application updates
Company: Have valid encryption and hashing
Preconditions:

- User has access to a computer with ChilPass installed
- The user has selected the an existing password file compatible with ChilPass

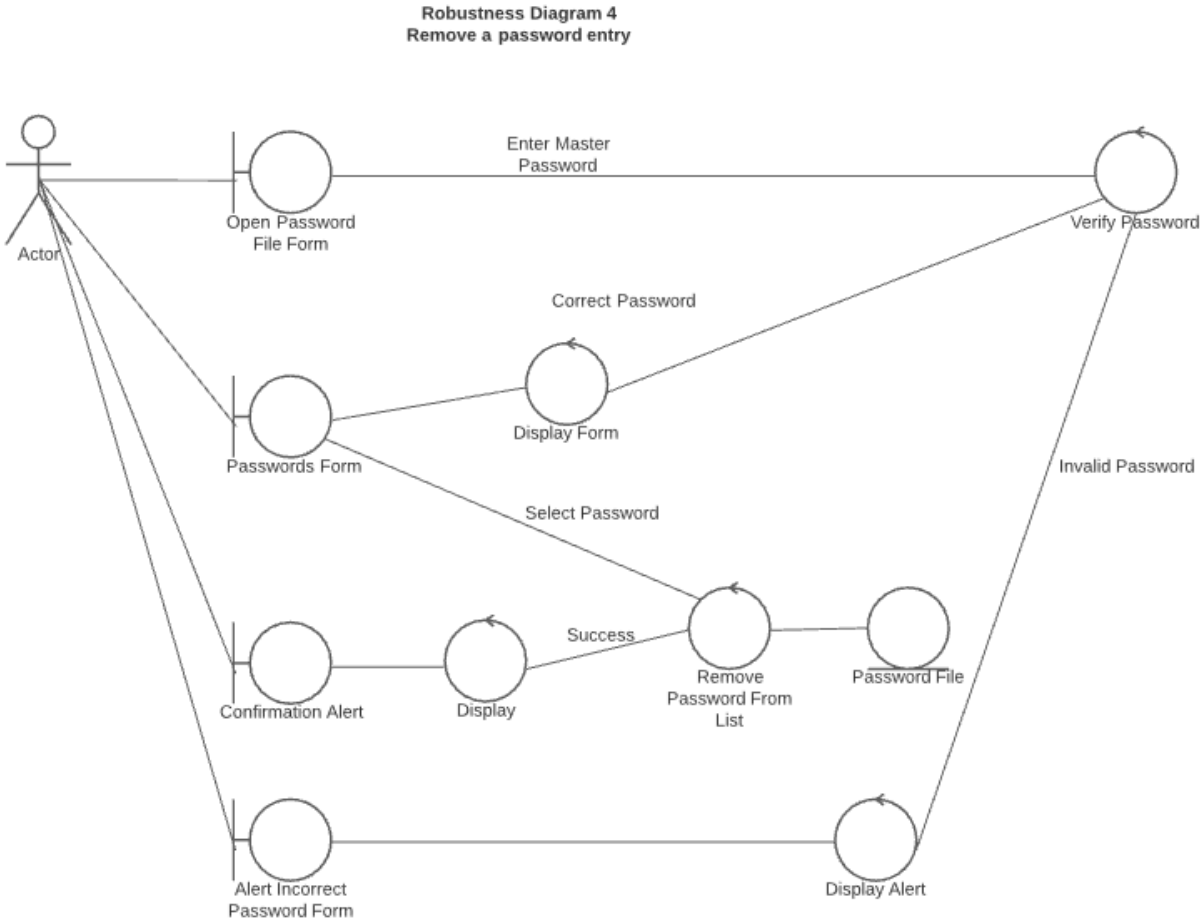
Postconditions: Existing password entry is removed from the database password file
Trigger: User initiates the deletion of an entry in a selected password file
Basic Flow:

1. User interacts with ChilPass UI to initiate the subtraction of an entry in a given password file
2. The user enters the entry name and a password for the entry to be removed
3. The application decrypts the password entered and removes the entry to the selected database password file

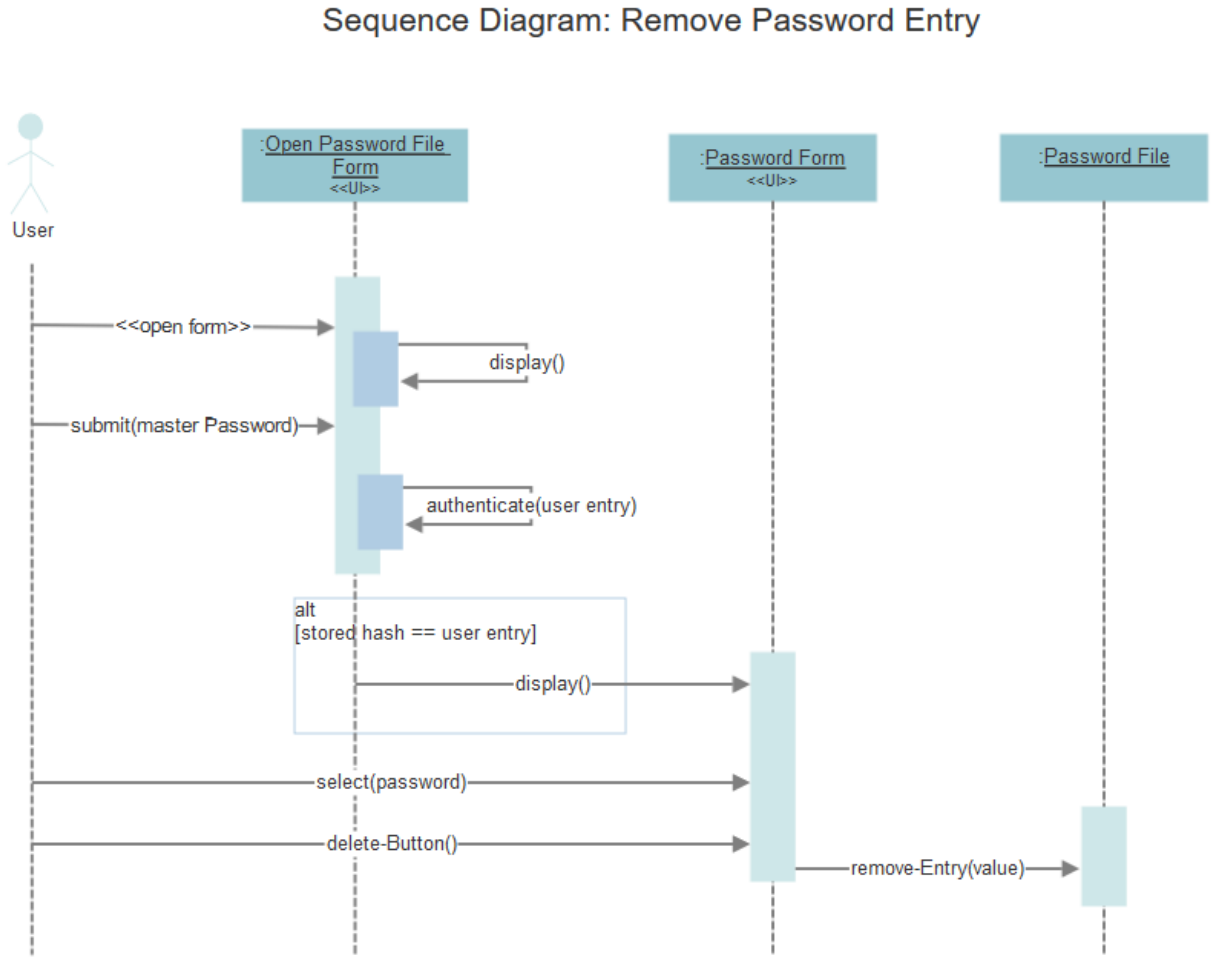
Extensions:

