MAYA – THE SECRET VAULT

PROJECT REPORT

Project submitted in partial fulfillment of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

MD ABDUL MUSAVVIR (O161764)
TALLAPAKA MANASA (O162041)
NOSSAM VIGNATHA (O161530)
ERUKALA ANUSHA (O161005)

Under the supervision of

Ms. S. Nagamani

Dept. of Computer Science and Engineering ,
RGUKT Ongole Campus
March 2021-2022

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES
DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is certify that the project report entitled “**Maya The Secret Vault**" Submitted by **Md Abdul Musavvir**, Roll No : **O161764**, **Tallapaka Manasa** Roll No : **O162041**, **Nossam Vignatha**, Roll No : **O161530**, **Erukala Anusha** Roll No: **O161005**, to the Department of Computer Science and Engineering, Rajiv Gandhi University of Knowledge Technologies , Ongole Campus, during the academic year 2020-2021 is a partial fulfillment for the award of Undergraduate degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a bonafide record carried out by him under my supervision. The project has fulfilled all the requirements as per regulations of this institute and in my opinion reached the standard for submission.

Ms. K. Sandhya,	Ms. S. Nagamani
Head Of The Department,	Asst. Professor,
Dept of CSE,	Dept of CSE,
RGUKT Ongole Campus,	RGUKT Ongole Campus,
Ongole – 523225.	Ongole – 523225.

Date:

Place: Ongole

ACKNOWLEDGEMENT

We take this opportunity to express my profound gratitude and deep regards to my teacher **Ms. S. Nagamani** for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by them from time to time shall carry me a long way in the journey of life on which I am about to embark.

We have equally taken efforts in this project. We are also grateful to our Sirs and Madam for their consistent support and assistance.

Finally, last but by no means least; also to everyone in the university it was great sharing premises with all of you during the last five years.

Thanks for all your encouragement!

ABSTRACT

This project aims at developing a system which will hide the file where the user needs to first setup the master password and after creating the password the users has four operations, it is shown in the screen and can select anyone operation by entering the desired number.

So this project aims at hiding the file, unhide the file, view the hidden files by thier index storage, delete and reset the vault, encrypt the file using AES encryption and finally exit from the vault if and if user wants to exit.

The password should be remembered and it should be very complex so that no one guess easily by social engineering. To do the project we have used the vault python package where it keeps the data or a file secret.

KEYWORDS: pyAesCrypt, OS.

CONTENTS

1. INTRODUCTION:

The project is based on the secret file storage vault in terminal with AES file encryption support. This project aims at developing a system which will hide the file where the user needs to first setup the master password and after creating the password the users has four operations, it is shown in the screen and can select anyone operation by entering the desired number. So this project aims at hiding the file, unhide the file, view the hidden files by thier index storage, delete and reset the vault, encrypt the file using AES encryption and finally exit from the vault if and if user wants to exit. The password should be remembered and it should be very complex so that no one guess easily by social engineering. To do the project we have used the vault python package where it keeps the data or a file secret.

AES[1] : The Advanced Encryption Standard (AES) is a symmetric block cipher chosen by the U.S. government to protect classified information. AES is implemented in software and hardware throughout the world to encrypt sensitive data. It is essential for government computer security, cybersecurity and electronic data protection.

The National Institute of Standards and Technology (NIST) started development of AES in 1997 when it announced the need for an alternative to the Data Encryption Standard (DES), which was starting to become vulnerable to brute-force attacks.

1.1 Project Aim and Objective :-

The objective of our project is to hide the file which are giving by the user and encrypt the file with AES encryption algorithm which ofcourse the users will.

1.2 Existing System

The existing system is the hashicorp vault where the Vault is a tool that enables you to keep your secrets secured and encrypted. It can also do other things like encrypt data for your applications (so they don't need a special library to do so) and generate temporary secrets and certificates for users and services.

