

ISLAMIC UNIVERSITY OF TECHNOLOGY



VISUAL PROGRAMMING LAB

CSE 4402

---

# Java Password Manager and Decryptor App

---

*Author:*

Mueed Ibne Sami — 210041149

Miraj Mahmud Mahee — 210041101

Kazi Akib Zaoad — 210041117

# Contents

<b>1</b>	<b>Objective</b>	<b>2</b>
<b>2</b>	<b>Introduction</b>	<b>2</b>
2.1	Project Purpose and Problem Statement . . . . .	2
2.2	Key Functionalities and Features . . . . .	2
<b>3</b>	<b>System Design</b>	<b>3</b>
3.1	Overall System Architecture . . . . .	3
<b>4</b>	<b>Implementation Details</b>	<b>8</b>
4.1	High-Level Overview . . . . .	8
4.2	Critical Code Sections and Algorithms . . . . .	8
4.3	Design Choices . . . . .	10
<b>5</b>	<b>GUI Design</b>	<b>10</b>
5.1	User Interface Design and Layout . . . . .	10
<b>6</b>	<b>Testing and Evaluation</b>	<b>14</b>
6.1	Testing Strategies . . . . .	14
6.2	Challenges and Solutions . . . . .	14
<b>7</b>	<b>Conclusion</b>	<b>15</b>
7.1	Key Achievements . . . . .	15
7.2	Limitations and Future Improvements . . . . .	15

# 1 Objective

This report documents the development process, technical details, and functionalities of our Java application, including a Password Manager and a Decryptor App. The primary objective of this project is to provide a secure and user-friendly solution for managing passwords. Users can encrypt their passwords using a security key and later decrypt them as needed. This report also serves as the comprehensive documentation for our project.

## 2 Introduction

### 2.1 Project Purpose and Problem Statement

The main purpose of this project is to create a secure environment for users to manage their passwords. With increasing online services, users often struggle to remember multiple passwords. This application addresses the issue by providing a secure password manager where passwords are encrypted and stored securely, and a decryptor app to decrypt the passwords when needed.

### 2.2 Key Functionalities and Features

- **User Registration:** Users can register with personal information including a unique security key.
- **Login:** Users can log in using their username and password.
- **Password Management:** Users can add, update, and delete passwords.
- **Encryption:** Passwords are encrypted using a custom encryption process involving the main password and the security key given during user registration.
- **Decryption:** Users can decrypt passwords using the decryptor app by providing the encrypted password and security key.

## 3 System Design

### 3.1 Overall System Architecture

The system consists of two main components: the Password Manager App and the Decryptor App. The architecture is designed to ensure data security and ease of use. The Password Manager App handles user registration, login, and password management, while the Decryptor App is used to decrypt the encrypted passwords.

### Design Patterns and Technologies Used

- **Design Patterns:**

- **Singleton:** Ensures that only one instance of the database connection exists, reducing the overhead of creating multiple connections.
- **Factory:** Creates encryption and decryption objects, promoting flexibility and reusability of the code.

- **Technologies:**

- **Java:** The core programming language used for the application.
- **JavaFX:** Used for creating the graphical user interface.
- **MySQL:** Chosen for its lightweight nature and ease of integration with Java for the database.