1. User does not enter a valid name for the entry
    a. Show warning that no valid entries are entered that can be deleted

3.7.1 Robustness Diagram 4



3.7.2 Sequence Diagram 4



3.8 Use Case 5

Use Case Scenario: Generate a new random password
Actor: User
Stakeholders and Needs:
User: Enters data
Software Engineer: Maintain application updates
Company: Have valid encryption and hashing
Preconditions:

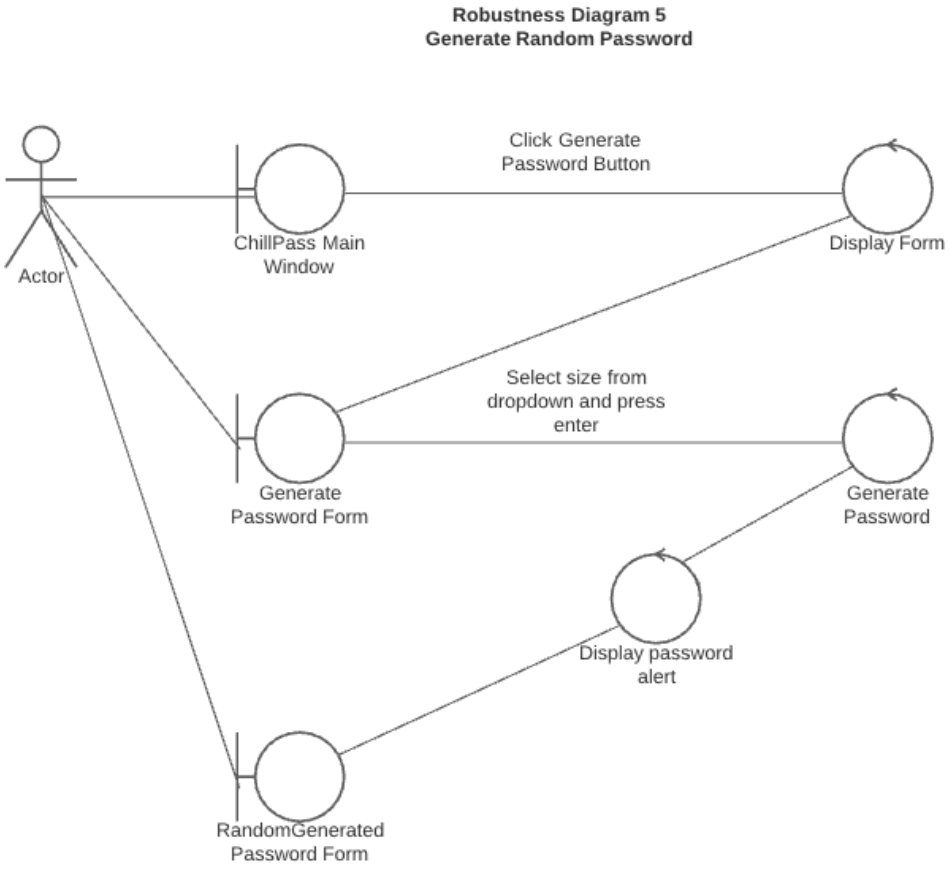
- User has access to a computer with ChilPass installed
- The user has selected the Generate New Password option

Postconditions: New password generated must be added to the database password file
Trigger: User initiates the creation of a random password entry in a selected password file
Basic Flow:

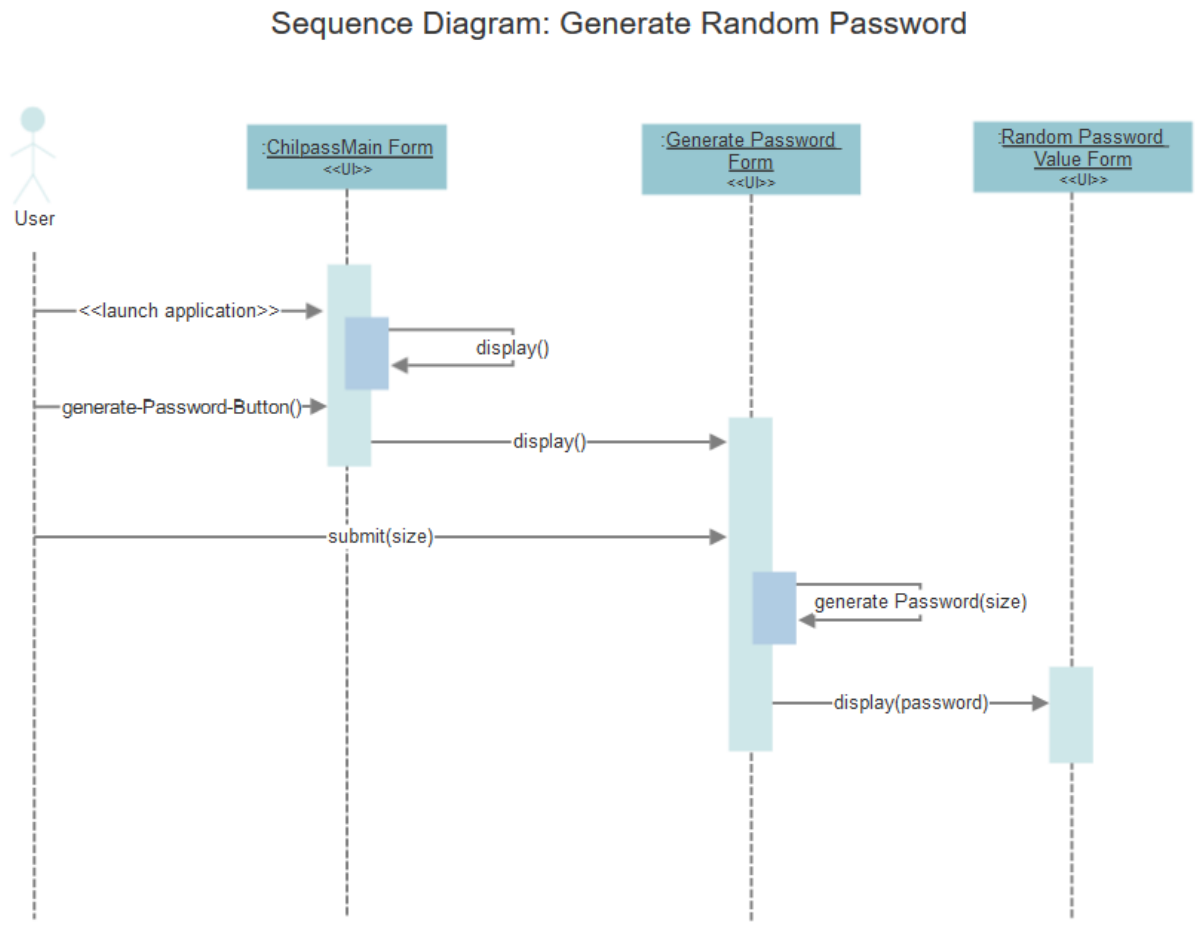
1. User interacts with ChilPass UI to initiate the generation of a random password
2. The user selects the size of the password that they want to generate
3. The application generates a random password of the specified size