1.3 Proposed System

The proposed approach aims to use knowledge of python and create the system without the UI and with the help of the inbuilt modules like cryptography(fernet, hashes) it comes with the simple terminal usage where the files can be hidden and unhide anywhere. And the master password is different from the screen lock of the system.

1.4 Scope of Investigation

We will be implementing the hiding of the file, unhide the file, view the hidden files by thier index storage, delete and reset the vault, encrypt the file using AES encryption and finally exit from the vault if and if user wants to exit.

2. THEORITICAL BASES AND LITERATURE REVIEW

2.1 Theoretical Background of the Problem:

We have been using the hashicorp vault with more learning ofcourse its a software where the software cannnot be able to understand by the first usage so the basic learning is needed and the proposed system does not need the excessive learning.

2.2 Related Research of the Problem:

Now the hashicorp vault is being used vigorously and Hashicorp vault is a phenomenal tool to store and dynamically create secrets, provide encryption for your applications on the fly and even manage certificates. There aren’t really any other tools quite as powerful and versatile. But it is very complex in nature and to undestand where the key value version 2 is stored but bit confusing.

2.3 Our Solution to the Problem:

The system is mainly focuses on the ease of user point of view where the complexity is being reduced and the encryption is being used is the AES.

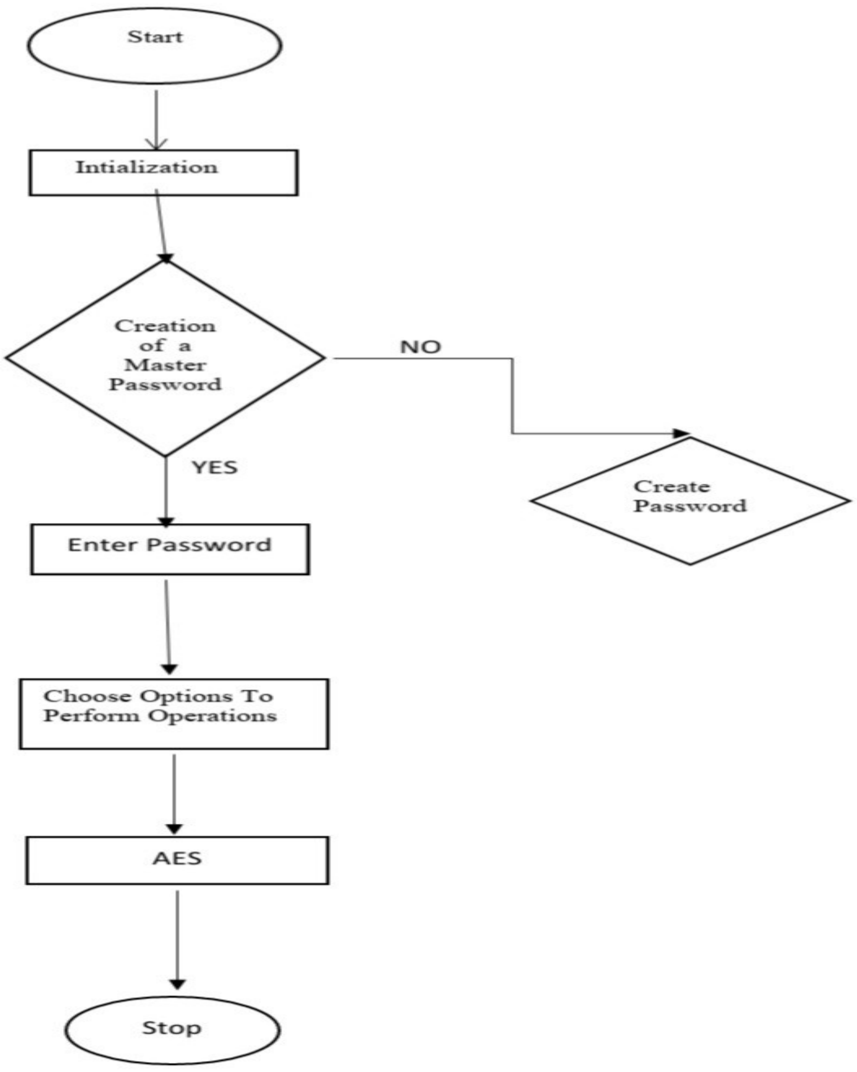
3. METHODOLOGY

3.1 Data Collection:

With the domain level knowledge of Python we will create the system where it can be easily done by using the python modules where it provides more libraries and provide more flexibility.

3.2 Solution Structure:

3.2.1 Algorithm design


Start
Intialization
Creation of a Master Password
NO
Create Password
YES
Enter Password
Choose Options To Perform Operations
AES
Stop

3.2.2 Language:

Our project uses Python 3.8. Python has many readily available and proven open sourcelibraries. All our required libraries support Python 3.8.

3.2.3 Tools used:

PyAesCrypt[2] : pyAesCrypt is a Python 3 file-encryption module and script that uses AES256-CBC to encrypt/decrypt files and binary streams.

Base64 : The Base64 encoding is used to convert bytes that have binary or text data into ASCII characters. Encoding prevents the data from getting corrupted when it is transferred or processed through a text-only system. In this article, we will discuss about Base64 encoding and decoding and its uses to encode and decode binary and text data.

3.3 Output:

The program with output is to hide the file or unhide the file along with encrypt the file.

4. SYSTEM ANALYSIS

4.1 SYSTEM MODULES:

There are 7 modules in this project.