- UML Class Diagram

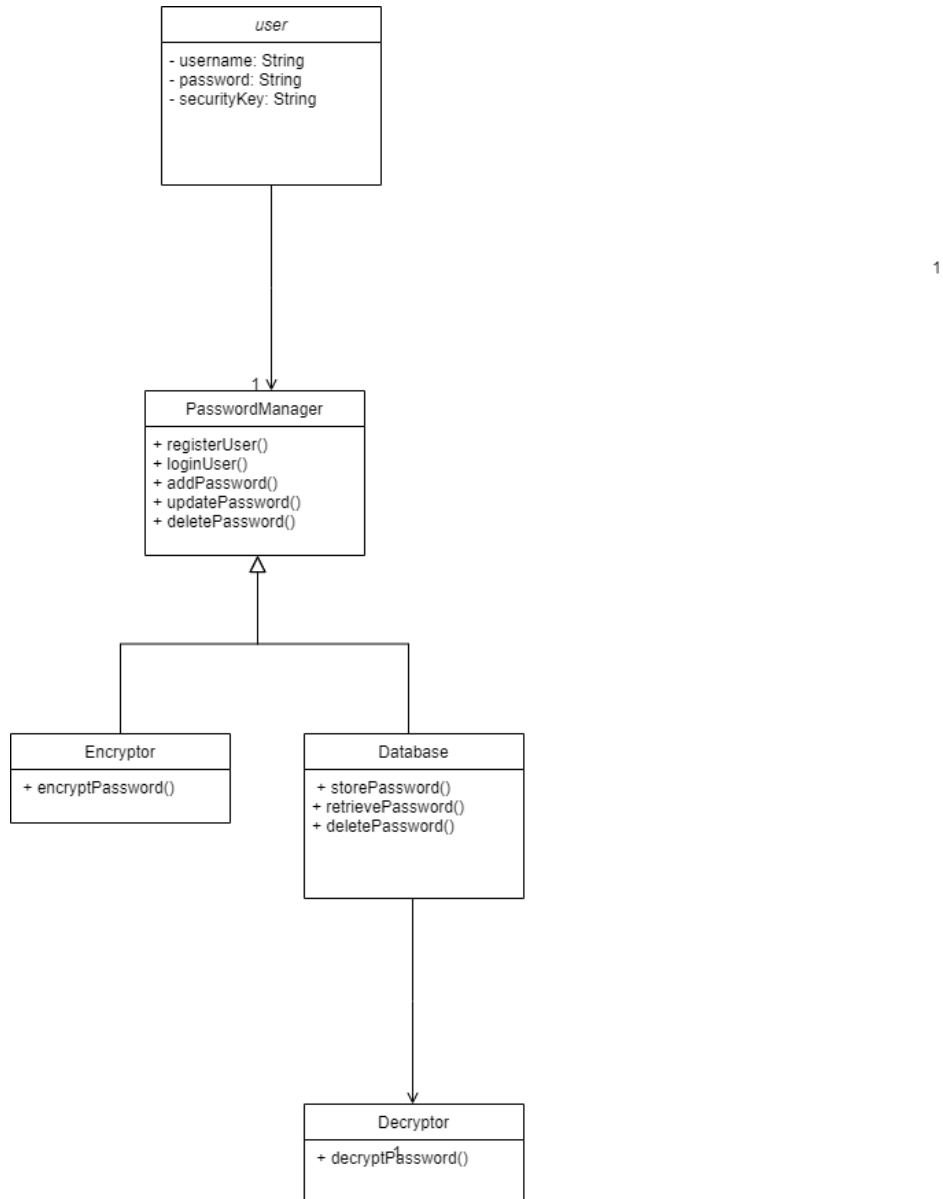


Figure 1: UML Class Diagram.

