

PASSWORD MANAGER USING CRYPTOGRAPHIC TECHNIQUES

Minor Project (CS801PC)

Submitted

*in partial fulfillment of the requirements for
the award of the degree of*

Bachelor of Technology

in

Computer Science and Engineering

by

KATHULA MADHAVI

(18261A0583)

Under the Guidance of

Mrs. P POORNIMA

Assistant Professor



Department of Computer Science and Engineering

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

GANDIPET, HYDERABAD – 500075, INDIA.

**MAHATMA GANDHI INSTITUTE OF TECHNOLOGY (Affiliated to
Jawaharlal Nehru Technological University Hyderabad)GANDIPET,
HYDERABAD – 500 075, Telangana**

CERTIFICATE



This is to certify that this is entitled **PASSWORD MANAGER USING CRYPTOGRAPHIC TECHNIQUES** is being submitted by Ms. **K.Madhavi** bearing Roll No:**18261A0583** in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** to **Jawaharlal Nehru Technological University Hyderabad** is a record of bonafide work carried out by her under our guidance and supervision.

Supervisor

Mrs. P. Poornima

Assistant Professor

Head of Department

Dr. C.R.K. Reddy

Professor

External Examiner



**MAHATMA GANDHI
INSTITUTE OF TECHNOLOGY**

Kokapet(Village), Gandipet, Hyderabad, Telangana - 500075. www.mgit.ac.in



MOTIVATE
INNOVATE
EMPOWER **24**
YEARS

CERTIFICATE

This is to certify that the project work entitled **Password Manager Using Cryptographic Techniques** submitted by **Kathula Madhavi(18261A0583)**, in partial fulfilment of requirements for the award of degree of Bachelor of Technology in Computer Science and engineering as specialization is a record of the bonafide work carried out under the supervision of **Mrs. P.Poornima**, and this has not been submitted to any other university or institute for award of degree or diploma.

Project Guide

Mrs. P. Poornima
Assistant professor
Dept. of CSE

Project Coordinator

Dr. A. Nagesh
professor
Dept. of CSE

EXTERNAL EXAMINER

Dr. C.R.K Reddy

Professor and HOD
Dept. of CSE

DECLARATION

This is to certify that the work reported in this project titled “**PASSWORD MANAGER USING CRYPTOGRAPHIC TECHNIQUES**” is a record of work done by me in the Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred to in the text. This report is based on the work done entirely by me and not copied from any other source.

The results embodied in this project have not been submitted to any other university or Institute for the award of any degree or diploma.

Kathula Madhavi

18261A0583

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. They have been a guiding light and source of inspiration towards the completion of the project.

I would like to express my sincere thanks to **Dr. K. Jaya Sankar, Principal MGIT**, for providing the work facilities in the college.

I am also thankful to **Dr. C.R.K Reddy, Professor & Head of Department**, Dept. of Computer Science and Engineering for providing excellent infrastructure and a conducive atmosphere for completing this project successfully.

I am also extremely thankful to my Project Coordinators, **Dr. A.Nagesh, Professor, Dept. of CSE, Mr. A.Ratnaraju, Assistant Professor, Dept. of CSE and Mr. V.Subbaramaiah, Assistant Professor, Dept. of CSE**, for their valuable suggestions and interest throughout the course of this project.

I would like to express my sincere gratitude and indebtedness to my Project Guide, **Ms. P Poornima, Assistant Professor, Dept. of CSE** who has supported me throughout the project with patience and knowledge.

I convey my heartfelt thanks to the lab staff for allowing me to use the required equipment whenever needed. Finally, I would like to take this opportunity to thank my family for their support throughout the work. I sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

Kathula Madhavi
18261A0583

TABLE OF CONTENTS

Certificate	i
Declaration	ii
Acknowledgement	iii
List of Figures	vi
List of Tables	vii
Abstract	viii
1. INTRODUCTION	1
1.1 Problem Definition	3
1.2 Existing System	3
1.3 Proposed System	3
1.4 Requirements Specification	4
1.4.1 Software Requirements	4
1.4.2 Hardware Requirements	4
2. LITERATURE SURVEY	7
3. METHODOLOGY	10
4. SYSTEM DESIGN	13
4.1 Password Managing Process	14
4.2 UML Diagrams	19

4.2.1 Sequence Diagram	19
4.2.2 Use Case Diagram	20
5. RESULTS	21
6. CONCLUSION AND FUTURE SCOPE	28
6.1 Conclusion	28
6.2 Future Scope	29
REFERENCES	31
APPENDIX	33

LIST OF FIGURES

Figure 3.1	AES Algorithm	10
Figure 3.2	Network Diagram	12
Figure 4.1	Structure of PMS	13
Figure 4.2.1	Sequence Diagram	19
Figure 4.2.2	Use Case Diagram	20
Figure 5.1	Login/SignUpFigure	21
Figure 5.2	Password Manager with various options	22
Figure 5.3	Add Password	23
Figure 5.4	Added Passwords	24
Figure 5.5	Search for a password using ID	24
Figure 5.6	Searched Password	25
Figure 5.7	Edit password using ID	25
Figure 5.8	Password Edited Successfully	26
Figure 5.9	Delete Password	27
Figure 5.10	Delete password	27

LIST OF TABLES

Table-2.1	Literature Survey for Password Manager	8
Table-6.1	Result of Functionality Testing	29

ABSTRACT

Many systems rely on passwords for authentication. Due to numerous accounts for different services, users have to choose and remember a significant number of passwords. Password-Manager applications address this issue by storing the user's passwords. They are especially useful on mobile devices, because of the ubiquitous access to the account passwords. Password-Managers often use key derivation functions to convert a master password into a cryptographic key suitable for encrypting the list of passwords, thus protecting the passwords against unauthorized, off-line access. Therefore, design and implementation flaws in the key derivation function impact password security significantly. Design and implementation problems in the key derivation function can render the encryption on the password list useless, by for example allowing efficient brute force attacks, or even worse direct decryption of the stored passwords. In this project, analysis of key derivation functions of popular Android Password-Managers with often startling results. With this analysis, the awareness of developers of security critical apps for security is raised, and an overview about the current state of implementation security of security-critical applications is provided.

1. INTRODUCTION

The objective of this Password Management System is to manage passwords for different accounts on the internet. The user shall be able to save all the username and passwords information of the accounts he holds on the internet using this application. These details shall be saved in the database in encrypted format. This will help the user to remember different usernames and passwords for accounts on the internet. The user shall be able to add an account, edit and delete an account using the system. The user has to login to the system in order to use this tool. It takes user login and password. It allows the user to change the password for the tool being used. This interface is developed using python and an encryption system is managed by AES algorithm. This Password Management System tool will allow the user to view all the different accounts he holds in a list view on the left screen. All the details related to the account like the user name, password etc shall be displayed on the screen. The password is visible in encrypted format. The user can unhide the password and view the password details. The user can save any number of account information using this tool.

Authentication is a process to ensure and confirm the user's identity before accessing a certain information. Most applications need a password for authentication. There are many types of authentication such as biometrics, cards, tokens and passwords. By using biometric authentication, it consumes a lot of money to install and to maintain the device. Therefore, the most common authentication method used by people is a password or Personal Identification Number (PIN).

However, since there are too many passwords to memorize, people tend to forget the password. They keep forgetting and ask for password recovery and these situations keep on repeating especially to the website that are rarely used. Another problem is the selection of the password. People create their password according to something that is related to them such as birth date, personal name, pet's name and phone number. Simple password, especially a password that is related to personal information is not recommended. This is because the simple password can

can easily be hacked by an attacker for example by bruteforce attack and if the password is exposed to the attacker, the entire user's information can be stolen. Hence, encrypted data is needed to protect the user information.

Besides, some individuals have a tendency to compose their passwords wherever they find helpful for them to recover the data quicker. For example, the sticky notes where they stick close to PC, smartphone notes, and also depend on browser cookies to remember. Passwords that are kept in many places are possible to be stolen or hacked by other people. Therefore, the aim of this project is to develop an android application that is able to link users authenticate data directly to its account and also be able to store password information securely in the database. Information in the application stored in the database is secured by using an encryption method.

1.1 Problem Definition

This project “Password Manager” is an application that allows you to generate and store all your passwords in a safe location. AES(Advanced Encryption Standard) algorithm is used to encrypt the information that needs to be secured. Password management is a set of principles and best practices to be followed by users while storing and managing passwords in an efficient manner to secure passwords as much as they can to prevent unauthorized access.

1.2 Existing System

The Password Management System was done manually. Users had to remember the list of passwords for different accounts on the internet. The user had to manually maintain a list of all usernames and passwords. This task was very tedious. In cases where privacy for user accounts was required, it was difficult to manually maintain this list of passwords. The user had to save a password details list in some word format so that it would help him to remember the username and passwords for different accounts on the internet.

1.3 Proposed System

This is a simple Password Management System graphical user interface. It allows the user to add account information for different accounts he holds on the internet. Password Management System basically stores the username and password details in an encrypted format in the database. The encryption is managed by the AES algorithm. The user has to login to the system. It checks for the validity of the user. The user shall be able to add an account. The user can add user name and password details and save to the database. The list of accounts held by the user is displayed on the left screen. When the user selects the particular account on the left panel, the details of the account gets displayed on the right screen. The password shall be displayed in encrypted format. Once the user selects the unhide option, he shall be able to view the password on screen. The user can add any number of accounts using this application. The encryption algorithm is very robust and uses the symmetric key block cipher which is a very effective way of handling passwords.

1.4 Requirements Analysis

The following are the requirements for the execution of the project:

1.4.1 Software Requirements

The following are the software requirements for **PASSWORD MANAGER USING CRYPTOGRAPHIC TECHNIQUES**:

Software: Python IDLE

Platform: Jupyter Notebook

Language: Python 3.3 and above

Packages required:

- 1) Cryptography
- 2) Sqlite3
- 3) Fernet
- 4) Tkinter

1.4.2 Hardware Requirements

- OS – Windows 7,8 or 10 (32 or 64 bit)
- Intel dual core
- 512 MB Ram
- Hard disk 80 GB
- RAM – 8GB
- KeyBoard Standard
- Mouse

Software Tools Used:

Cryptography

Cryptography is a package intended to provide utility for important cryptographic and linear algebra functions in Python. This package includes functions for generating primes, primality tests including AKS, Miller-Rabin, Baillie-PSW, functions for factoring integers including Pollard's P-1, Quadratic Sieve and Lenstra's Elliptic Curve factorization algorithm, functions for solving discrete logarithm problems including index calculus method, Pollard's Rho method for logarithms, baby-step-giant-step, and Pohlig-Hellman, support for working with Elliptic Curves including 'fast power algorithm' and baby-step-giant-step, general algebra functions such as Chinese Remainder and the Euclidean Extended Algorithm for gcd's, and support for matrices including Matrix and Linear Map objects that extend numpy's array to include more functionality such as solving non-square systems, solving systems over the field of integers, change-of-basis, etc.

Required Packages

- NumPy
- SymPy

NumPy

NumPy, an open source Python library, is used along with the pandas library to handle multidimensional data and perform complex scientific and mathematical operations on the data.

Sympy

Sympy is used to generate reference values for unit tests and some code generation.