Extensions:

3.8.1 Robustness Diagram 5



3.8.2 Sequence Diagram 5



3.9 Use Case 6

Use Case Scenario: Remove password file
Actor: User
Stakeholders and Needs:
User: Enters data
Software Engineer: Maintain application updates
Company: Have valid encryption and hashing
Preconditions:

- User has access to a computer with ChilPass installed

- The user has selected the Remove Password File option with an existing password file compatible with ChilPass

Postconditions: Existing password file removed
Trigger: User initiates the deletion of a password file
Basic Flow:

1. User interacts with ChilPass UI to initiate deletion of a password file
2. The user selects the text file to be deleted from the file tree view
3. The application deletes the database file and the text file

Extensions:

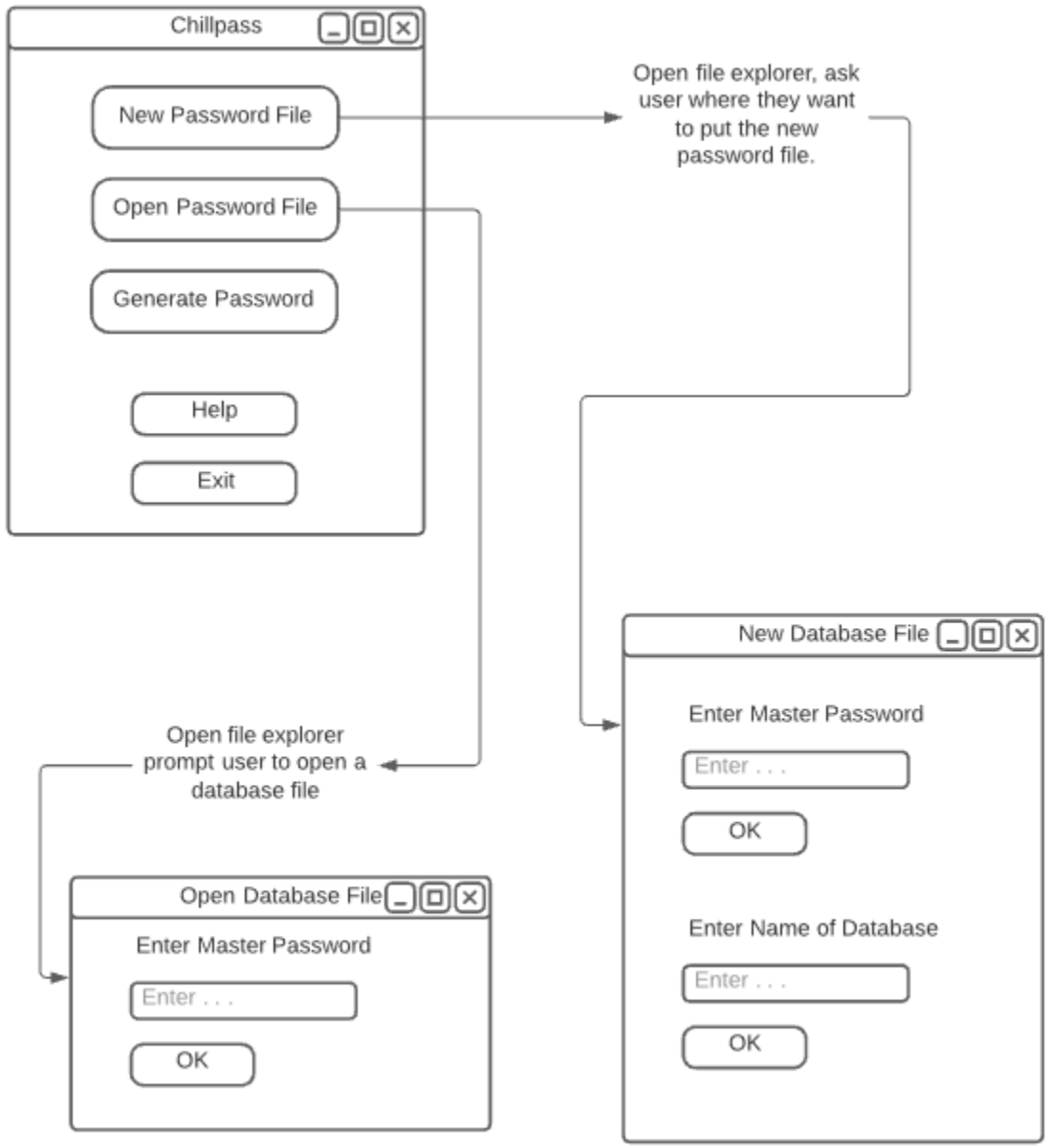
3.9.1 Robustness Diagram 6

3.9.2 Sequence Diagram 6

4. User Interface (Mock)