- Use Case Diagram

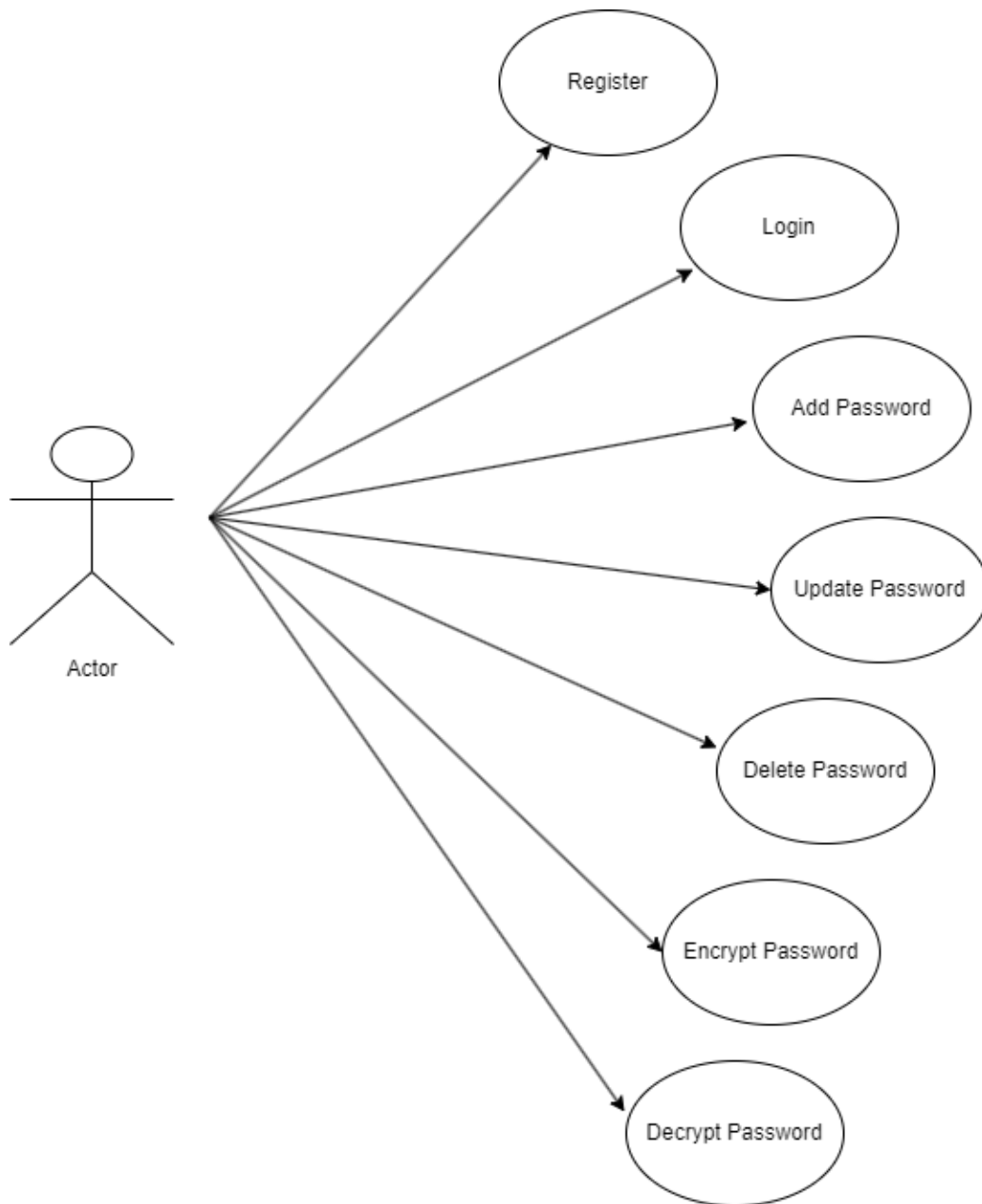


Figure 2: Use Case Diagram.

- Sequence Diagram

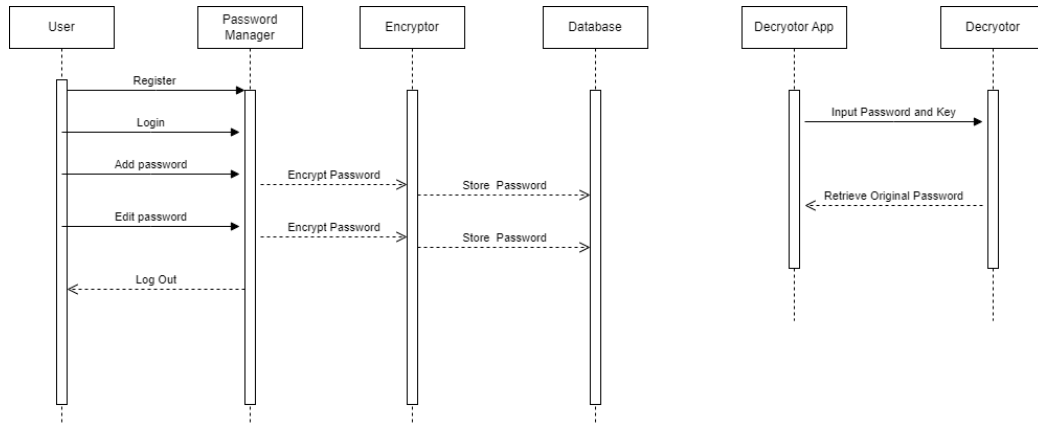


Figure 3: Sequence Diagram.