They are -
1.OS
2.Base64
3.Shutil
4.pyAesCrypt
5.Sub process
6.Getpass
7.Cryptography

4.1.1.Os:

Python OS module provides the facility to establish the interaction between the user and the operating system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system.
The OS comes under Python's standard utility modules. This module offers a portable way of using operating system dependent functionality.The Python OS module lets us work with the files and directories.

4.1.2.Base64:

Base 64 encoding represents a common scheme for encoding binary into ASCII string format using radix 64. The base64 module is part of the standard library, which means it installs along with Python.

The base64 encode and decode functions both require a bytes like object. To get our string into bytes, we must encode it using Python's built in encode function.

4.1.3. Shutil:

Python shutil module provides the facility to perform the high-level file operation. It can operate with the file object and offers us the ability of copy and remove the files. It handles the low-level semantic such creating and closing file objects after performing all operations.

4.1.4. pyAesCrypt:

pyAesCrypt is a Python 3 file-encryption module and script that uses AES256-CBC to encrypt/decrypt files and binary streams. pyAesCrypt is compatible with the AES Crypt file format (version 2). It is Free Software, released under the Apache License, Version 2.0.

4.1.5.Subprocess:

Subprocess in Python is a module used to run new codes and applications by creating new processes. It lets you start new applications right from the Python program you are currently writing. So, if you want to run external programs from a git repository or codes from C or C++ programs, you can use subprocess in Python.

4.1.6.Getpass:

The getpass() function is used to prompt to users using the string prompt and reads the input from the user as Password. The input read defaults to “Password: ” is returned to the caller as a string

4.1.7.Cryptography:

Python supports a cryptography package that helps us encrypt and decrypt data. The fernet module of the cryptography package has inbuilt functions for the generation of the key, encryption of plaintext into ciphertext, and decryption of ciphertext into plaintext using the encrypt and decrypt methods respectively

4.2 System Requirements:

4.2.1 User Interface:

Every user may not be skilled at handling the interfaces. Hence the product that we developed used a simple and easy to use for user is via terminal.

4.2.2 Hardware Interface:

The minimum requirements that are required (RAM >=2GB) to interact with a terminal are well enough to support this product.

4.2.3 Software Interface:

This product is developed in ubuntu environment using Terminal. The toolboxes used to develop this product are python3 programming language.