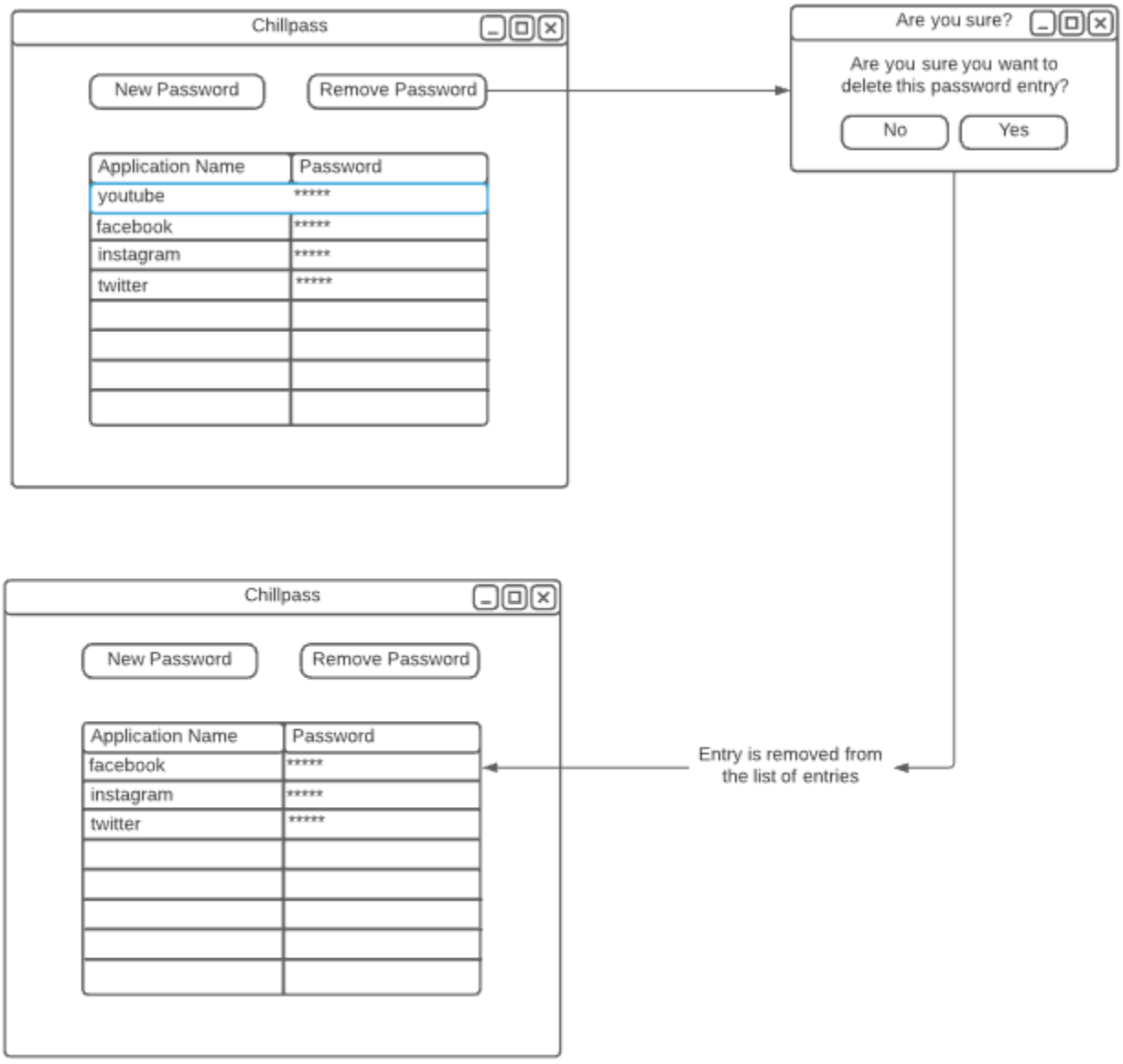
4.1 Home Window and Open / Create File

The Home Window and Open / Create File UI mockup showcases the basic UI flow that a user will experience when creating or opening a password file from the home window.



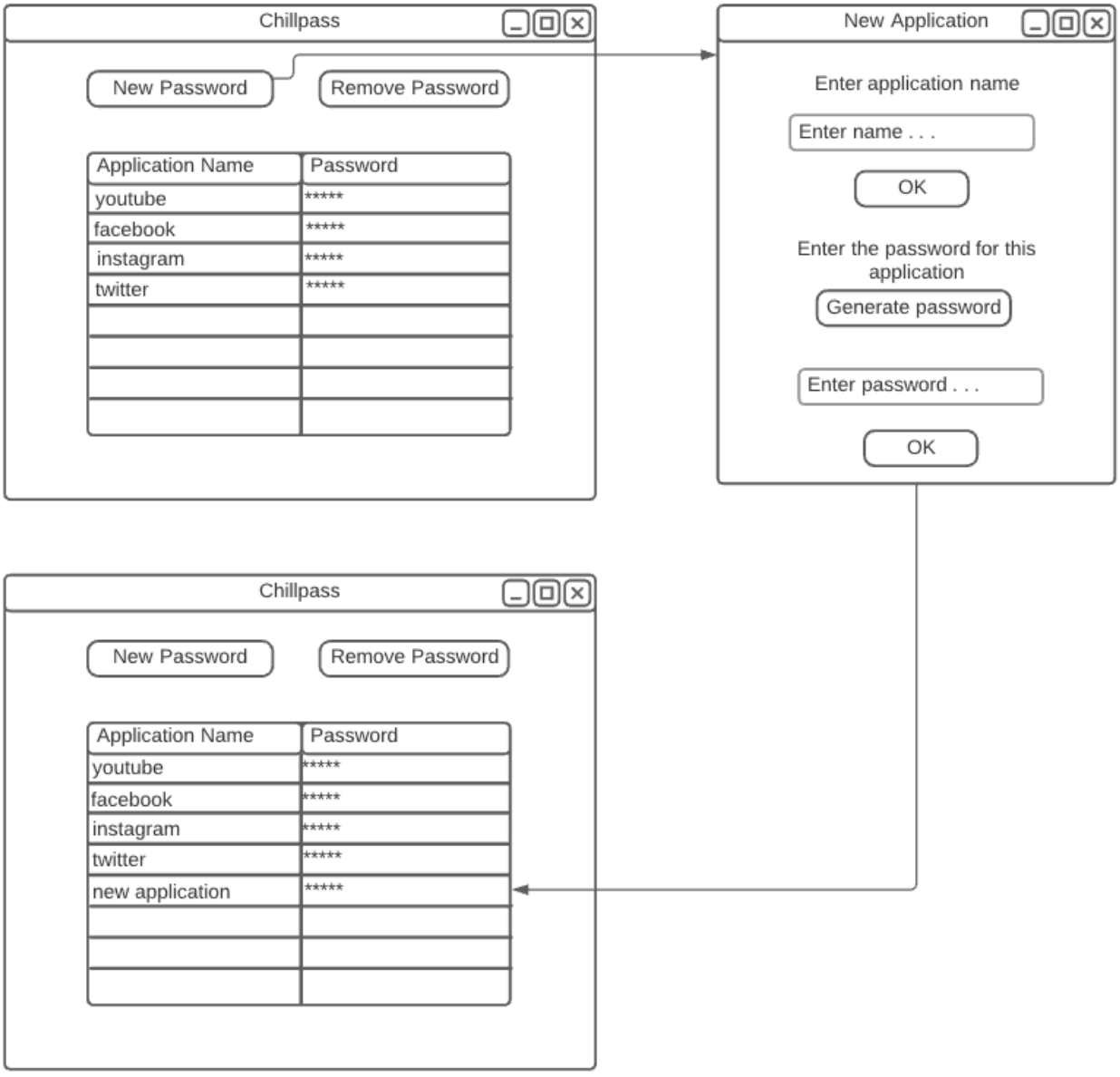
4.2 Removing a password entry

This UI wireframe showcases the basic UI functionality that a user will experience when interacting with the ChilPass system to remove an existing entry for a select password file.


Chillpass
New Password
Remove Password
Application Name
Password
youtube
facebook
instagram
twitter
*****
Are you sure?
Are you sure you want to delete this password entry?
No
Yes
Entry is removed from the list of entries

4.3 Adding a new password entry

This UI wireframe showcases the basic UI functionality that a user will experience when interacting with the ChilPass system to add an new entry for a selected password file.