- Activity Diagram

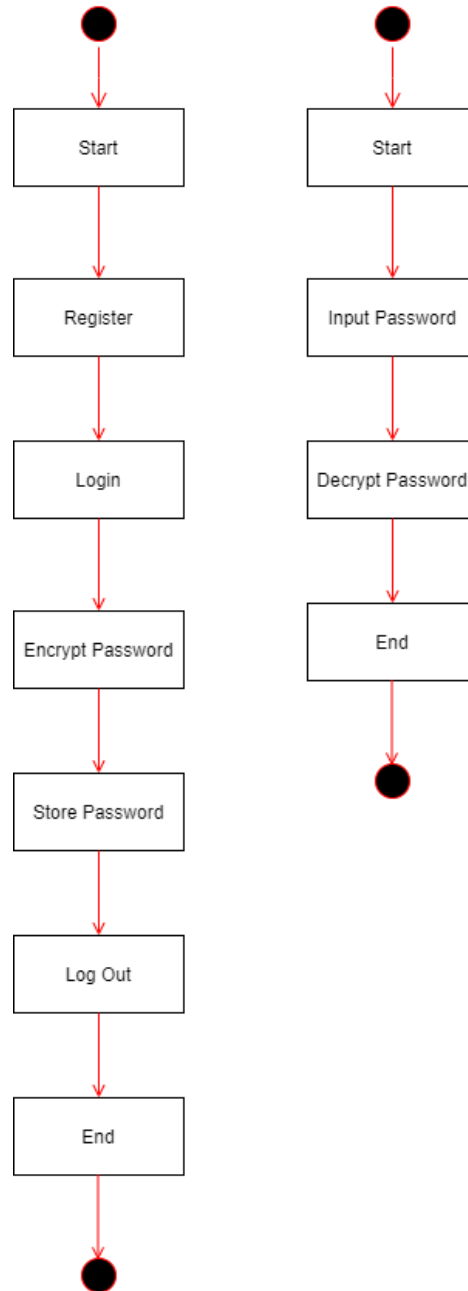


Figure 4: Activity Diagram.



## 4 Implementation Details

### 4.1 High-Level Overview

The implementation involves several key modules:

- **User Module:** Handles registration and authentication. Users register with a username, password, and a security key, which is used for encryption.
- **Password Module:** Manages password CRUD (Create, Read, Update, Delete) operations. Passwords are encrypted before being stored in the database.
- **Encryption Module:** Implements the custom encryption and decryption logic, ensuring that passwords are securely encrypted using the user's security key.

### 4.2 Critical Code Sections and Algorithms

- **Encryption Algorithm:**

```
1      public static void encryp(int[] asci, int key
2      , int len, int[] en_asci) {
3          int mod_key = key % 7;
4          for (int i = 0; i < len; ++i) {
5              switch (i % 8) {
6                  case 0:
7                      en_asci[i] = (asci[i] + (mod_key - 5)
8                      - 32 + 95) % 95 + 32;
9                      break;
10                     case 1:
11                         en_asci[i] = (asci[i] + (mod_key * 5)
12                         - 32 + 95) % 95 + 32;
13                         break;
14                     case 2:
15                         en_asci[i] = (asci[i] + ((key / 2) -
16                         5) - 32 + 95) % 95 + 32;
17                         break;
18                     case 3:
19                         en_asci[i] = (asci[i] + (mod_key + 7)
20                         - 32 + 95) % 95 + 32;
21                         break;
22                     case 4:
```

```

18         en_ascii[i] = (ascii[i] + mod_key - 32
19         + 95) % 95 + 32;
20         break;
21     case 5:
22         en_ascii[i] = (ascii[i] + (mod_key +
23         19) - 32 + 95) % 95 + 32;
24         break;
25     case 6:
26         en_ascii[i] = (ascii[i] - 4 - 32 + 95)
27         % 95 + 32;
28         break;
29     case 7:
30         en_ascii[i] = (ascii[i] + 5 - 32 + 95)
31         % 95 + 32;
32         break;
33     }
34 }
35 }

```

#### • Decryption Algorithm:

```

1 private static void decryptAscii(int[] en_ascii, int key,
2     int len, int[] de_ascii) {
3     int mod_key = key % 7;
4     for (int i = 0; i < len; ++i) {
5         switch (i % 8) {
6             case 0:
7                 de_ascii[i] = (en_ascii[i] - (mod_key -
8                 5) - 32 + 95) % 95 + 32;
9                 break;
10            case 1:
11                de_ascii[i] = (en_ascii[i] - (mod_key *
12                5) - 32 + 95) % 95 + 32;
13                break;
14            case 2:
15                de_ascii[i] = (en_ascii[i] - ((mod_key
16                / 2) - 5) - 32 + 95) % 95 + 32;
17                break;
18            case 3:
19                de_ascii[i] = (en_ascii[i] - (mod_key +
20                7) - 32 + 95) % 95 + 32;
21                break;
22            case 4:
23                de_ascii[i] = (en_ascii[i] - mod_key -
24                32 + 95) % 95 + 32;
25            }
26        }
27    }
28 }