SQLite3

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others.

Fernet

Uses `cryptography.fernet` for symmetric encryption/decryption with 256-bit key derived from secret of arbitrary size.

- Supports encryption of binary files

Tkinter

Tkinter is a library to render math on a Tkinter canvas. It is made for docal but making it work with at least with MathML is straightforward and is planned to be implemented.

Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Its uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. The purpose of Jupyter notebook is to provide a more accessible interface for code used in digitally-supported research.

Notebook name

The name displayed at the top of the page, next to the Jupyter logo, reflects the name of the `.ipynb` file. Clicking on the notebook name brings up a dialog which allows you to rename it. Thus, renaming a notebook from `—Untitled0` to `—My first notebookl` in the browser, renames the `Untitled0.ipynb` file to `My first notebook.ipynb`.

Menu bar

The menu bar presents different options that may be used to manipulate the way the Notebook functions.

2. LITERATURE SURVEY

Passwords are used for many reasons such as to protect data, systems, and networks. We use passwords to authenticate users, to protect files and other stored information. User authentication happens when human-to-computers interact to grant rights and the process of checking user account permissions for access to resources. There are three types of authentication that can be classified into which are knowledge-based authentication, physical based authentication and biometric-based authentication.

In this research we have a discussion on knowledge-based authentication with encryption in securing the data. When applying One Time Password (OTP), the admin has to consider the security of the data in the database and during the transaction. There are different types of password, personal identification number (PIN) and passphrase. PINs are rarely used as the only form of authentication for IT system access because they are short and less complex.

Passphrase is another type of password that is relatively long and complex. When choosing which password authentication to be used in the system, admin have to consider the complexity and security of the data. Park et al said that to increase the security level scheme, the process of encryption and decryption happen during and after the transaction. However, as a developer when embedded a security component in the system, we should also focus on usability and deployability of the system.

Organizations need to protect the confidentiality, integrity, and availability of passwords so that all authorized users can use passwords successfully as they need. They found that a simple password authentication system is a one-way authentication system and serves as a multi-user system to authenticate legitimate users that can log in and access system resources. They also stated that in a simple password authentication system, the session key exchanges and distributes between server and users.

Table 2.1: Literature Survey for Password Manager Using Cryptographic Techniques

S.no	Title of paper	Year	Author	Objective	Merits	Demerits
1.	Password manager using Encryption Techniques	2016	1.Zuhri Arafah zulkifli 2.Hidaya zukri 3.Md Rashid 4.Morkhushaini Awang	Effective use of Encryption Techniques	-It provides security	-Limited password import options from other password manager accounts
2.	Stateless or token-based password managers	2016	1.S.Agholor 2.A.S.Sodiya 3.A.T.Akinwale 4.O.J.Adernian	A local piece of hardware, Such as a flash USB device contains a key to unlock your particular account	-Your credentials are stored in a separate device	-if you lose your device, you lose access -This method usually requires proprietary hardware and software
3.	Web-based or online password manager services	2017	1.Carlos Luevanos 2.John Elizanaras 3.Khai Hirschi	Stores the data on your device	-Eliminates the risk that someone will breach your password vault -It's a free service	-You can access your vault on only one device -If you lose your device, you lose your vault

4.	A Comparative Usability Evaluation of Traditional Password Managers	2010	1.Ambarish Karole 2.Nitesh Saxena 3.Nicolas Christin	A comparative study of three traditional passwords(online manager, portable managers-phone manager or a USB manager	-Portable managers provide more promising password management approach	-portable managers do not offer usability
----	---	------	--	--	--	--

3. METHODOLOGY

The approach is to implement the AES Encryption algorithm for storing data in the database. The AES Encryption process contains the round-functions such as substitute bytes, shift rows, mix column and add round key. Figure 3.1 shows the same functions applied to the decryption process are in inverse steps. The process is looping 10 rounds, as the first 9 rounds it will pass through all four (4) functions, and the last round the mix column will not be carried out.

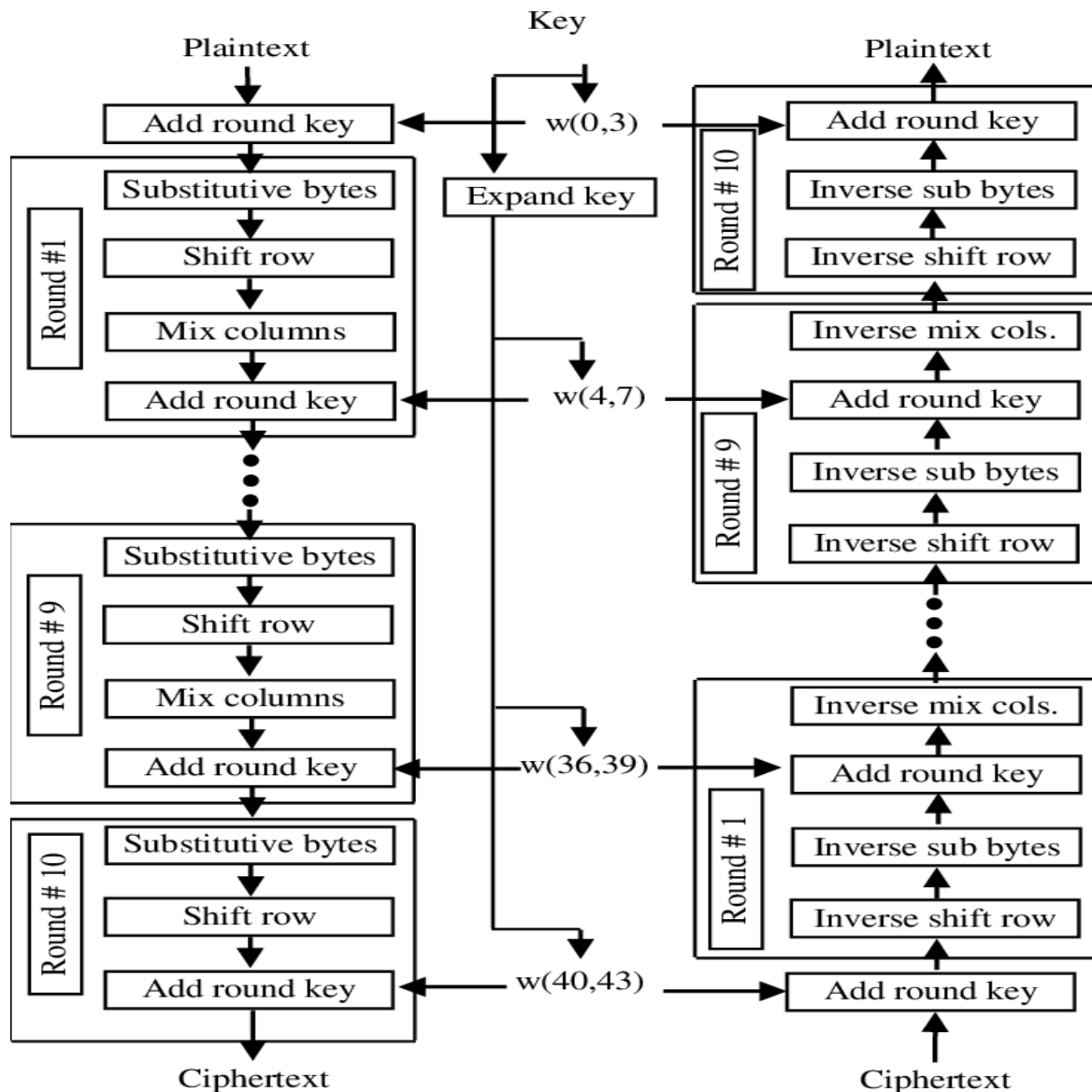


Figure 3.1 AES Algorithm

AES Algorithm

AES encryption function is used to encrypt the information that needs to be secured. It happens in a background process during the saving of a new memo created by the user. Cipher class provides access to implementation of cryptographic ciphers for encryption and decryption. Once getInstance method is called, the data will be encrypted to base 64 format by using the AES algorithm.

```
Cipher c=Cipher.getInstance("AES");  
c.init(Cipher.ENCRYPT_MODE,key);  
byte[] enVal= c.doFinal(Data.getBytes());  
String encryptedValue = Base64.encodeToString(enVal,Base64.Default);
```

The code below used to decrypt all data encrypted in the database. The decryption process happened during the editing memo whereby the user can view the username and password saved earlier. The decrypt mode is used to initialize the cipher to decryption mode. Decode method will decrypt the data in string value.

```
Cipher c=Cipher.getInstance("AES");  
c.init(Cipher.DECRYPT_MODE,key);  
byte[] decodedVal= Base64.decode(encryptData,Base64.Default);  
byte[] decValue=c.doFinal(decodedVal);  
String decryptedValue = new String(decValue);
```

Network Diagram

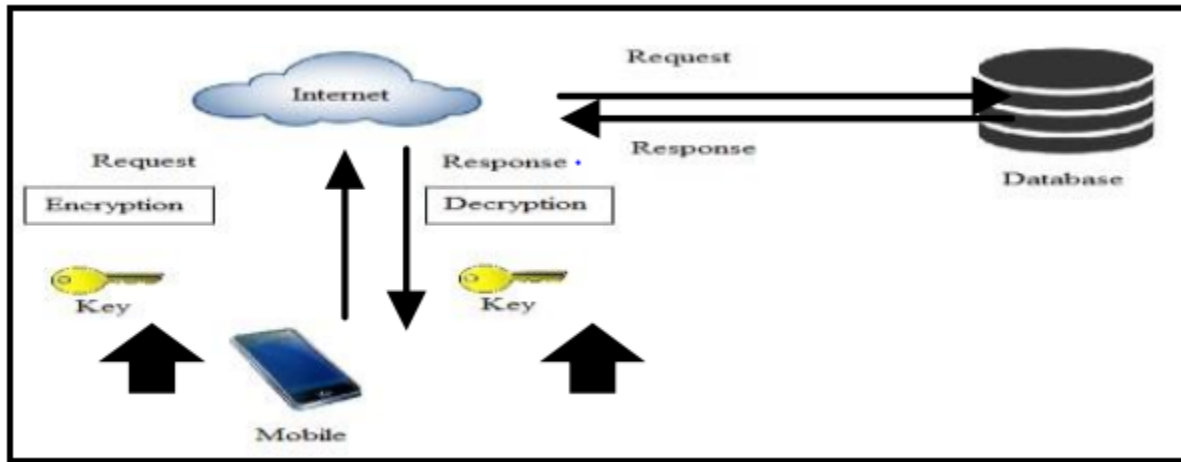


Figure 3.2: Network Diagram

Figure 3.2 shows the network diagram of the Password Management Application that requires the network coverage. When a user creates a new memo in the application, the AES algorithm will generate a secret key for encryption purposes. Then the encrypted data of username and password is saved in the online database. The process of decryption happens whenever a user retrieves the information from the database. The secret key was used to decrypt back the information.

4. SYSTEM DESIGN

Everyone has the security problem of private information which can be solved by password protection and extraction. Password Management System (PMS) is used to manage a variety of username and password info on the Internet, which is stored in the disk file (called password database) in the form of ciphertext. The user can handle the information in the database as long as he has succeeded in passing the identification. Usually, we set each operation on the Internet starting through the PMS (e.g.: adding the website address you browse frequently into PMS). When it's time to enter username and password, we can get them from the corresponding record.