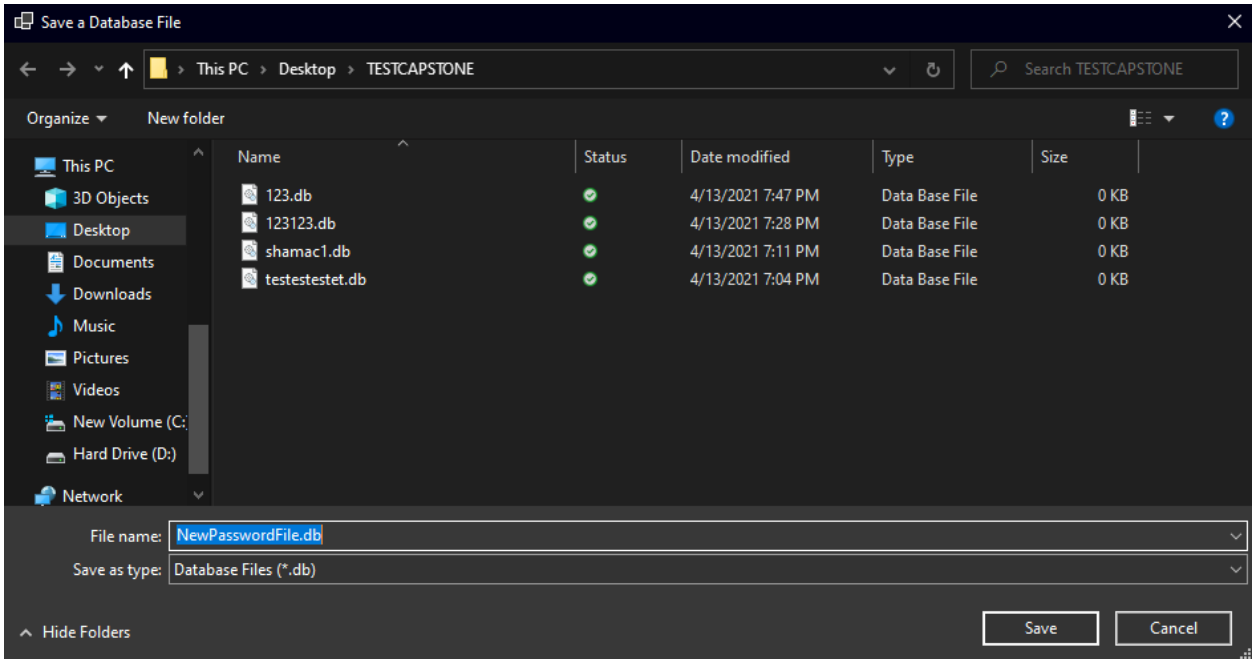


5. User Interface

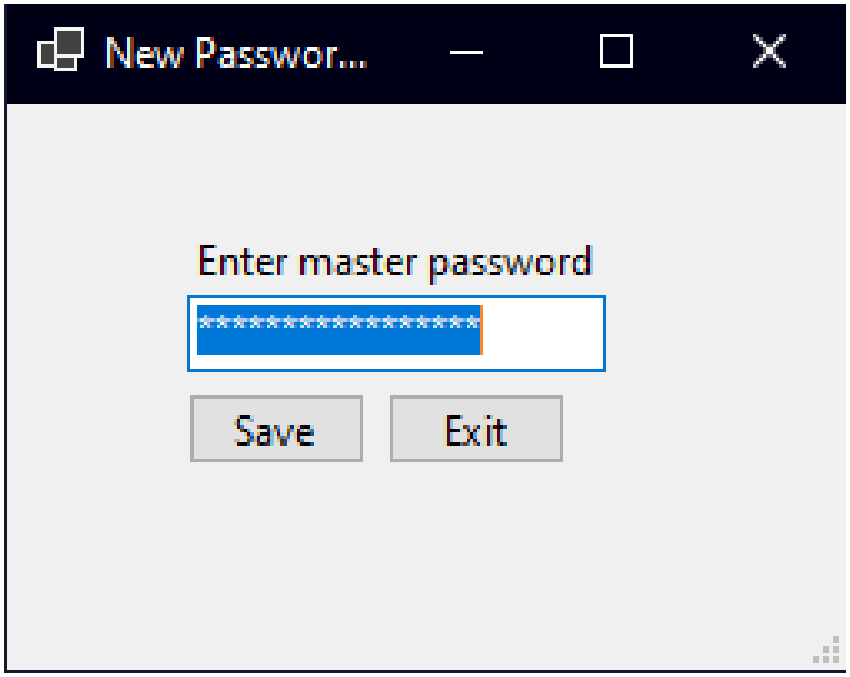
5.1 Home Interface / Create File

5.1.1

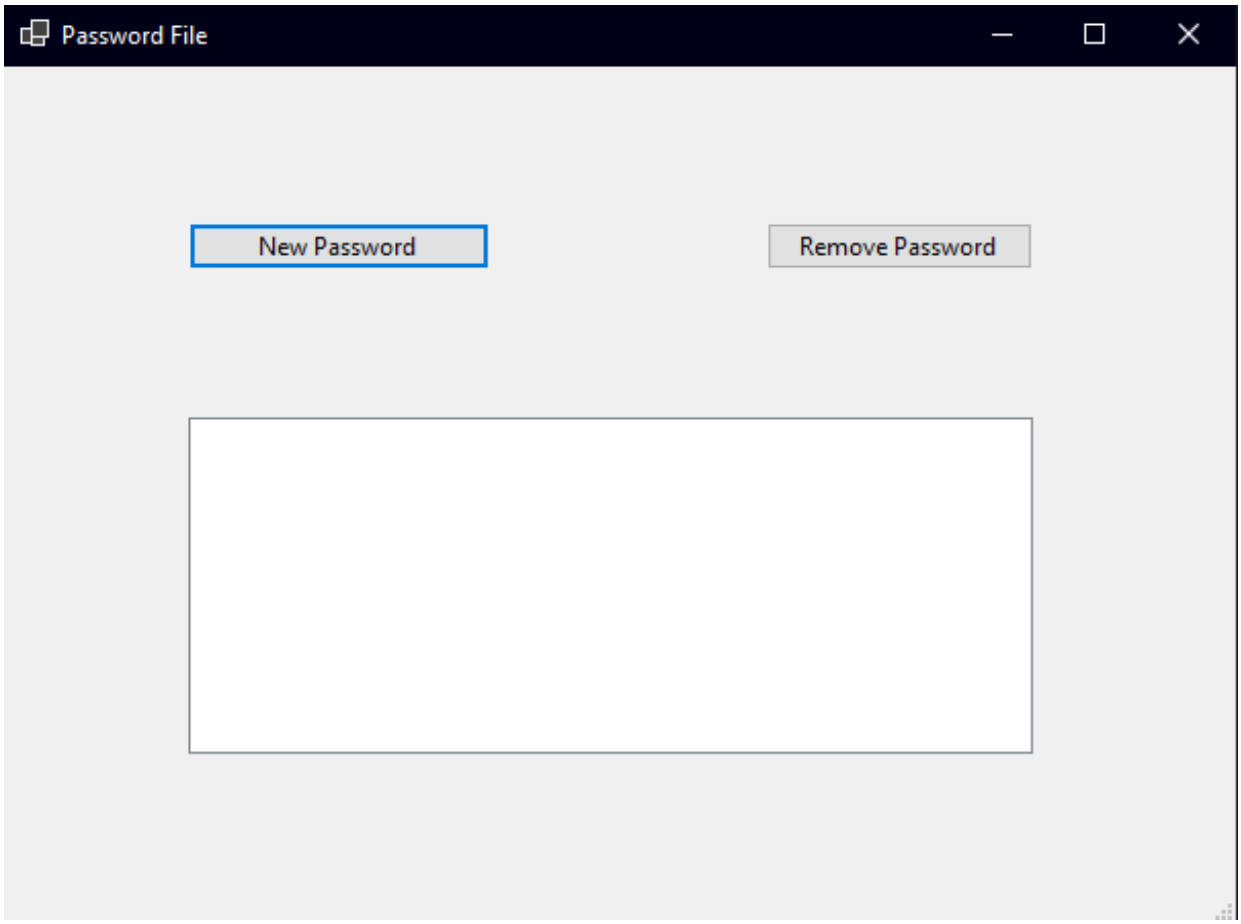
5.1.2



5.1.3

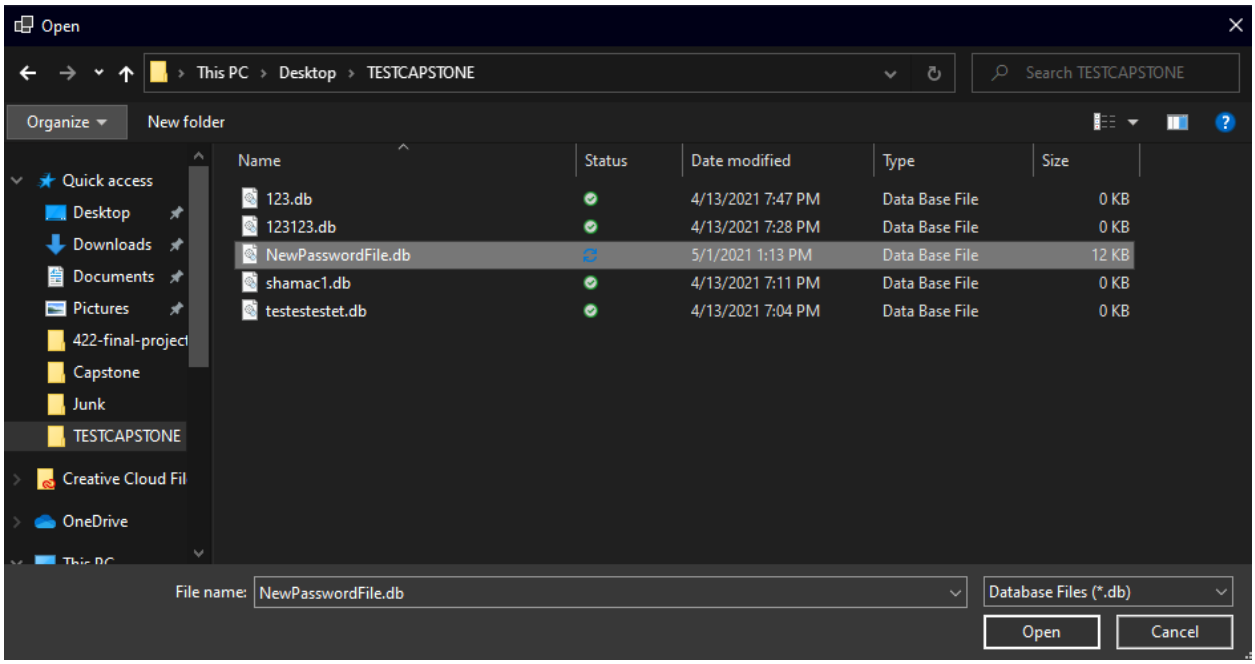

New Passwor...
Enter master password
Save
Exit