4.3 Hardware and Software Requirements

4.3.1.Hardware Requirements:

Processor : Intel® CoreTM i3 2.53 GHz / Above or AMD processors of any
RAM : RAM 2 GB / Above

HDD : 120 GB / Above

4.3.2.Software Requirements:

Operating System : Linux and Unix
Developing Environment : Terminal

5.SYSTEM ENVIRONMENT

5.1 Technology:

A The dvanced Encryption Standard (AES) is a symmetric block cipher chosen by the U.S. government to protect classified information. AES[1] is implemented in software and hardware throughout the world to encrypt sensitive data. It is essential for government computer security, cybersecurity and electronic data protection.

The National Institute of Standards and Technology (NIST) started development of AES[1] in 1997 when it announced the need for an alternative to the Data Encryption Standard (DES), which was starting to become vulnerable to brute-force attacks.

5.2 Language:

Our project uses Python 3.6. Python has many readily available and proven open source libraries. All our required libraries support Python 3.6.

5.3 Tools used:

Python OS[3] module provides the facility to establish the interaction between the user and the operating system. Base 64 encoding represents a common scheme for encoding binary into ASCII string format using radix 64. Python shutil module provides the facility to perform the high-level file operation.pyAesCrypt is a Python 3 file-encryption module and script that uses AES256-CBC to encrypt/decrypt files and binary streams.Subprocess in Python is a module used to run new codes and applications by creating new processes.The getpass() function is used to prompt to users using the string prompt and reads the input from the user as Password. Python supports a cryptography package that helps us encrypt and decrypt data

6. SYSTEM DIAGRAM

UML[4] is a method for describing the system architecture in detail using the blueprint. UML[4] represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. UML is a very important part of developing objects oriented software and the software development process.

UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

UML Diagrams:

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships).

There are two types of diagrams, they are:

• Structural Diagrams

• Behavioral Diagrams

Structural Diagrams:

The UML‘s[4] four structural diagrams exist to visualize, specify, construct and document the static aspects of a system. I can View the static parts of a system using one of the following diagrams. Structural diagrams consist of Class Diagram, Object Diagram, Component Diagram, and Deployment Diagram.

Behavioral Diagrams:

The UML’s five behavioral diagrams are used to visualize, specify, construct, and document the dynamic aspects of a system. The UML’s behavioral diagrams are roughly organized around the major ways which can model the dynamics of a system. Behavioral diagrams consists of Use case Diagram, Sequence Diagram, Collaboration Diagram, State chart Diagram, Activity Diagram.

6.1 Use Case Diagram:

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system / subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