After the identification, the user can add, modify and get info from his database. The modified information will be restored to the database after being encrypted automatically by PMS. Therefore, if the user remembers the identification password, then he remembers all.

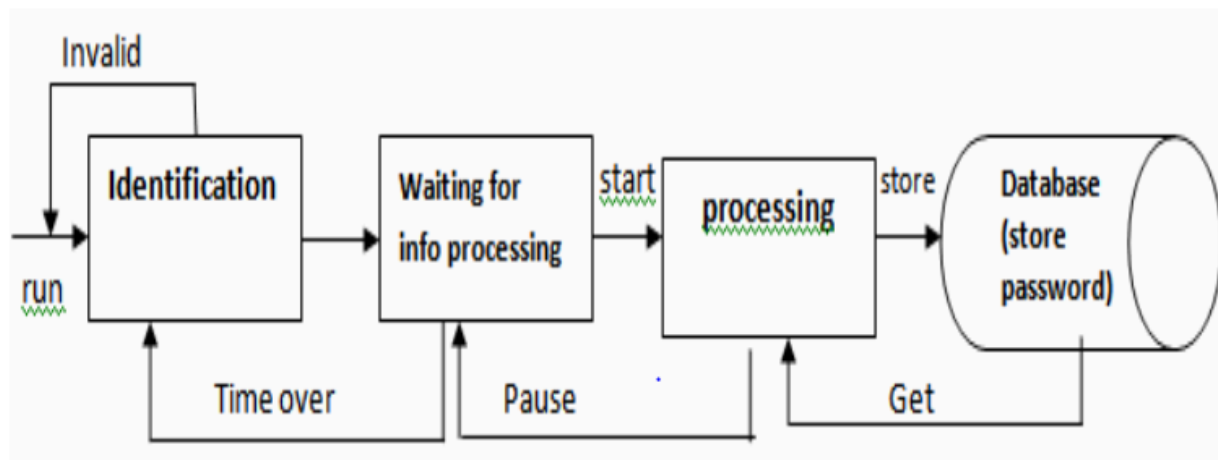


Figure 4.1: Structure of PMS

4.1 Password Management

Passwords are a set of strings provided by users at the authentication prompts of web accounts. Although passwords still remain as one of the most secure methods of authentication available to date, they are subjected to a number of security threats when mishandled. The role of password management comes in handy there. Password management is a set of principles and best practices to be followed by users while storing and managing passwords in an efficient manner to secure passwords as much as they can to prevent unauthorized access.

Challenges in Password Management

There are many challenges in securing passwords in this digital era. When the number of web services used by individuals are increasing year-over-year on one end, the number of cyber crimes is also skyrocketing on the other end. Here are a few common threats to protecting our passwords:

- Login spoofing - Passwords are illegally collected through a fake login page by cybercriminals.
- Sniffing attack - Passwords are stolen using illegal network access and with tools like keyloggers.
- Shoulder surfing attack - Stealing passwords when someone types them, at times using a micro-camera and gaining access to user data.
- Brute force attack - Stealing passwords with the help of automated tools and gaining access to user data.
- Data breach - Stealing login credentials and other confidential data directly from the website database.

Password vaults

Password vaults or password managers are programs that keep your passwords in a secure location. When you use this software, the passwords are encrypted, and you use a single master password to access them. Depending on the tool being used, these passwords may be stored on a secure website, or on your local computer. In using it, you only need to remember one password to view the database of other passwords for various sites and apps. Some of the password managers available include

- LastPass
- RoboForm
- NordPass
- DashLane

Each of the above-mentioned tools are free, although premium versions can be purchased. All of them have versions available that work with mobile devices and Windows computers, while RoboForm also has a version that works with Macs. By installing an extension to your browser, you navigate to a site, enter your username and password, and have the option to save it to the password manager. After this, the user can navigate to the bookmarked sites and automatically login. At any time, you can view and edit the login information you've saved.

Password Manager Tools

Creating strong passwords is a necessary procedure in order to stay safe online. Unfortunately complex passwords are difficult to remember; besides, you need to create a separate password for each account. For example, if you have 10 login passwords for different Internet accounts, memorizing becomes very difficult, especially if you are changing them every 3 months. Fortunately there is a solution for this problem as for many others.

Password manager helps you to organize your passwords according to each Internet service used. You can set complex passwords for each account and retrieve them easily when needed.

KeePass

KeePass is a free open source password manager. You can put all of your passwords in one database, which is locked with one master key or a keyfile. So you have only to remember one single master password to unlock the whole database. The databases are encrypted using secure encryption algorithms like (AES, Rijndael) and the Twofish algorithm.

Master Password

This application uses a different approach to secure your passwords. It is based on an ingenious password generation algorithm that guarantees your passwords can never be lost. While password managers generally save your passwords in an encrypted vault or upload them to the cloud for safe-keeping, unfortunately, they make you dependent on syncing, backups, or Internet access. Master Password has none of these downsides. Its passwords aren't stored, they are generated on-demand from your name, the site, and your master password.

Master Password is based on a stateless algorithm that frees it from the reliance on storage of secrets. Since your generated passwords aren't saved to your device, there is no risk you'll be forced to divulge them to aggressive peers. And since these passwords don't need to be backed up or synchronized between devices over the network, there is no risk of them getting intercepted.

Password Security Issues and Threats

There are several threats to authentication procedures and data that come from different methods to penetrate the process. The major threat of authentication is the human itself, where users forget their authentication key, losses or damage. This case happened because of mistakes and carelessness of the users itself that can affect the reliability and management of the authentication system.

Some threats can also come from the mobile technical problem, for example the software or device used contain vulnerability or corruption that can affect the authentication process.

Other vulnerabilities such as Viruses and Trojan can also cause the process to interrupt. Threats described above are related to the software problem, where sometimes hardware also can be a factor of threat to authentication processes such as hardware attacks. Karen (n.d) classifies the threats on password into groups which are threats that take advantage of weak passwords and password hashes, such as password guessing & cracking; threats that replace passwords; and threats that involve attackers reusing compromised passwords.

According to Kajal (2013) attackers commonly compromise passwords in three ways; through password capturing; password guessing and capturing and social engineering.

Protection on Password

According to Cheng and Ding (2012), to be able to protect the password, a password protection system should be able to meet the following criteria.

1. The protection should offer the strongest security assurance: the system should be able to defeat attacks from rootkits which undermine the operating system, because kernel rootkits keyloggers are not uncommon in cyberspace.
2. The protection should induce little or no modification on the operating system and be fully compatible with existing browsers. This is due to the fact that proprietary operating systems such as Windows and Mac OS are more widely used than open source operating systems.
3. On the user side, they must not require user possession of extra devices such as USB token and mobile phone as the protection scheme should be as simple as possible. In the other hand no changes should be needed on the server side.
4. The cost for the protection system should be as low as possible in terms of time delay during the password authentication session and the overall computation load on the platform. The crucial part of this is that the user should not notice any delay in an authentication session.

Password management deals with issues like phishing attacks and malware, thus needing protection against both of them. Gajek et al. (2009) used a secure compartment which functions as a proxy to help the user's authentication as the password protection approach. For

instance, TruWallet and TruWalletM used different techniques to secure the authentication proxy which securely stores the user credentials and properly submits them to a remote server. The main disadvantages of these schemes are the architectural change on the platform and the high. In addition, it is challenging to isolate the browser and the user interface due to the Virtualization Based Password Protection against Malware in Untrusted OSs 203 enormous code size. Contrast this with Gajek et al. (2009) propose the use of trusted mobile devices for the users to delegate their credentials to perform a task. Meanwhile Cheng and Ding (2012) presented a virtualization based password input protection system, which is composed of a novel user hypervisor interaction channel, a keyboard stroke interception mechanism, and a hypervisor based SSL client.

4.2 UML DIAGRAMS

Unified Modified Language is a standard language for writing software blueprints. It can be used to visualize,specify,construct and document the artifacts of a software intensive system. Modelling is a proven and well-accepted engineering technique.

4.2.1 Sequence Diagram

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.

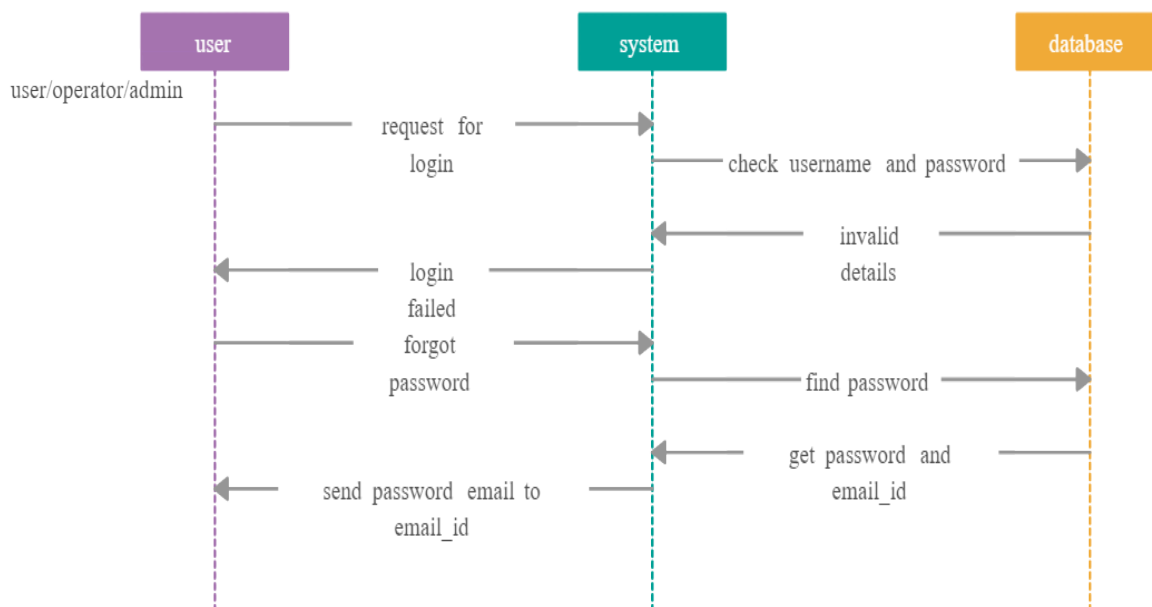


Figure 4.2.1: Sequence diagram of Password Manager

4.2.2 Use Case Diagram

A Use case diagram in the Unified Modified Language(UML) is a type of behavioural diagram defined by and created from a Use-case analysis. The main purpose of the use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted as shown in Figure 4.2.2