5.1.4

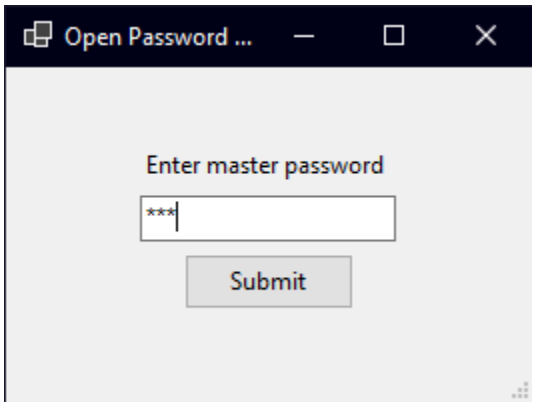

Password File
New Password
Remove Password

5.2 Open Password File

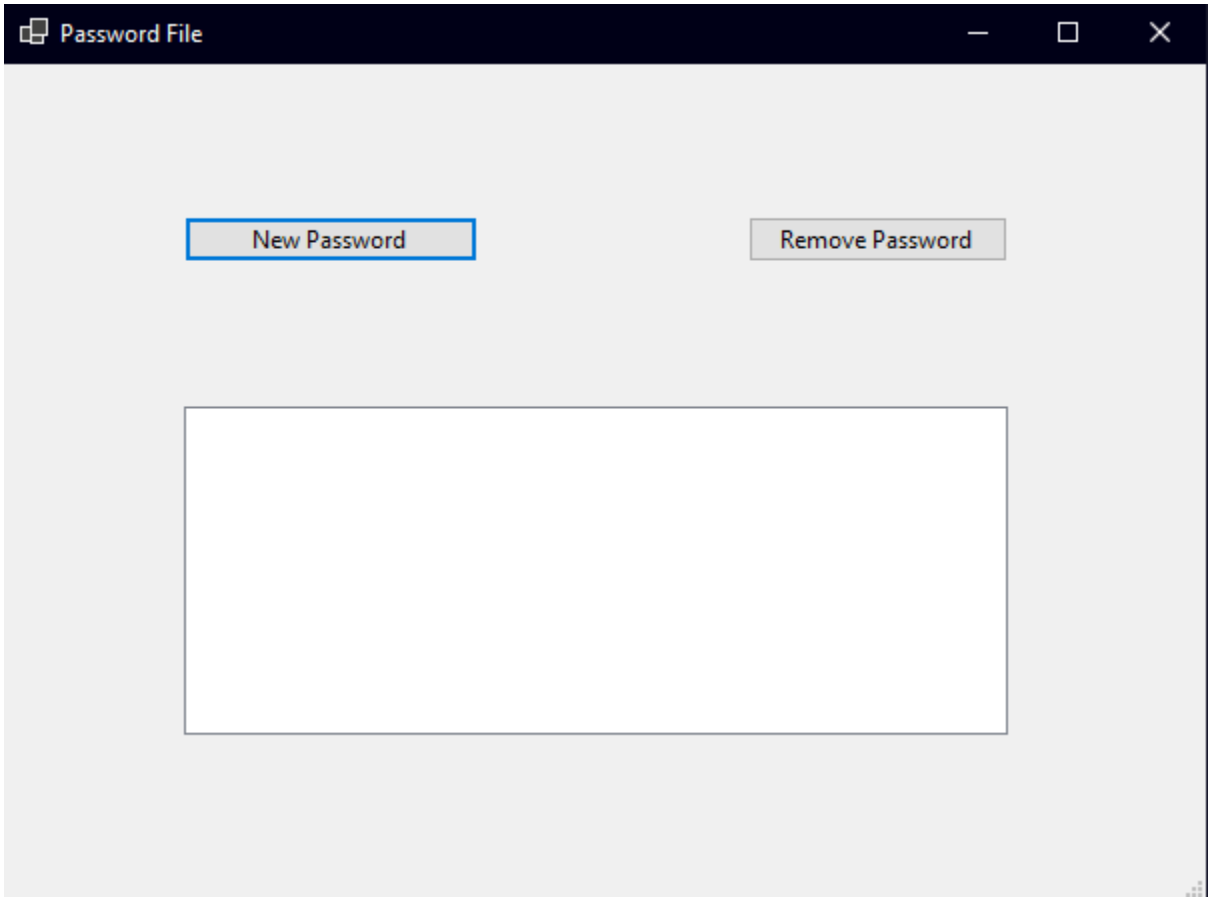
5.2.1



5.2.2

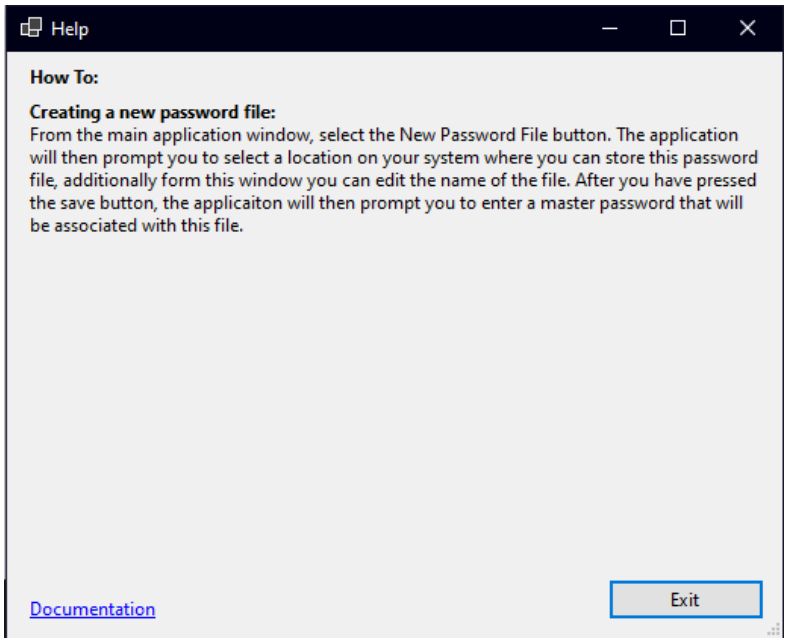


5.2.3



5.3 Help

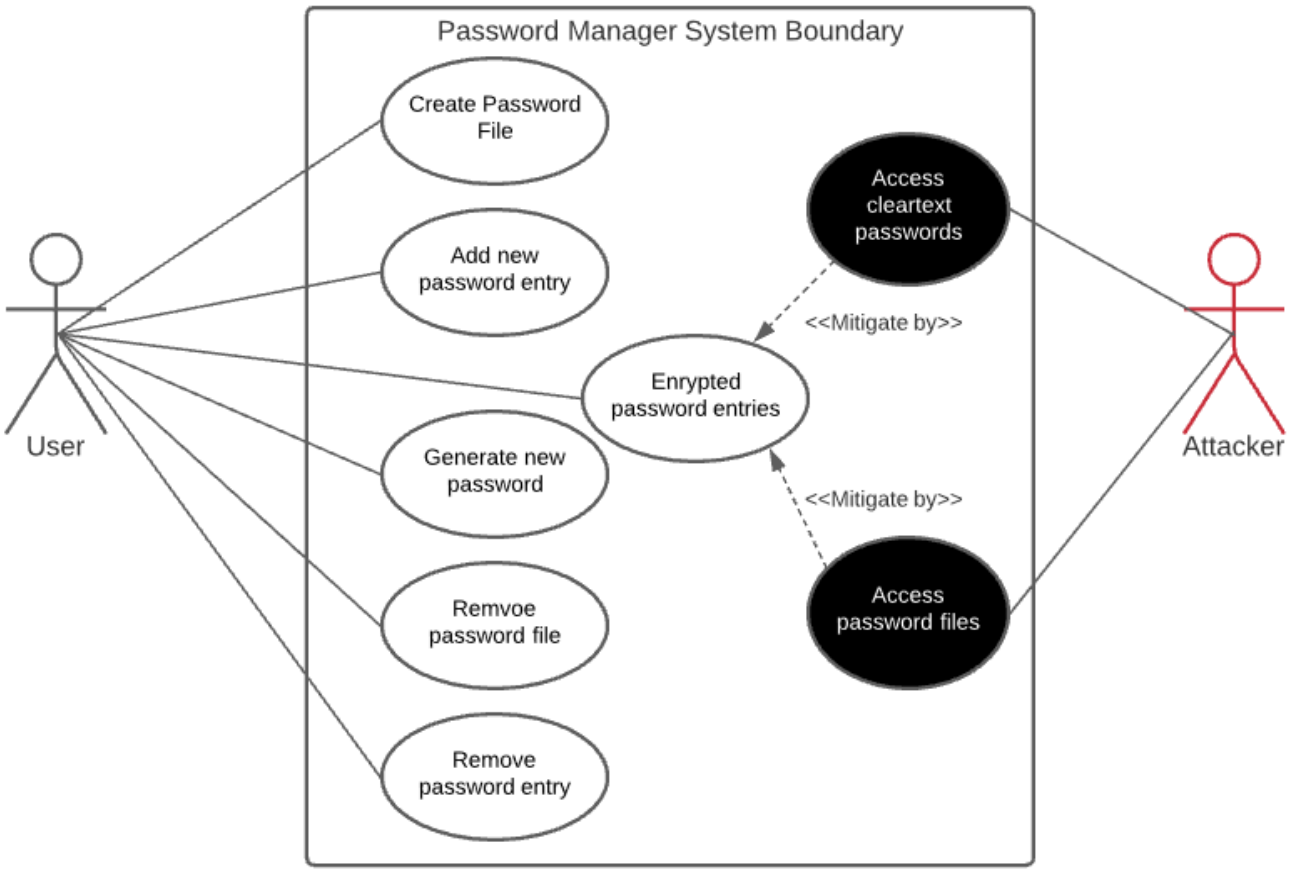
5.3.1



6. System Security

6.1 Misuse Case Diagram

The Misuse Case Diagram displays the use cases with which a user may interact with the ChilPass system in addition to ways with which an attacker might misuse or gain unintended access to the system and how it can be mitigated by the system.