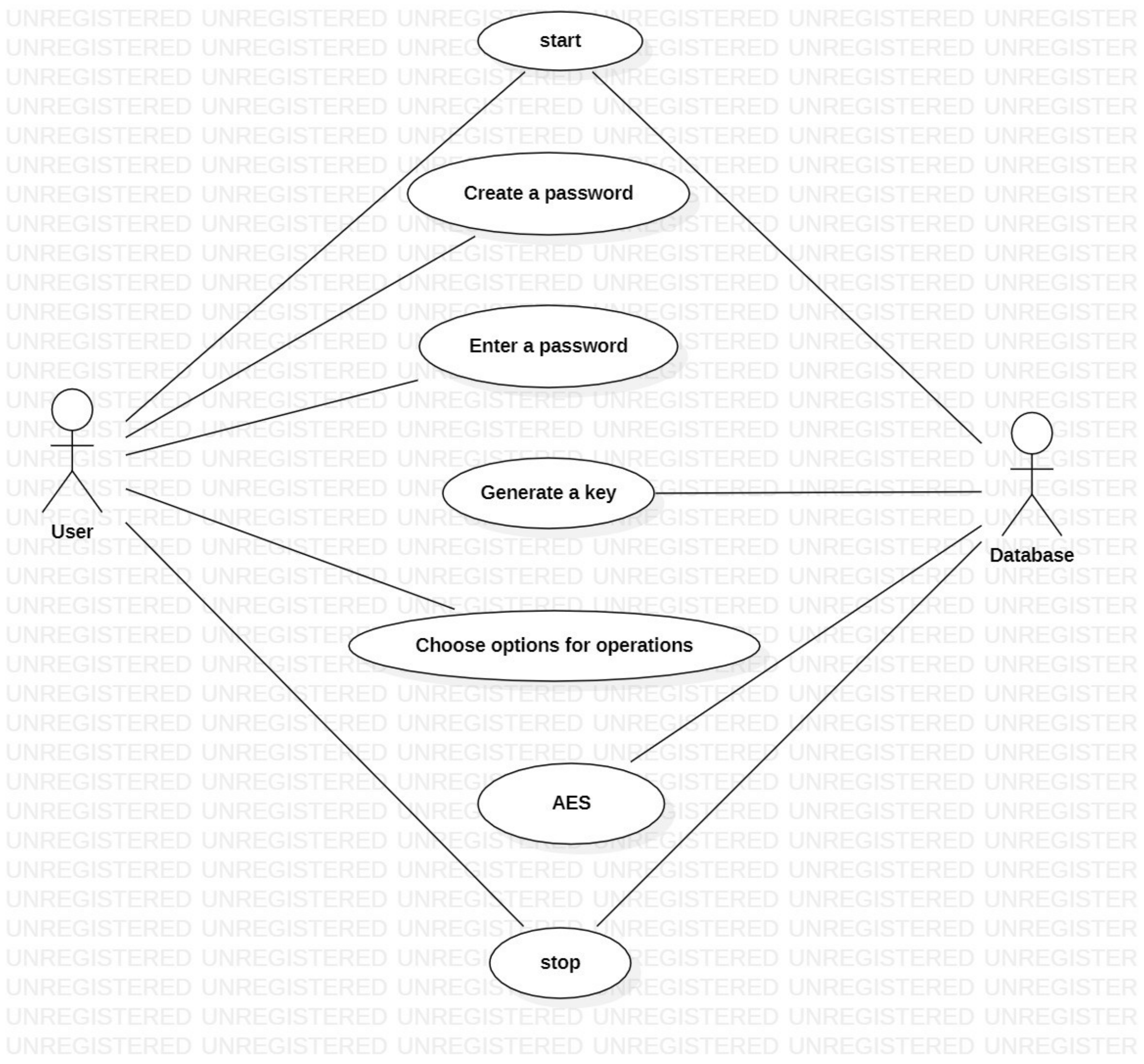


Figure 6.1 Use Case Diagram

6.2 Sequence Diagram:

A sequence diagram is a model that describe how groups of objects collobrate in some behaviour over time and capturing the behaviour of a single use case. It shows the interactions between the objects in terms of messages exchanged over time. Sequence diagrams are a popular dynamic modeling solution in UML[ix] because they specifically focus on lifelines, or the processes and objects that live simultaneously, and the messages exchanged between them to perform a function before the lifeline ends.