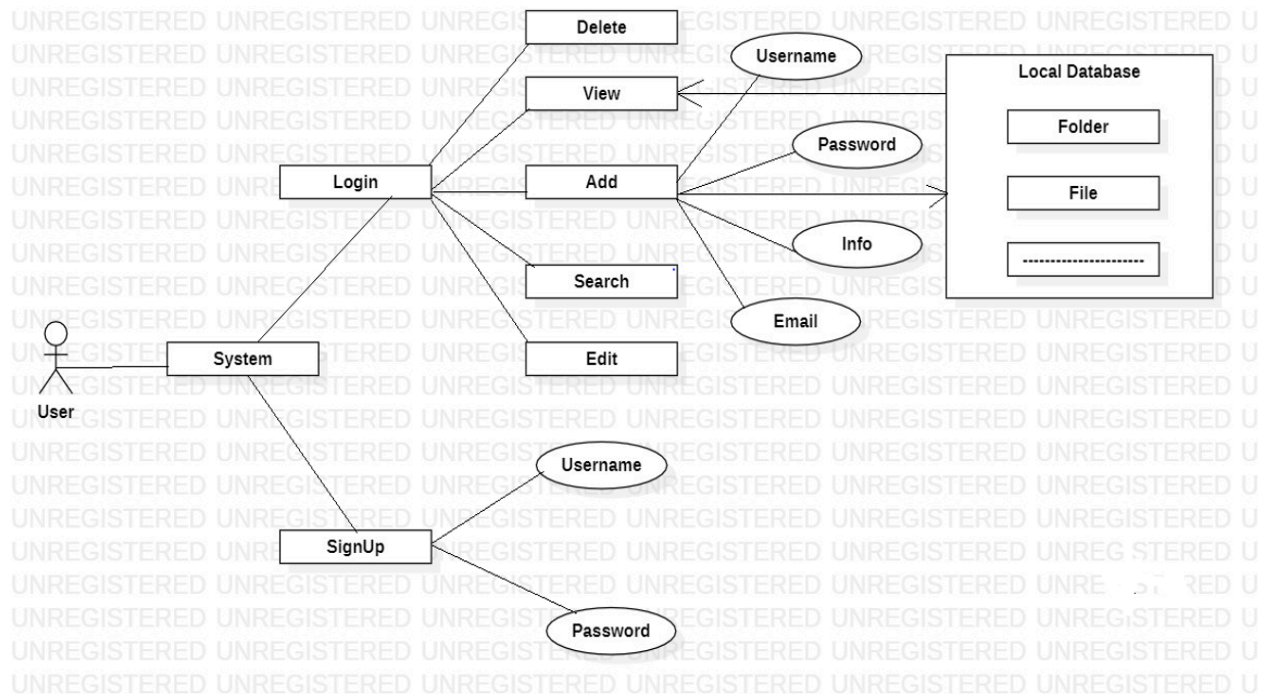


Figure 4.2.2: Use Case diagram of Password Manager

5.RESULTS

GUI With Username And Password:



Figure 5.1: Login/SignUp

Figure 5.1 This is a GUI application. It is displayed with a welcome message , username and password labels, Login,Signup and Info buttons.The users need to create their accounts by signing up. This application allows the users to provide their username and password in order to login or sign up into their account to store passwords.

Password Manager with View, Add, Search, Edit and Delete Passwords:

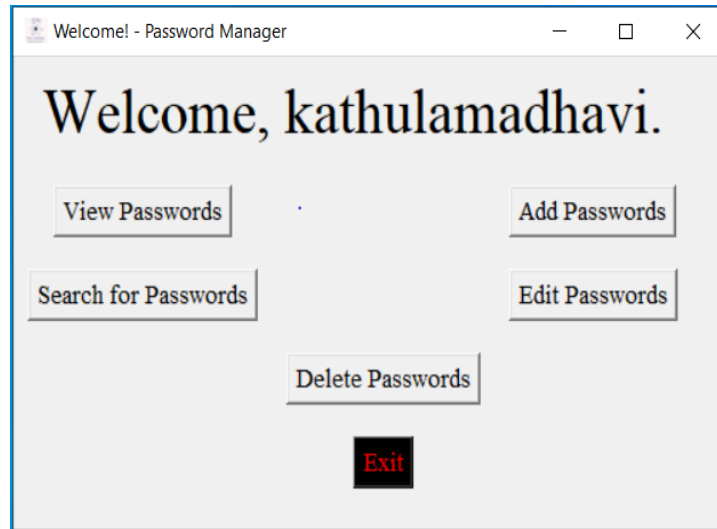
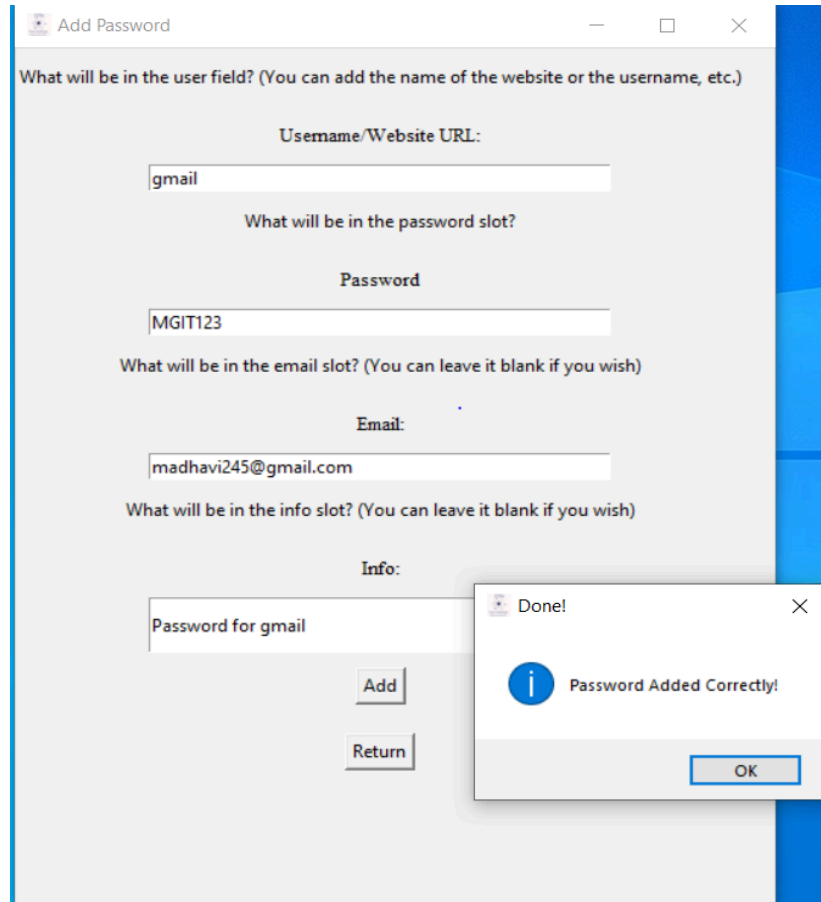


Figure 5.2: Password Manager with various options

Figure 5.1 When the users log into their accounts, password manager provides with many options such as View passwords, Add Passwords, Search Passwords, Edit Passwords, Delete Passwords.

Add Password:



The image shows a Windows-style dialog box titled "Add Password". It contains four text input fields with the following labels and values:

- Username/Website URL:** The input field contains the text "gmail".
- Password:** The input field contains the text "MGIT123".
- Email:** The input field contains the text "madhavi245@gmail.com".
- Info:** The input field contains the text "Password for gmail".

Below the input fields are two buttons: "Add" and "Return". A smaller "Done!" dialog box is overlaid on the bottom right, displaying an information icon and the message "Password Added Correctly!" with an "OK" button.

Figure 5.3: Add Password

Figure 5.3 Passwords can be added by providing the UserName/Website URL, Password, Email and Info.

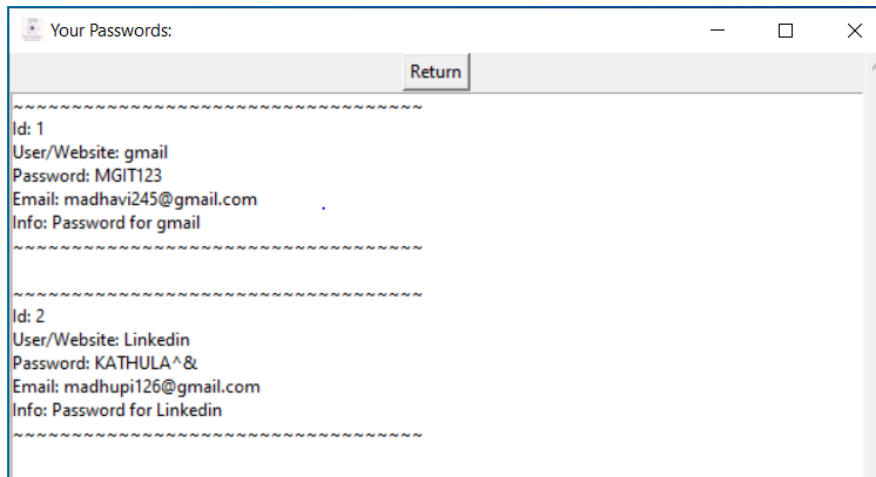


Figure 5.4: Added Passwords

Figure 5.4 Passwords that are added are stored at one place by generating the IDs.

Search Password: Passwords can be searched using ID, Email,Website URL

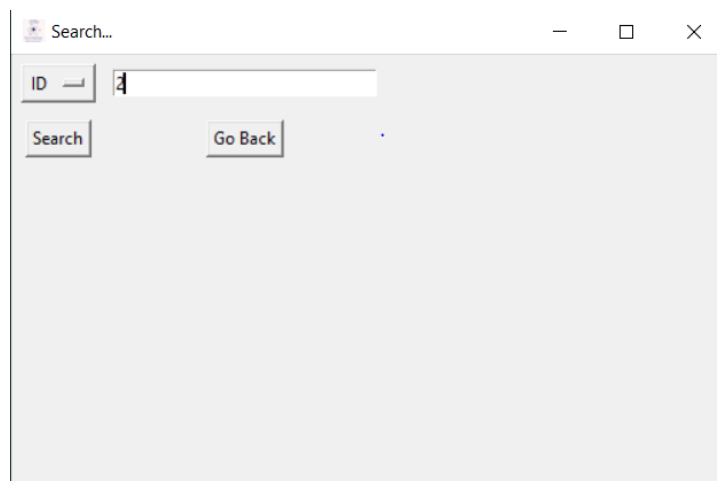


Figure 5.5: Search for a password using ID

Figure 5.5 The user can choose the searching options. Searching options are ID, Email, User/Website.



Figure 5.6: Searched Password

Figure 5.6: The searched password is displayed

Edit Password: Edit passwords by selecting the ID of the record

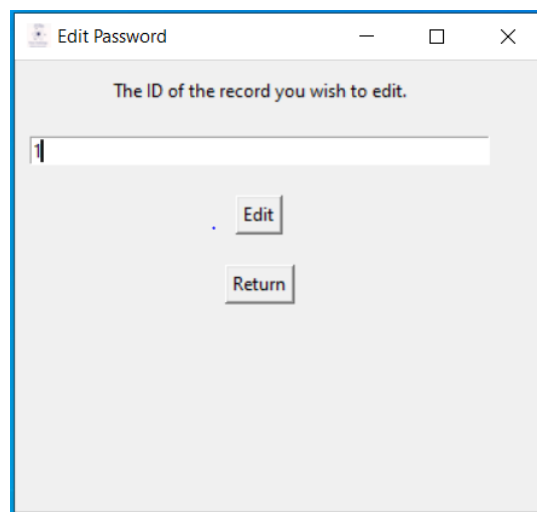


Figure 5.7: Edit password using ID

Figure 5.7 The ID of the record which the user wants to edit needs to be provided to edit the passwords.