```

```

19         break;
20     case 5:
21         de_ascii[i] = (en_ascii[i] - (mod_key +
22             19) - 32 + 95) % 95 + 32;
23         break;
24     case 6:
25         de_ascii[i] = (en_ascii[i] + 4 - 32 +
26             95) % 95 + 32;
27         break;
28     case 7:
29         de_ascii[i] = (en_ascii[i] - 5 - 32 +
30             95) % 95 + 32;
31         break;
32     }
33 }

```

### 4.3 Design Choices

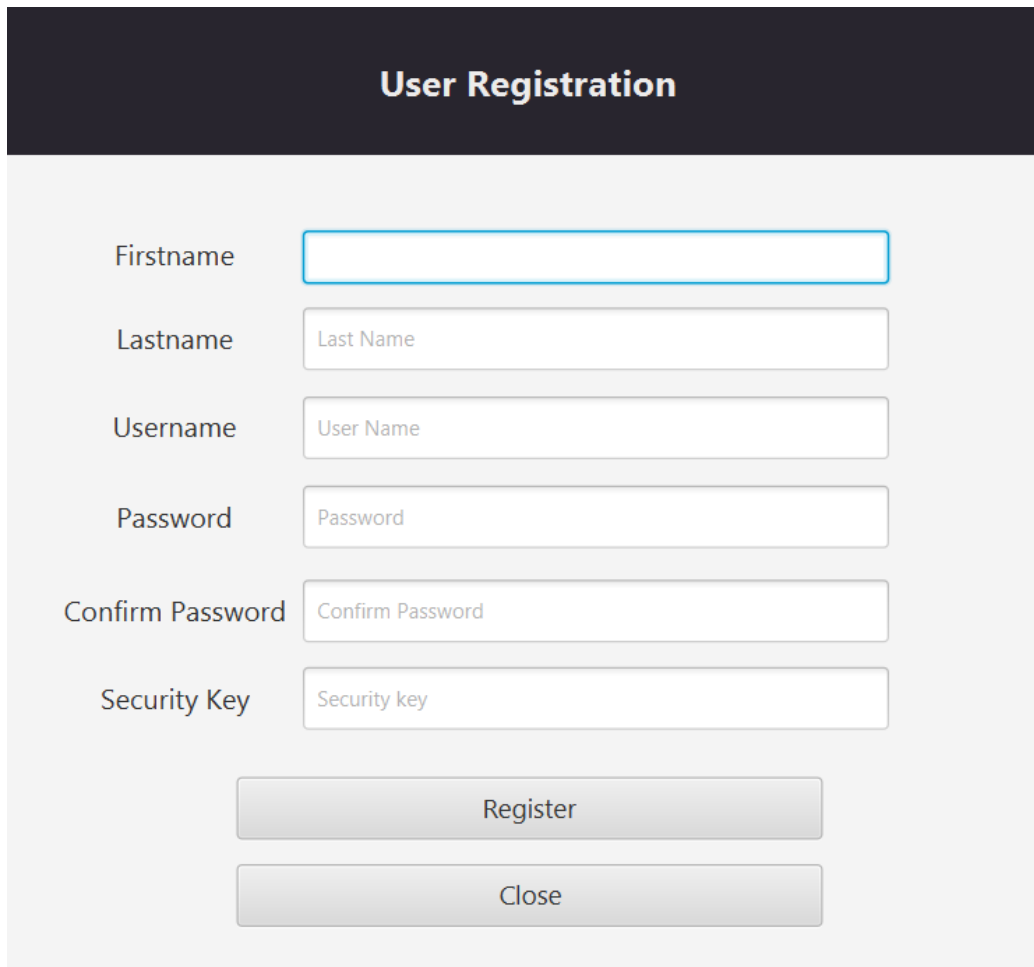
- **Database:** Chose MySQL for its lightweight and ease of integration with Java. It provides a simple way to store encrypted passwords locally.
- **Encryption:** Developed a custom encryption algorithm to ensure the security of passwords. The choice was driven by the need to tailor the encryption to the specific requirements of the project.

## 5 GUI Design

### 5.1 User Interface Design and Layout

The GUI is designed using JavaFX to provide a clean and intuitive interface. The main design principles followed are simplicity and user-centric design. The interface includes forms for user registration and login, as well as a main dashboard for password management.

- **Registration Screen:** Users put their credentials to create new account.