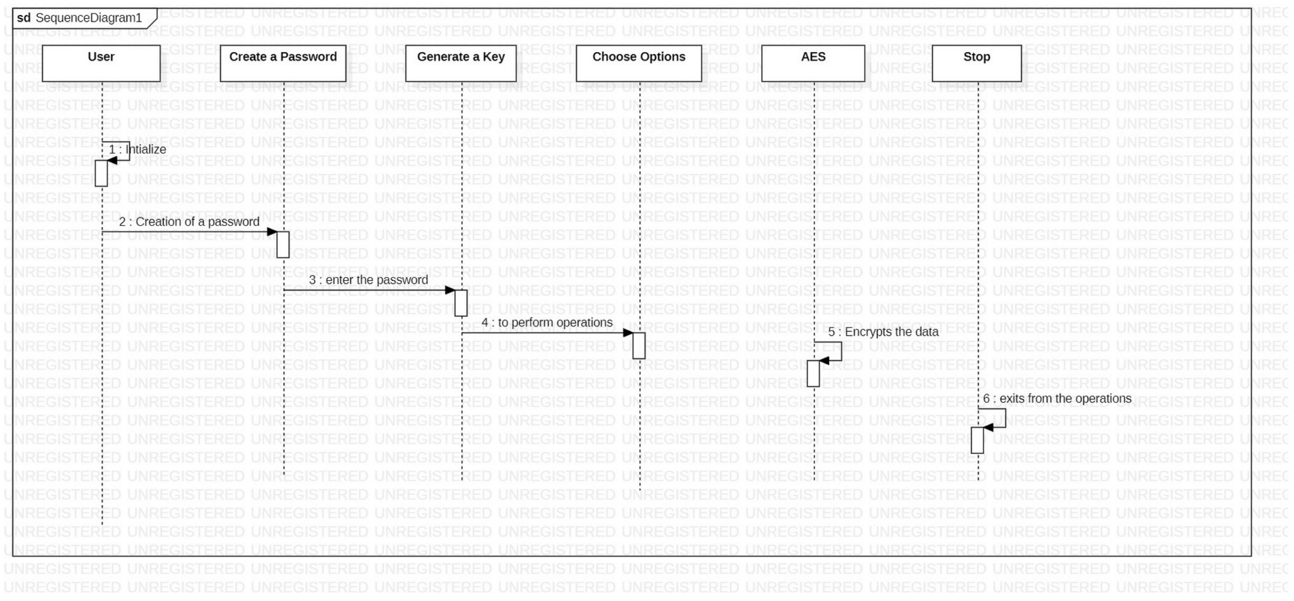


Figure 6.2 Sequence Diagram

6.3 Activity Diagram:

An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. We can depict both sequential processing and concurrent processing of activities using an activity diagram. They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system.