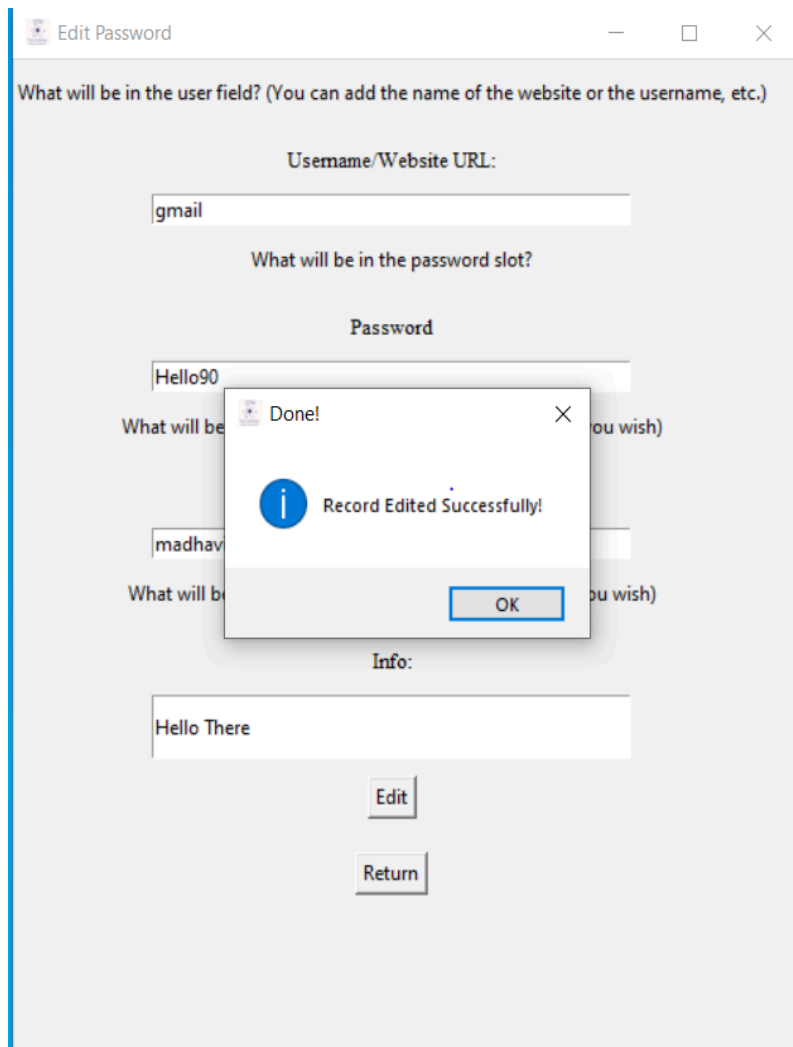


Figure 5.8: Password Edited Successfully

Figure 5.8 Enter the new password and the message is displayed that the record is edited successfully.

Delete Password: Delete the password by selecting the ID of the record

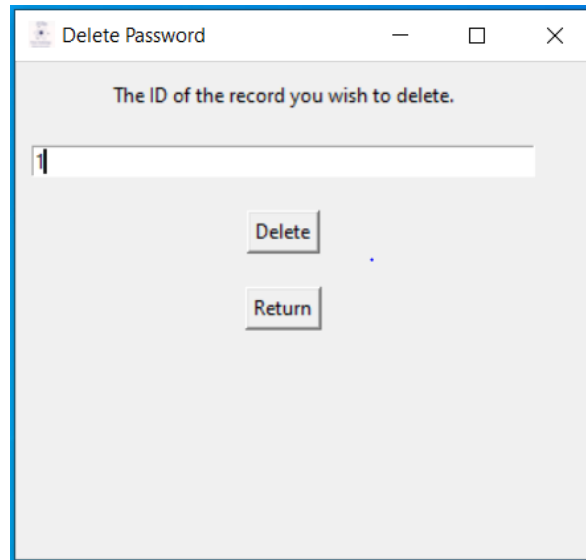


Figure 5.9: Delete Password

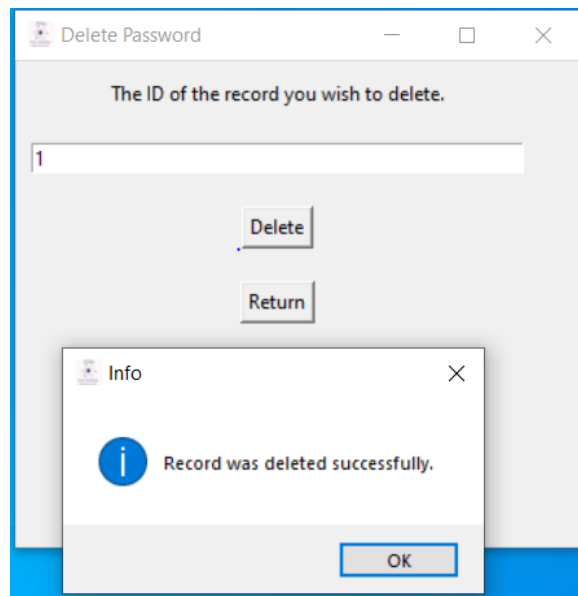


Figure 5.10: Delete password

Figure 5.9,5.10 Passwords can be deleted by providing the ID.

6.CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION:

This project has been approved to become the solution for users to manage their password more efficiently after functionality testing that has been made by a qualified tester. This application can help users manage and keep their password of different accounts securely. The secure part is implemented in the application by using an Advance Encryption Standard (AES) algorithm. In this application, the encryption happened during the process of saving a memo created by the user before entering the database. Other features also included in the application is the password strength checker. The password strength checker gives users the benefit to check their password strength.

This application is well constructed and able to give satisfaction to the end users. But, there are still other additional features that can be add in future development of this application in term of technology use.

Findings

The focus of the project is to develop an application that keeps the password and implements the encryption algorithm. The functionality testing was conducted to verify the functionality of the application performing according to the specification that has been planned from an earlier project development process. The process of testing is done by using an android device pre-installed with the application. The testing has been carried out by three experts among the lecturers of Computer Sciences. The test case which lists all the modules in the application and the android device was given to the expertes. The expert has checked and confirmed that the Password Management Application is working correctly and received positive comments from the experts. Figure 6.1 showed the result of the functionality testing on the Password Management Application.

Table 6.1 Result of Functionality Testing

Module	TestCase	Expected Result	Outcome	Result (Pass/Fail)
1	Login and SignUp	Users can sign up and login to the application	Users can sign up and login to the application	Pass
2	Add data	Users can add data entry such as title, Email, username, password and website	Users can add data entry such as title, Email, username, password and website	Pass
3	View data	Users can view data enter in a list	View data enter in a list	Pass
4	Update data	Users can update data from the original entry	Update data from the original entry	Pass
5	Delete data	Users can delete the data	Delete data from the list	Pass
6	Change Password	Users can change their password	Change their password	Pass
7	Sign in to website	Users can access to website such as twitter	Access to website such as twitter	Pass

6.2 FUTURE SCOPE:

On a personal level, cyber-aware people have started using secure digital password managers across their devices. Use of password managers has resulted in people adopting best practices by default, overriding a natural inclination to be sloppy. Many have adopted 2-factor authentication, and have become more cognizant of the benefits of VPNs to further protect their

passwords and other information. These individuals are aware of the value of password security and are more likely to practice better cyber hygiene in the workplace too.

On a business level, conscientious companies have installed enterprise-level privileged access management (PAM) software and are enforcing password management best practices across their organizations. PAM software has enabled companies to introduce automation to password management, so passwords can be changed, rotated, and expired on an automated schedule. Plus, passwords can be better managed when an employee leaves the company or when another high-risk event has occurred.

Password use can be tracked and reported on, and employees' actions can be monitored and recorded as they access the sensitive information protected by company passwords. And PAM software can help companies establish and prove compliance to fulfill their industry's audit requirements for protecting passwords.

Passwords are the staple of secure access to accounts and sensitive information. They will remain so for the foreseeable future, despite advancements in biometric authentication which simply augments password interactions.

REFERENCES

- [1]Al-Hussain, A., & Al-Rassan, I. (2010). A biometric-based authentication system for web services mobile users. Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia - MoMM '10, 447. <http://doi.org/10.1145/1971519.1971596>
- [2]Ben-Asher, N., Kirschnick, N., Sieger, H., Meyer, J., Ben-Oved, A., & Möller, S. (2011). On the need for different security methods on mobile phones. Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '11, 465. <http://doi.org/10.1145/2037373.2037442>
- [3]Cheng, Y., & Ding, X. (2012). Virtualization based password protection against malware in untrusted operating systems. In the International Conference on Trust and Trustworthy Computing (pp. 201-218). Springer Berlin Heidelberg.
- [4]Ciampa, M., & Green, B. (2010). Are Password Management Applications Viable? An Analysis of User Training and Reactions, 1–11.
- [5]Gajek, S., Löhr, H., Sadeghi, A. R., & Winandy, M. (2009). TruWallet: trustworthy and migratable wallet-based web authentication. In Proceedings of the 2009 ACM workshop on Scalable trusted computing (pp. 19-28). ACM.
- [6]Gaw, S., & Felten, E. W. (2006). Password management strategies for online accounts. Proceedings of the Second Symposium on Usable Privacy and Security - SOUPS '06, 44. <http://doi.org/10.1145/1143120.1143127>
- [7]Jeslet, D. S., Sivaraman, G., Uma, M., Thangadurai, K., & Punithavalli, M. (2010). Survey on Awareness and Security Issues in Password Management Strategies, 10(4), 19–23.
- [8]Kajal, A. (2013). Secure Password Selection method. International Journal of Advanced Research in Computer and Communication Engineering, 2 (7), 2613-2615.
- [9]Karen Scarfone, Murugiah Souppaya, “Guide to Enterprise Password Management (Draft) ”, US Department of Commerce, NIST, Special Publication 800-118.