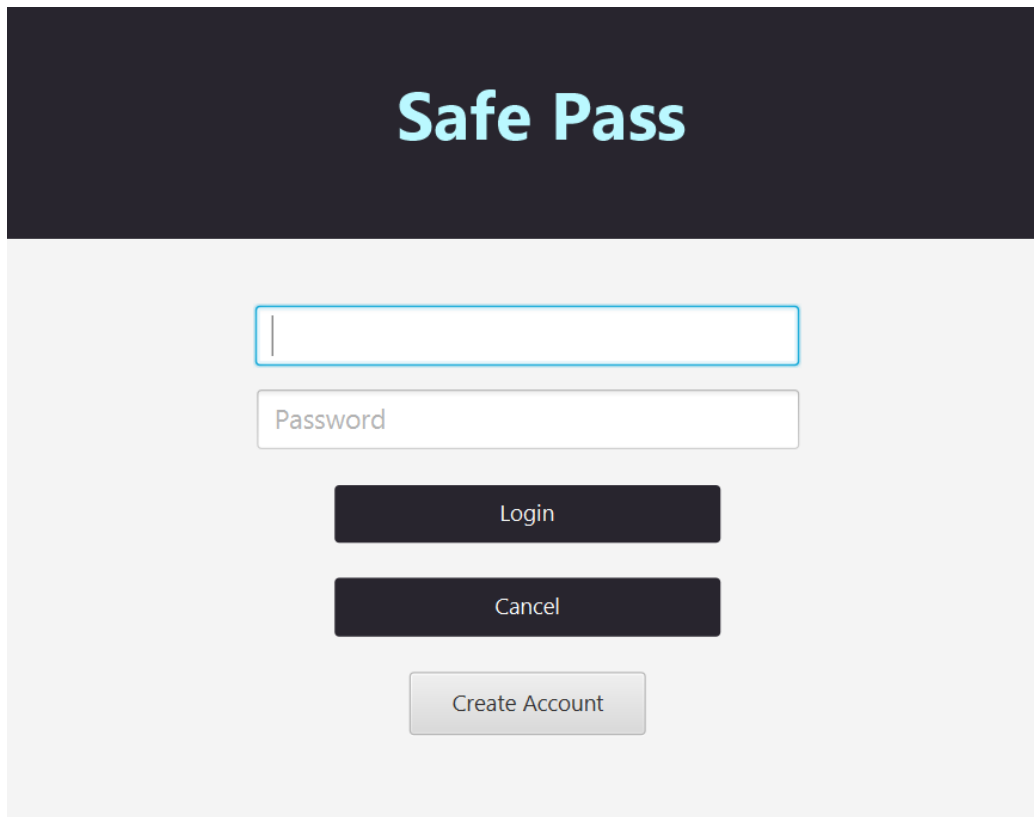


The image shows a 'User Registration' form with a dark header. The form contains six input fields: 'Firstname' (empty), 'Lastname' (placeholder 'Last Name'), 'Username' (placeholder 'User Name'), 'Password' (placeholder 'Password'), 'Confirm Password' (placeholder 'Confirm Password'), and 'Security Key' (placeholder 'Security key'). Below the fields are two buttons: 'Register' and 'Close'.

User Registration	
Firstname	<input type="text"/>
Lastname	<input type="text" value="Last Name"/>
Username	<input type="text" value="User Name"/>
Password	<input type="password" value="Password"/>
Confirm Password	<input type="password" value="Confirm Password"/>
Security Key	<input type="text" value="Security key"/>
<input type="button" value="Register"/>	
<input type="button" value="Close"/>	

Figure 5: User Registration page.

- **Login Screen:** Users enter their credentials (username and password) to access the app.



The image shows a login page for an application named "Safe Pass". The page has a dark blue header with the text "Safe Pass" in white. Below the header, there is a light gray background. In the center, there is a white input field with a blue border. Below this, there is a white input field with a gray border and the placeholder text "Password". Below the password field, there are three buttons: a dark blue "Login" button, a dark blue "Cancel" button, and a light gray "Create Account" button.

Figure 6: Login Page.

- **Main Screen:** After logging in, users are presented with a dashboard where they can view encrypted passwords, add new passwords, update existing ones, and delete passwords.

Welcome

Password er chinta?  
Aar na... Aar na

Add

Update

Delete

ID	App	User Name	Password	Comments
No content in table				

Figure 7: Dashboard.