6.2 Authentication

ChilPass authenticates users that are attempting to access a database file through the master password. This is done through comparing the resulting PBKDF-2 output hash of the user entered password to the PBKDF-2 hash stored in the database file. If the comparison of the hash is found to match, the system moves proceeds to grant authorization to the user.

6.3 Authorization

Once the authentication process is completed, the system moves onto the authorization process. In this stage, if the master password entered was correct, the system grants authorization to the user and decrypts the entries as they are displayed to the user. Once the user exits the display window, they will be required to go through the authentication process again if they wish to view the passwords again.

6.3.1 PBKDF2 (for authorization)

```
4 references
public static string PBKDF2(string password, byte[] theSalt, int iterations)
{

    // TODO: Leave a notifaction that it is computing hash or running so the user doesn't think its dying

    string hashed = Convert.ToBase64String(KeyDerivation.Pbkdf2(
        password: password,
        salt: theSalt,
        prf: KeyDerivationPrf.HMACSHA256,
        iterationCount: iterations, // increase later
        numBytesRequested: 128 / 8));

    return hashed;
}
```

```
if (authorized)
{
    System.Diagnostics.Debug.WriteLine("Correct Password, granting access...");
    // decryption time
    var openPasswordFile = Application.OpenForms["FileForm"];
    if (openPasswordFile == null)
    {
        openPasswordFile = new FileForm(encKey, file);
    }

    openPasswordFile.ShowDialog();
}
else
{
    // password incorrect
    System.Diagnostics.Debug.WriteLine("Wrong password!");
}
```

6.4 Encryption

When it comes to encrypting individual password entries, Chilpass utilizes AES using a key derived from the user entered master password. Each password file will generate a different encryption key regardless of the password entered, which is accomplished through the use of PBKDF-2. AES is the recognized standard for encryption that is employed by the U.S. government and agencies such as the NSA.

6.4.1 Encryption Algorithm

```
/*
 * Referenced: https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.cryptostream?view=net-5.0
 */
2 references
public static string Encrypt(string key, string value)
{
    byte[] iv = new byte[16];
    byte[] encryptedText;

    using (Aes aes = Aes.Create())
    {
        aes.KeySize = 128;
        aes.Key = Encoding.UTF8.GetBytes(key);
        aes.IV = iv;


        ICryptoTransform encryptor = aes.CreateEncryptor(aes.Key, aes.IV);

        using (MemoryStream mEncrypt = new MemoryStream())
        {
            using (CryptoStream cEncrypt = new CryptoStream(mEncrypt, encryptor, CryptoStreamMode.Write))
            {
                using (StreamWriter sEncrypt = new StreamWriter(cEncrypt))
                {
                    sEncrypt.WriteLine(value);
                }
                encryptedText = mEncrypt.ToArray();
            }
        }
    }
    return Convert.ToBase64String(encryptedText);
}
```

6.4.2 Decryption Algorithm

```
public static string Decrypt(string key, string encryptedText)
{

    byte[] iv = new byte[16];
    byte[] buffer = Convert.FromBase64String(encryptedText);

    using (Aes aes = Aes.Create())
    {
        //aes.Padding = PaddingMode.None;
        aes.KeySize = 128;
        aes.Key = Encoding.UTF8.GetBytes(key);
        aes.IV = iv;

        ICryptoTransform decryptor = aes.CreateDecryptor(aes.Key, aes.IV);

        using (MemoryStream mEncrypt = new MemoryStream(buffer))
        {
            using (CryptoStream cEncrypt = new CryptoStream(mEncrypt, decryptor, CryptoStreamMode.Read))
            {
                using (StreamReader sEncrypt = new StreamReader(cEncrypt))
                {
                    return sEncrypt.ReadToEnd();
                }
            }
        }
    }
}
```

6.5 Hashing

Hashing Algorithm

Chilpass hashes the master password using PBKDF-2 or the Password-Based Key Derivation Function 2. PBKDF-2 is a function for deriving a key or hash from a password with a given set of parameters. It's parameters include a salt, a number of iterations, and a hashing algorithm.

```
/*
 * Creates a pseudorandom salt to make a hash unique between like passwords
 */
1 reference
private byte[] GenerateSalt()
{
    // generate a salt
    using (var rng = RandomNumberGenerator.Create())
    {
        rng.GetBytes(salt);
    }
    return salt;
}
```

- Salt: The salt is a pseudo-random generated string that is stored together with the master password in the password file.
- Password-Based Key Derivation Function 2 (PBKDF2)

```
/*
 * HashPassword method takes a string cleartext password as a paramater.
 *  Iterations is set to a million to combat brute force attacks. Slow for the user and attackers.
 *  Uses SHA512 hashing algorithm
 *  Returns the hashed string
 */
0 references
public static string PBKDF2(string password, int iterations)
{
    // TODO: Leave a notifaction that it is computing hash or running so the user doesn't think its dying
    string hashed = Convert.ToBase64String(KeyDerivation.Pbkdf2(
        password: password,
        salt: salt,
        prf: KeyDerivationPrf.HMACSHA256,
        iterationCount: iterations, // increase later
        numBytesRequested: 128 / 8));

    System.Diagnostics.Debug.WriteLine("Hash: " + hashed);
    return hashed;
}
```

- Iterations: The number of iterations corresponds with the complexity and the amount of time that the function takes to execute. Chilpass uses 10,000 iterations as its base amount as a balance between the user experience and security.
- Hashing Algorithm: PBKDF-2 defaults to using HMAC-SHA1 for hashing, which can still work in the context of the greater function, is not ideal as it is

recommended by various security professionals not to use SHA-1 as modern day technology makes it not a reliably safe hash. Therefore, Chilpass uses HMAC-SHA-256 for its internal hashing.

6.6 Random Generation

Random Generation is done through the use of .NET Cryptographic Random Number Generation methods and classes. These classes and methods are utilized to effectively mitigate the amount of bias that otherwise occurs when generating characters or numbers from a computer based random number generator.

7. Detailed Design

7.1 High Level Sequence Diagram

7.2 Class Diagram

7.3 Interfaces

8. Development Information / Resources

8.1 Database Management System

- SQLite serverless database

8.2 Application Building Tools

Visual Studio Windows Forms

- Open File
- Save File

8.3 Links and References

8.3.1 General Information

C# Cryptographic Services
C# Cryptography Documentation
Review Chapter 5 and Chapter 9

8.3.2 Keys

Key
KDF - Key Derivation Function
KEK - Key-Encryption-Key

8.3.3 Hashing

C# SHA-256

- SHA256 Class
- SHA256 Managed Class
- SHA256 Crypto Service Provider Class

8.3.4 Encryption

C# AES - Synchronous Encryption

- AES Class

8.3.5 Random Generation

C# Random Generation

- RNG Service Provider Class