- [10]Liang, S., Liu, J., Zhang, R., & Wang, C. (2010). A modified AES algorithm for the platform of Smartphone. Proceedings - International Conference on Computational Aspects of Social Networks, CASoN'10, 749–752. <http://doi.org/10.1109/CASoN.2010.172>
- [11]Rayarikar, R., Upadhyay, S., & Pimpale, P. (2012). SMS Encryption using AES Algorithm on Android. International Journal of Computer Applications, 50(19), 12–17. <http://doi.org/10.5120/7909-1038>
- [12]Schlöglhofer, R., & Sametinger, J. (2012). Secure and usable authentication on mobile devices. Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia - MoMM '12, 257. <http://doi.org/10.1145/2428955.2429004> Sujithra, M., & Padmavathi, G. (2012). Next generation biometric security system: an approach for mobile device security. Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology - CCSEIT '12, 377–381. <http://doi.org/10.1145/2393216.2393280>

APPENDIX

Main.py

```
import sqlite3 # Import sqlite for database
import main_def_func # Define all the functions used here
import password_keeper # Goes to another section where all the passwords are kept (encrypted)
import password_keeper_def_func # Defines all functions for the password_keeper
import cryptographyDefFunc # Defines all the functions for encrypting, decrypting, getting and
setting the keys, etc.
from tkinter import * # Used for the graphical user interface (gui)
from tkinter import messagebox
import os
root = Tk()
root.title("Login / Signup")
root.iconbitmap("ExtraSupportContent/florestechtechnologylogo.ico")
width = 500
height = 300
root.geometry(str(width) + "x" + str(height))
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
x_coordinate = int((screen_width/2) - (width/2))
y_coordinate = int((screen_height/2) - (height/2))
root.geometry(str(width) + "x" + str(height) + "+" + str(x_coordinate) + "+" + str(y_coordinate))
# Create Labels and Adjust them in main window
welcome_label = Label(root, text="Welcome!", font=("Times New Roman", 30))
welcome_label.grid(row=1, column=3, columnspan=100, padx=10, pady=10)
username_label = Label(root, text="Username:", font=("Times New Roman", 15))
username_label.grid(row=5, column=3, padx=3, pady=10)
password_label = Label(root, text="Password:", font=("Times New Roman", 15))
password_label.grid(row=7, column=3, padx=3, pady=10)
# Create Entry and Adjust them in main window
username_entry = Entry(root, width=50)
username_entry.grid(row=5, column=5, columnspan=2)
password_entry = Entry(root, width=50)
password_entry.grid(row=7, column=5, columnspan=2)
# Commands for buttons
def login():
    if (username_entry.get() == "" or password_entry.get() == ""):
        messagebox.showwarning("Warning", "The username or password fields are blank!")
    else:
        username = username_entry.get() # gets username
        password = password_entry.get() # gets password
        try:
            key = cryptographyDefFunc.get_key(username) # Gets the key for the specified use
```

```

        # Because the username and password were encrypted when inserting into table we need
        to compare them
        if main_def_func.searchUser(key, username, password) == True:
            root.destroy()
            password_keeper.mainPasswordKeeper(username) # We give all the necessary
parameters
        else:
            messagebox.showerror("Error", "User has not been found. Please try again, or create an
account.")
            except Exception as error:
                messagebox.showerror("Error", error)
def signup():
    if (username_entry.get() == "" or password_entry.get() == ""):
        messagebox.showwarning("Warning", "The username or password fields are blank!")
    else:
        try:
            main_def_func.tableCreate() # Searches to see if database has all ready been created
        except sqlite3.OperationalError as error:
            messagebox.showwarning("Error", error)
        username = username_entry.get() # gets username
        password = password_entry.get() # gets password
        try:
            cryptographyDefFunc.set_key(username) # sets key for encryption for specified user
            encrypted_user = cryptographyDefFunc.encrypt_some(username, password) # encrypts
user and password
            encrypted_username = encrypted_user[0]
            encrypted_password = encrypted_user[1]
            main_def_func.signup(encrypted_username,
                                encrypted_password) # creates record in first table with encrypted data
            password_keeper_def_func.createTable(username) # Creates second table for that
specified user
            root.destroy()
            password_keeper.mainPasswordKeeper(username) # We give all the necessary
parameters
        except Exception as error:
            messagebox.showwarning("Error", error)
def info():
    os.system('notepad ExtraSupportContent/about.txt')
# Create Buttons and Adjust them in main window
login_button = Button(root, text="Log In", font=("Times New Roman", 13), command=login)
login_button.grid(row=20, column=4, pady=15)
signup_button = Button(root, text="Sign Up", font=("Times New Roman", 13),
command=signup)
signup_button.grid(row=20, column=6, pady=15)
info_button = Button(root, text="Info", font=("Times New Roman", 13), command=info)

```

```
info_button.grid(row=1000, column=5, padx=15, pady=15)
root.mainloop()
```

Main_def_fun.py

```
import sqlite3
from cryptography.fernet import Fernet
def signup(username, password):
    # Inserts username and password to database (both are encrypted)
    connection = sqlite3.connect("registryUsers.db")
    cursor = connection.cursor()
    cursor.execute("INSERT INTO users VALUES(?,?)", (username, password))
    connection.commit()
    connection.close()
def searchUser(key, username, password):
    # Compares if the user and password are correct and if the account exists
    # Gets a hold of the key of the specified user to decrypt the info
    f = Fernet(key)
    # Connect to database
    conn = sqlite3.connect("registryUsers.db")
    # Create a cursor
    c = conn.cursor()
    # Query database
    c.execute("SELECT * FROM users")
    items = c.fetchall()
    for item in items:
        decrypted_user = f.decrypt(item[0])# Decrypts the username of the user
        original_user = decrypted_user.decode()# Decode the encoded user
        decrypted_pass = f.decrypt(item[1]) # Decrypts the password of the user
        original_pass = decrypted_pass.decode()# Decode the encoded pass
        if original_user == username and original_pass == password:
            return True
    # Commit changes
    conn.commit()
    # Close connection
    conn.close()
def tableCreate():
```

```

# Creates database table
connection = sqlite3.connect("registryUsers.db")
cursor = connection.cursor()
cursor.execute("""CREATE TABLE users (
    username text,
    password text
)""")
connection.commit()
connection.close()

```

password_keeper.py

```

import password_keeper_def_func # Defines all the functions used in password_keeper
import cryptographyDefFunc # Defines all the functions for encrypting, decrypting getting and
setting the keys, etc.
import sqlite3 # Import sqlite for database
from tkinter import * # Used for the graphical user interface (gui)
from tkinter import messagebox
from PIL import ImageTk, Image
import os
def mainPasswordKeeper(username):
    root = Tk()
    root.title("Welcome! - Password Manager")
    root.iconbitmap("ExtraSupportContent/florestechologylogo.ico")
    width = 500
    height = 300
    root.geometry(str(width) + "x" + str(height))
    screen_width = root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()
    x_coordinate = int((screen_width / 2) - (width / 2))
    y_coordinate = int((screen_height / 2) - (height / 2))
    root.geometry(str(width) + "x" + str(height) + "+" + str(x_coordinate) + "+" +
str(y_coordinate))
    # Create Labels
    welcome_label = Label(root, text="Welcome, " + username + ".", font=("Times New Roman",
30))
    welcome_label.grid(row=1, column=1, columnspan=50, padx=10, pady=10)
    # Define Functions for buttons
    def view():

```

```

try:
    # Gets the key for the specified user
    key = cryptographyDefFunc.get_key(username)
    # Shows all passwords, usernames, emails and info + rowid

    root.destroy()
    password_keeper_def_func.show_all(key, username)
except Exception as error:
    messagebox.showerror("Error", error)
    print(error)
def add():
    try:
        # Sees if table has all ready been created
        password_keeper_def_func.createTable(username)
    except sqlite3.OperationalError as error:
        pass
        # messagebox.showerror("Error", error)
    try:
        root.withdraw()
        add_new = Tk()
        add_new.title("Add Password")
        add_new.iconbitmap("ExtraSupportContent/florestechtechnologylogo.ico")
        width2 = 500
        height2 = 600
        add_new.geometry(str(width2) + "x" + str(height2))
        x_coordinate2 = int((screen_width / 2) - (width2 / 2))
        y_coordinate2 = int((screen_height / 2) - (height2 / 2))
        add_new.geometry(str(width2) + "x" + str(height2) + "+" + str(x_coordinate2) + "+" +
str(y_coordinate2))
        # Create Labels
        info1_label = Label(add_new, text="What will be in the user field? (You can add the
name of the website or the username, etc.)")
        info1_label.grid(row=0, column=0, pady=10)
        user_label = Label(add_new, text="Username/Website URL:", font=("Times New
Roman", 10))
        user_label.grid(row=2, column=0, pady=10)
        info2_label = Label(add_new, text="What will be in the password slot?")
        info2_label.grid(row=6, column=0, pady=10)
        password_label = Label(add_new, text="Password", font=("Times New Roman", 10))

```

```

password_label.grid(row=8, column=0, pady=10)
info3_label = Label(add_new, text="What will be in the email slot? (You can leave it
blank if you wish)")
info3_label.grid(row=12, column=0, pady=10)
email_label = Label(add_new, text="Email:", font=("Times New Roman", 10))
email_label.grid(row=14, column=0, pady=10)
info4_label = Label(add_new, text="What will be in the info slot? (You can leave it blank
if you wish)")
info4_label.grid(row=18, column=0, pady=10)
info_label = Label(add_new, text="Info:", font=("Times New Roman", 10))
info_label.grid(row=20, column=0, pady=10)
# Create Entry
user_entry = Entry(add_new, width=50)
user_entry.grid(row=4, column=0)
password_entry = Entry(add_new, width=50)
password_entry.grid(row=10, column=0)
email_entry = Entry(add_new, width=50)
email_entry.grid(row=16, column=0)
info_entry = Entry(add_new, width=50)
info_entry.grid(row=22, column=0, ipady=10)
# Command for button
def submit_add():
    # Encrypts all the info given by the user
    encrypted_info = cryptographyDefFunc.encrypt_all(username, user_entry.get(),
password_entry.get(), email_entry.get(), info_entry.get())
    # Gives the encrypted info to submit to the db (data base)
    password_keeper_def_func.add_one(username, encrypted_info[0], encrypted_info[1],
encrypted_info[2],
                                encrypted_info[3])
    messagebox.showinfo(title="Done!", message="Password Added Correctly!")
    user_entry.delete(0, END)
    password_entry.delete(0, END)
    email_entry.delete(0, END)
    info_entry.delete(0, END)
def go_back():
    add_new.destroy()
    root.deiconify()
# Create Buttons
submit_btn = Button(add_new, text="Add", command=submit_add)

```



```

submit_btn.grid(row=26, column=0, pady=10)
return_btn = Button(add_new, text="Return", command=go_back)
return_btn.grid(row=27, column=0, pady=10)
except Exception as error:
    messagebox.showerror("Error", error)
add_new.mainloop()
def search():
    top = Toplevel()
    top.title("Search...")
    top.iconbitmap("ExtraSupportContent/florestechtechnologylogo.ico")
    top.geometry(str(width) + "x" + str(height))
    top.geometry(str(width) + "x" + str(height) + "+" + str(x_coordinate) + "+" +
str(y_coordinate))
    try:
        # Create a Tkinter Var
        clicked = StringVar()
        clicked.set("Choose a Searching Option")
        drop = OptionMenu(top, clicked, "ID", "Name", "Email")
        drop.grid(row=0, column=0, padx=5, pady=5)
        search_entry = Entry(top, width=30)
        search_entry.grid(row=0, column=10, padx=5, pady=5)
        def search():
            search_result = search_entry.get()
            if search_result == "":
                messagebox.showerror("Error", "Searching field is empty. :/")
            if (clicked.get()=="ID"):
                # Gets the key for the specified user
                key = cryptographyDefFunc.get_key(username)
                # Searches for a match in database
                password_keeper_def_func.id_lookup(key, username, search_entry.get())
                print("If blank, none where found.")
            elif (clicked.get() == "Name"):
                # Gets the key for the specified user
                key = cryptographyDefFunc.get_key(username)
                # Searches for a match in database
                password_keeper_def_func.name_lookup(key, username, search_entry.get())
                print("If blank, none where found.")
                pass
            elif clicked.get() == "Email":