- **Decryptor Screen:** Users enter the encrypted password and security key in the decryptor app to decrypt and view the original password.

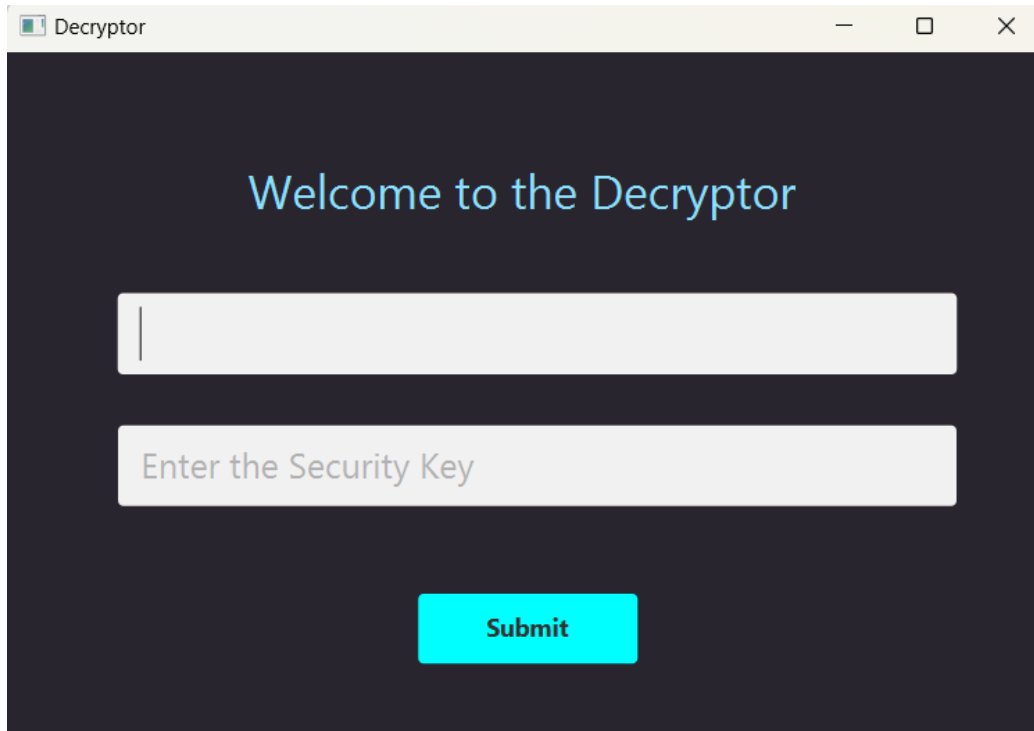


Figure 8: Decryptor App.

## 6 Testing and Evaluation

### 6.1 Testing Strategies

Initially, we faced a problem with some unprintable ASCII characters. We tried various ASCII numbers and found that after encryption, some characters were not printable after encryption. This was our initial test run. We discovered that some ASCII characters less than 32 are not printable. This was our finding from the test run. We then modified our algorithm to convert all the ASCII codes within the range of 32 to 126, solving this problem.

### 6.2 Challenges and Solutions

- **Challenge:** Ensuring encryption strength while maintaining performance.

- **Solution:** Optimized the encryption algorithm for better performance without compromising security.
- **Challenge:** Handling various user inputs and ensuring data validation.
  - **Solution:** Implemented robust input validation mechanisms to ensure data integrity and security.

## 7 Conclusion

### 7.1 Key Achievements

- Successfully implemented a secure password manager with encryption and decryption functionalities.
- Developed a user-friendly GUI for managing passwords, making it easy for users to navigate and perform operations.
- Ensured data security through custom encryption algorithms, providing users a secure way to store and manage their passwords.

### 7.2 Limitations and Future Improvements

- **Limitations:** Currently, the encryption algorithm is custom and may not be as secure as standardized algorithms. Additionally, the application is a desktop-based solution and may not be accessible from multiple devices.
- **Future Improvements:**
  - **Integration of Standardized Encryption Algorithms:** Plan to integrate standardized encryption algorithms, such as AES, to enhance security.
  - **Cloud Storage:** Implement cloud storage solutions to allow users to access their passwords from multiple devices.
  - **Password Strength Analysis:** Add features to analyze and suggest stronger passwords.
  - **Multi-Factor Authentication:** Enhance security by integrating multi-factor authentication during login.