```

```

        # Gets the key for the specified user
        key = cryptographyDefFunc.get_key(username)
        # Searches for a match in database
        password_keeper_def_func.email_lookup(key, username, search_entry.get())
        print("If blank, none where found.")
        pass
    else:
        messagebox.showerror("Error", "Invalid Selection. Please try again.")
        show_label = Label(top, text=clicked.get()).grid(row=4, column=0, padx=5, pady=5)
        search_btn = Button(top, text="Search", command=search).grid(row=2, column=0,
padx=5, pady=5)
    def go_back():
        top.destroy()
        return_btn = Button(top, text="Go Back", command=go_back)
        return_btn.grid(row=2, column=10, padx=5, pady=5)
        top.mainloop()
    except Exception as error:
        print(error)
def edit():
    root.withdraw()
    add_new = Tk()
    add_new.title("Edit Password")
    add_new.iconbitmap("ExtraSupportContent/florestechologylogo.ico")
    width2 = 350
    height2 = 300
    add_new.geometry(str(width2) + "x" + str(height2))
    x_coordinate2 = int((screen_width / 2) - (width2 / 2))
    y_coordinate2 = int((screen_height / 2) - (height2 / 2))
    add_new.geometry(str(width2) + "x" + str(height2) + "+" + str(x_coordinate2) + "+" +
str(y_coordinate2))
    messagebox.showinfo("Info", "Once updated there is no going back")
    # Create Labels
    info0_label = Label(add_new, text="The ID of the record you wish to edit.")
    info0_label.grid(row=0, column=0, pady=10, padx=10)
    # Create Entry
    id_entry = Entry(add_new, width=50)
    id_entry.grid(row=4, column=0, padx=10, pady=10)
    # Command for button
    def edit_record():

```

```

id_num_edit = id_entry.get()
if id_entry.get() == "":
    messagebox.showerror("Error", "Invalid id")
else:
    add_new.destroy()
    edit_new = Toplevel()
    edit_new.title("Edit Password")
    edit_new.iconbitmap("ExtraSupportContent/florestechologylogo.ico")
    width3 = 500
    height3 = 600
    x_coordinate3 = int((screen_width / 2) - (width3 / 2))
    y_coordinate3 = int((screen_height / 2) - (height3 / 2))
    edit_new.geometry(str(width3) + "x" + str(height3))
    edit_new.geometry(str(width3) + "x" + str(height3) + "+" + str(x_coordinate3) + "+" +
str(y_coordinate3))
    # Create Labels
    info1_label = Label(edit_new,
text="What will be in the user field? (You can add the name of the website or the
username, etc.)")
    info1_label.grid(row=0, column=0, pady=10)
    user_label = Label(edit_new, text="Username/Website URL:", font=("Times New
Roman", 10))
    user_label.grid(row=2, column=0, pady=10)
    info2_label = Label(edit_new, text="What will be in the password slot?")
    info2_label.grid(row=6, column=0, pady=10)
    password_label = Label(edit_new, text="Password", font=("Times New Roman", 10))
    password_label.grid(row=8, column=0, pady=10)
    info3_label = Label(edit_new, text="What will be in the email slot? (You can leave it
blank if you wish)")
    info3_label.grid(row=12, column=0, pady=10)
    email_label = Label(edit_new, text="Email:", font=("Times New Roman", 10))
    email_label.grid(row=14, column=0, pady=10)
    info4_label = Label(edit_new, text="What will be in the info slot? (You can leave it
blank if you wish)")
    info4_label.grid(row=18, column=0, pady=10)
    info_label = Label(edit_new, text="Info:", font=("Times New Roman", 10))
    info_label.grid(row=20, column=0, pady=10)
    # Create Entry
    user_entry = Entry(edit_new, width=50)

```

```

user_entry.grid(row=4, column=0)
password_entry = Entry(edit_new, width=50)
password_entry.grid(row=10, column=0)
email_entry = Entry(edit_new, width=50)
email_entry.grid(row=16, column=0)
info_entry = Entry(edit_new, width=50)
info_entry.grid(row=22, column=0, ipady=10)
# Command for button
def edit_add():
    # Encrypts all the info given by the user

    encrypted_info = cryptographyDefFunc.encrypt_all(username, user_entry.get(),
password_entry.get(),
                                email_entry.get(), info_entry.get())
    password_keeper_def_func.update_one(username, encrypted_info[0],
encrypted_info[1],
                                encrypted_info[2], encrypted_info[3], id_num_edit)
    messagebox.showinfo(title="Done!", message="Record Edited Successfully!")
    go_back()
def go_back():
    edit_new.destroy()
    root.deiconify()
# Create Buttons
submit_btn = Button(edit_new, text="Edit", command=edit_add)
submit_btn.grid(row=26, column=0, pady=10)
return_btn = Button(edit_new, text="Return", command=go_back)
return_btn.grid(row=27, column=0, pady=10)
edit_new.mainloop()
def go_back():
    add_new.destroy()
    root.deiconify()
# Create Buttons
submit_btn = Button(add_new, text="Edit", command=edit_record)
submit_btn.grid(row=26, column=0, pady=10)
return_btn = Button(add_new, text="Return", command=go_back)
return_btn.grid(row=27, column=0, pady=10)
add_new.mainloop()
def delete():
    root.withdraw()

```

```

delete_record = Tk()
delete_record.title("Delete Password")
delete_record.iconbitmap("ExtraSupportContent/florestechologylogo.ico")
width2 = 350
height2 = 300
delete_record.geometry(str(width2) + "x" + str(height2))
x_coordinate2 = int((screen_width / 2) - (width2 / 2))
y_coordinate2 = int((screen_height / 2) - (height2 / 2))
delete_record.geometry(str(width2) + "x" + str(height2) + "+" + str(x_coordinate2) + "+" +
str(y_coordinate2))
messagebox.showwarning("Warning", "Once deleted there is no going back!")
# Create Labels
info0_label = Label(delete_record, text="The ID of the record you wish to delete.")
info0_label.grid(row=0, column=0, pady=10, padx=10)
# Create Entry
id_entry = Entry(delete_record, width=50)
id_entry.grid(row=4, column=0, padx=10, pady=10)
def go_back():
    delete_record.destroy()
    root.deiconify()
def delete_passwr():
    try:
        if messagebox.askokcancel("Sure tho?", "Are you sure you want to delete this
record?") == 1:
            id_var = id_entry.get()
            print(id_entry.get())
            print(type(id_entry.get()))
            print(username)
            print(type(username))
            password_keeper_def_func.delete_one(username, id_var)
            messagebox.showinfo("Info", "Record was deleted successfully.")
        else:
            messagebox.showinfo("Info", "Transaction cancelled. Phew! That was a close one.")
            go_back()
    except Exception as error:
        print(error)
        messagebox.showerror("Error", error)
# Create Buttons
submit_btn = Button(delete_record, text="Delete", command=delete_passwr)

```

```

submit_btn.grid(row=26, column=0, pady=10)
    return_btn = Button(delete_record, text="Return", command=go_back)
    return_btn.grid(row=27, column=0, pady=10)
    delete_record.mainloop()
def exit_program():
    root.destroy()
    bye = Tk()
    bye.title("BYE!")
    bye.iconbitmap("ExtraSupportContent/florestechologylogo.ico")
    width = 500
    height = 530
    bye.geometry(str(width) + "x" + str(height))
    screen_width = bye.winfo_screenwidth()
    screen_height = bye.winfo_screenheight()
    x_coordinate = int((screen_width / 2) - (width / 2))
    y_coordinate = int((screen_height / 2) - (height / 2))
    bye.geometry(str(width) + "x" + str(height) + "+" + str(x_coordinate) + "+" +
str(y_coordinate))

    my_img =
ImageTk.PhotoImage(Image.open("ExtraSupportContent/FloresTechnologyLogo.png"))
    my_label = Label(bye, image=my_img)
    my_label.pack()
    quit_button = Button(bye, text="Exit", command=bye.destroy)
    quit_button.pack()
    bye.mainloop()
    os.system('notepad ExtraSupportContent/about.txt')
# Create Buttons
view_pass_btn = Button(root, text="View Passwords", font=("Times New Roman", 13),
command=view)
view_pass_btn.grid(row=3, column=1, padx=10, pady=10)
add_pass_btn = Button(root, text="Add Passwords", font=("Times New Roman", 13),
command=add)
add_pass_btn.grid(row=3, column=6, padx=10, pady=10)
search_pass_btn = Button(root, text="Search for Passwords", font=("Times New Roman", 13),
command=search)
search_pass_btn.grid(row=5, column=1, padx=10, pady=10)
edit_pass_btn = Button(root, text="Edit Passwords", font=("Times New Roman", 13),
command=edit)

```

```

edit_pass_btn.grid(row=5, column=6, padx=10, pady=10)
delete_pass_btn = Button(root, text="Delete Passwords", font=("Times New Roman", 13),
command=delete)
delete_pass_btn.grid(row=7, column=3, padx=10, pady=10)
exit_btn = Button(root, text="Exit", bg="black", fg="red", font=("Times New Roman", 13),
command=exit_program)
exit_btn.grid(row=10, column=3, pady=10)
root.mainloop()

```

password_keeper_def_func.py

```

import sqlite3
from cryptography.fernet import Fernet
import password_keeper
from tkinter import * # Used for the graphical user interface (gui)
from tkinter import messagebox
def show_all(key, username):
    f = Fernet(key)
    conn = sqlite3.connect("registryUsers.db")
    c = conn.cursor()
    c.execute("SELECT rowid, * FROM " + username)
    items = c.fetchall()
    root = Tk()
    root.title("Your Passwords:")
    root.iconbitmap("ExtraSupportContent/florestechtechnologylogo.ico")
    width = 600
    height = 800
    root.geometry(str(width) + "x" + str(height))
    screen_width = root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()
    x_coordinate = int((screen_width / 2) - (width / 2))
    y_coordinate = int((screen_height / 2) - (height / 2))
    root.geometry(str(width) + "x" + str(height) + "+" + str(x_coordinate) + "+" +
str(y_coordinate))
    main_frame = Frame(root)
    main_frame.pack(fill=BOTH, expand=1)
    my_canvas = Canvas(main_frame)
    my_canvas.pack(side=LEFT, fill=BOTH, expand=1)
    my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL, command=my_canvas.yview)
    my_scrollbar.pack(side=RIGHT, fill=Y)
    my_canvas.configure(yscrollcommand=my_scrollbar.set)
    my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))

```

```

second_frame = Frame(my_canvas)
my_canvas.create_window((0, 0), window=second_frame, anchor="nw")
sb = Scrollbar(root)
sb.pack(side=RIGHT, fill=Y)
mylist = Listbox(root, yscrollcommand=sb.set, width=width, height=(height-20))
for item in items:
    decrypted_user = f.decrypt(item[1])
    original_user = decrypted_user.decode()
    decrypted_pass = f.decrypt(item[2])
    original_pass = decrypted_pass.decode()
    decrypted_email = f.decrypt(item[3])
    original_email = decrypted_email.decode()
    decrypted_info = f.decrypt(item[4])
    original_info = decrypted_info.decode()
    mylist.insert(END, "~~~~~")
    mylist.insert(END, "Id: " + str(item[0]))
    mylist.insert(END, "User/Website: " + original_user)
    mylist.insert(END, "Password: " + original_pass)
    mylist.insert(END, "Email: " + original_email)
    mylist.insert(END, "Info: " + original_info)
    mylist.insert(END, "~~~~~")
    mylist.insert(END, "")
def go_back():
    root.destroy()
    password_keeper.mainPasswordKeeper(username)
Button(root, text="Return", command=go_back).pack()

mylist.pack(side=LEFT)
sb.config(command=mylist.yview)
conn.commit()
conn.close()
def add_one(username, user, password, email, info):
    conn = sqlite3.connect("registryUsers.db")
    c = conn.cursor()
    c.execute("INSERT INTO " + username + " VALUES (?, ?, ?, ?)", (user, password, email,
info))
    conn.commit()
    conn.close()

def delete_one(username, id_var):
    conn = sqlite3.connect("registryUsers.db")
    c = conn.cursor()
    c.execute("DELETE from " + username + " WHERE rowid = " + id_var)
    conn.commit()
    conn.close()

```



```

def name_lookup(key, username, name):
    f = Fernet(key)
    conn = sqlite3.connect("registryUsers.db")
    c = conn.cursor()
    c.execute("SELECT rowid, * FROM " + username)
    items = c.fetchall()
    toplevel = Toplevel()
    toplevel.title("Search...")
    toplevel.iconbitmap("ExtraSupportContent/florestechtechnologylogo.ico")
    width = 600
    height = 800
    screen_width = toplevel.winfo_screenwidth()
    screen_height = toplevel.winfo_screenheight()
    x_coordinate = int((screen_width / 2) - (width / 2))
    y_coordinate = int((screen_height / 2) - (height / 2))
    toplevel.geometry(str(width) + "x" + str(height))
    toplevel.geometry(str(width) + "x" + str(height) + "+" + str(x_coordinate) + "+" +
str(y_coordinate))
    sb = Scrollbar(toplevel)
    sb.pack(side=RIGHT, fill=Y)
    mylist = Listbox(toplevel, yscrollcommand=sb.set, width=width, height=(height - 20))
    for item in items:
        decrypted_name = f.decrypt(item[1]) # The rowid is in the '0' position, so name is in '1'
        original_name = decrypted_name.decode()
        if original_name == name:
            decrypted_pass = f.decrypt(item[2])
            original_pass = decrypted_pass.decode()
            decrypted_email = f.decrypt(item[3])
            original_email = decrypted_email.decode()
            decrypted_info = f.decrypt(item[4])
            original_info = decrypted_info.decode()
            mylist.insert(END, "~~~~~")
            mylist.insert(END, "Id: " + str(item[0]))
            mylist.insert(END, "User/Website: " + original_name)
            mylist.insert(END, "Password: " + original_pass)
            mylist.insert(END, "Email: " + original_email)
            mylist.insert(END, "Info: " + original_info)
            mylist.insert(END, "~~~~~")
            mylist.insert(END, "")
    def go_back():
        toplevel.destroy()
    Button(toplevel, text="Return", command=go_back).pack()
    mylist.pack(side=LEFT)
    sb.config(command=mylist.yview)
    toplevel.mainloop()

```

```

conn.commit()
conn.close()
def id_lookup(key, username, id):
    f = Fernet(key)
    conn = sqlite3.connect("registryUsers.db")
    c = conn.cursor()
    c.execute("SELECT rowid, * FROM " + username + " WHERE rowid = (?)", id)
    items = c.fetchall()
    toplvl = Toplevel()
    toplvl.title("Search...")
    toplvl.iconbitmap("ExtraSupportContent/florestechtechnologylogo.ico")
    width = 600
    height = 800
    screen_width = toplvl.winfo_screenwidth()
    screen_height = toplvl.winfo_screenheight()
    x_coordinate = int((screen_width / 2) - (width / 2))
    y_coordinate = int((screen_height / 2) - (height / 2))
    toplvl.geometry(str(width) + "x" + str(height))
    toplvl.geometry(str(width) + "x" + str(height) + "+" + str(x_coordinate) + "+" +
str(y_coordinate))

    try:
        for item in items:
            decrypted_name = f.decrypt(item[1])
            original_name = decrypted_name.decode()
            if int(id) == item[0]: # The rowid is in the '0' position
                decrypted_pass = f.decrypt(item[2])
                original_pass = decrypted_pass.decode()
                decrypted_email = f.decrypt(item[3])
                original_email = decrypted_email.decode()
                decrypted_info = f.decrypt(item[4])
                original_info = decrypted_info.decode()
                info = "Id: " + str(item[0]) + "\n" + "User/Website: " + original_name + "\n" +
"Password: " + original_pass \
                    + "\n" + "Email: " + original_email + "\n\n" + "Info: " + original_info
                Label(toplvl, text=info, font=("Times New Roman", 15)).pack()
    except:
        messagebox.showerror("Error", "There is no registry with that ID.")
    def go_back():
        toplvl.destroy()
    Button(toplvl, text="Return", font=("Times New Roman", 15), command=go_back).pack()
    toplvl.mainloop()
    conn.commit()
    conn.close()
def email_lookup(key, username, email):

```

```

f = Fernet(key)
conn = sqlite3.connect("registryUsers.db")

c = conn.cursor()
c.execute("SELECT rowid, * FROM " + username)
items = c.fetchall()
toplvl = Toplevel()
toplvl.title("Search...")
toplvl.iconbitmap("ExtraSupportContent/florestechologylogo.ico")
width = 600
height = 800
screen_width = toplevel.winfo_screenwidth()
screen_height = toplevel.winfo_screenheight()
x_coordinate = int((screen_width / 2) - (width / 2))
y_coordinate = int((screen_height / 2) - (height / 2))
toplvl.geometry(str(width) + "x" + str(height))
toplvl.geometry(str(width) + "x" + str(height) + "+" + str(x_coordinate) + "+" +
str(y_coordinate))
sb = Scrollbar(toplevel)
sb.pack(side=RIGHT, fill=Y)
mylist = Listbox(toplevel, yscrollcommand=sb.set, width=width, height=(height - 20))
for item in items:
    decrypted_name = f.decrypt(item[1]) # The rowid is in the '0' position, so name is in '1'
    original_name = decrypted_name.decode()
    decrypted_email = f.decrypt(item[3]) # The email is in position '3'
    original_email = decrypted_email.decode()
    if original_email == email:
        decrypted_pass = f.decrypt(item[2])
        original_pass = decrypted_pass.decode()
        decrypted_info = f.decrypt(item[4])
        original_info = decrypted_info.decode()
        mylist.insert(END, "~~~~~")
        mylist.insert(END, "Id: " + str(item[0]))
        mylist.insert(END, "User/Website: " + original_name)
        mylist.insert(END, "Password: " + original_pass)
        mylist.insert(END, "Email: " + original_email)
        mylist.insert(END, "Info: " + original_info)
        mylist.insert(END, "~~~~~")
        mylist.insert(END, "")
def go_back():
    toplevel.destroy()
Button(toplevel, text="Return", command=go_back).pack()
mylist.pack(side=LEFT)
sb.config(command=mylist.yview)
toplvl.mainloop()

```

```

conn.commit()
conn.close()
def update_one(username, user, password, email, info, id):
    conn = sqlite3.connect("registryUsers.db")
    c = conn.cursor()
    c.execute(" UPDATE " + username + " SET username = (?) WHERE rowid = (?)", (user, id))
    c.execute(" UPDATE " + username + " SET password = (?) WHERE rowid = (?)", (password,
id))
    c.execute(" UPDATE " + username + " SET email = (?) WHERE rowid = (?)", (email, id))
    c.execute(" UPDATE " + username + " SET info = (?) WHERE rowid = (?)", (info, id))
    conn.commit()
    conn.close()
def createTable(username):
    connection = sqlite3.connect("registryUsers.db")
    cursor = connection.cursor()
    cursor.execute("CREATE TABLE " + username + "(username text, password text, email text,
info text)")
    connection.commit()
    connection.close()

```

CryptographyDefFunc.py

```

from cryptography.fernet import Fernet
# Sets key for specific user
def set_key(username):
    # Every user has its own special key to encrypt and decrypt the info
    # Here with the username we get a key for that specific user
    key = Fernet.generate_key()
    distraction_name = "@@@!!!lave" + username + "k!!!@@" # If anyone is watching the
files I hope all the symbols will distract him/her
    file = open(distraction_name + ".key", "wb")
    file.write(key) # Write the key to the file
    file.close()

# Encrypts username and password
def encrypt_some(username, password):
    # Read key / Get the key from the file
    distraction_name = "@@@!!!lave" + username + "k!!!@@"
    file = open(distraction_name + ".key", "rb")
    key = file.read() # The key will be type 'bytes'
    file.close()
    # Encode username
    encoded_username = username.encode()
    # Encrypt username
    f = Fernet(key)

```

```

encrypted_username = f.encrypt(encoded_username)
    # Encode password
    encoded_password = password.encode()
    # Encrypt password
    f = Fernet(key)
    encrypted_password = f.encrypt(encoded_password)
    encrypted_list = (encrypted_username, encrypted_password)
    return encrypted_list

# Encrypts all info from the user
def encrypt_all(username, user, password, email, info):
    # The difference between this and encrypt is that this encrypts all info
    # Read key / Get the key from the file
    distraction_name = "@@@" + username + "k!!@"
    file = open(distraction_name + ".key", "rb")
    key = file.read() # The key will be type 'bytes'
    file.close()
    # Get key
    f = Fernet(key)

    # Encode username
    encoded_username = user.encode()
    # Encrypt username
    encrypted_username = f.encrypt(encoded_username)

    # Encode password
    encoded_password = password.encode()
    # Encrypt password
    encrypted_password = f.encrypt(encoded_password)

    # Encode email
    encoded_email = email.encode()
    # Encrypt email
    encrypted_email = f.encrypt(encoded_email)

    # Encode info
    encoded_info = info.encode()
    # Encrypt info
    encrypted_info = f.encrypt(encoded_info)

    encrypted_list = (encrypted_username, encrypted_password, encrypted_email, encrypted_info)
    return encrypted_list

# Gets the key from that specific user
def get_key(username):

```

```

# Read key / Get the key from the file
distraction_name = "@@@@!!llave" + username + "k!!!@@@"
file = open(distraction_name + ".key", "rb")
key = file.read() # The key will be type 'bytes'
file.close()
return key

# Decrypts the username
def decrypt(encrypted_username, key):
    # Decrypt the encrypted username
    f = Fernet(key)
    decrypted_username = f.decrypt(encrypted_username)
    # Decode the encoded username
    original_username = decrypted_username.decode()
    return original_username

# Decrypts the info in the specified tables
def decrypt_tables(tables, key):
    tables_original = []
    for table in tables:
        f = Fernet(key)
        decrypted = f.decrypt(table)
        original = decrypted.decode()
        tables_original.append(original)
    return tables_original

```