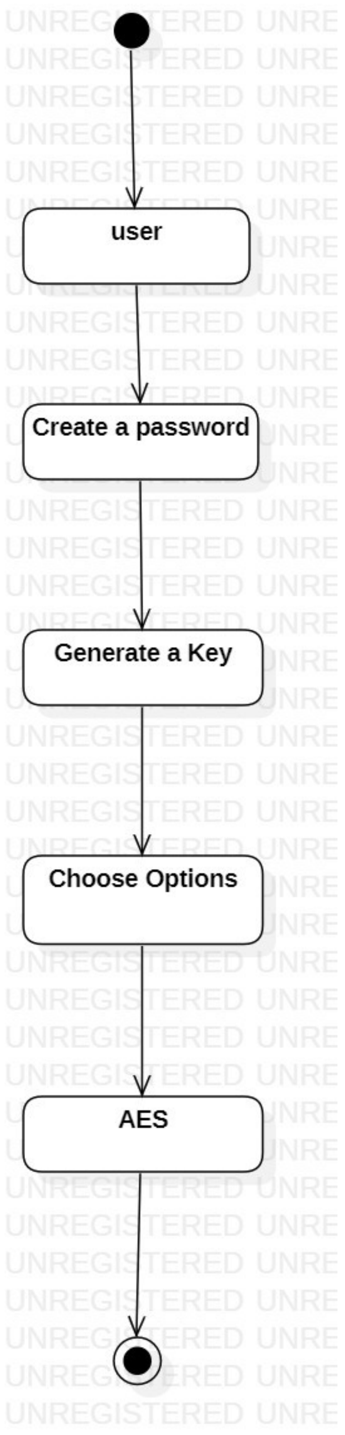


Figure 6.3 Activity Diagram

6.4 Class Diagram:

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

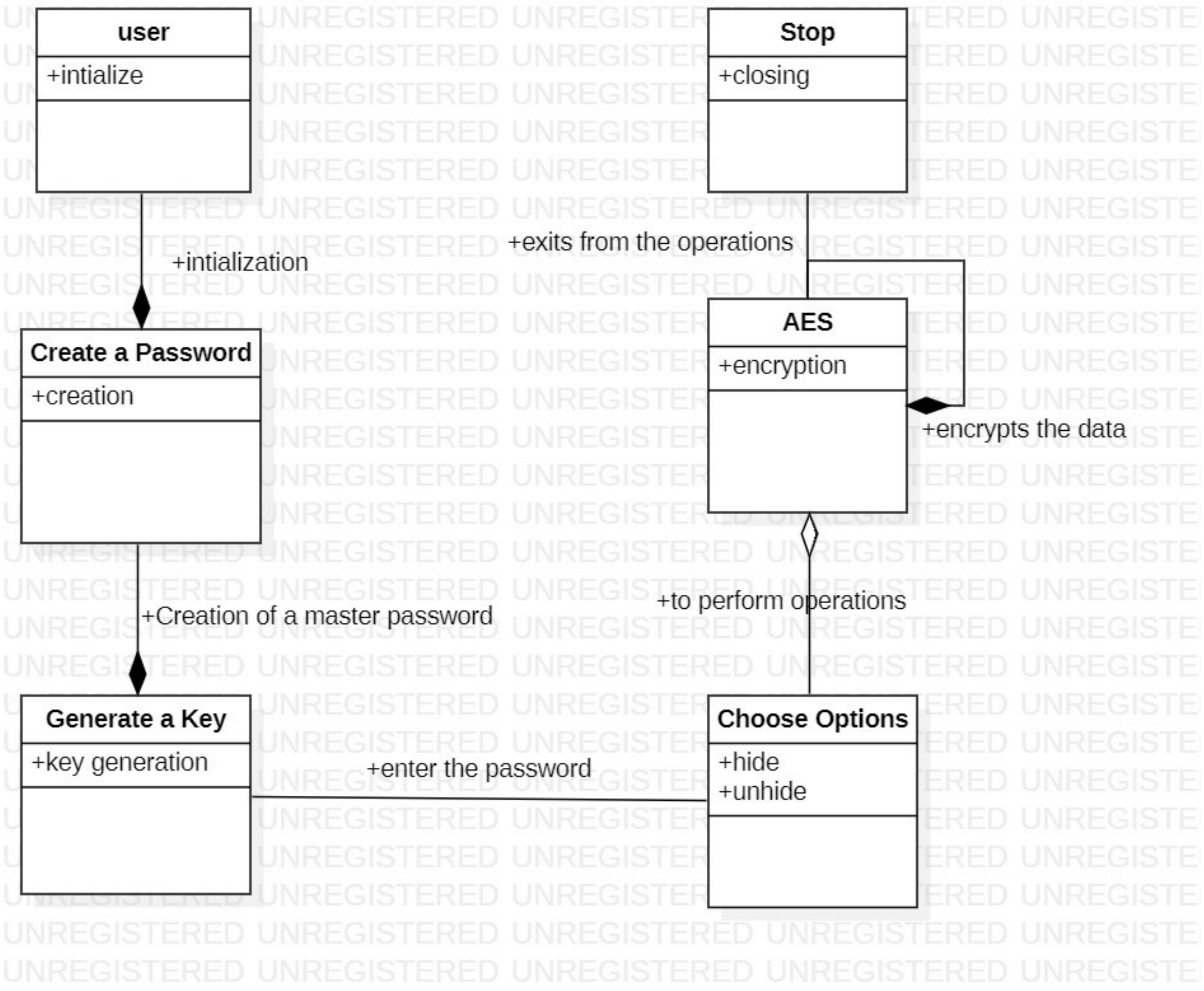


Figure 6.4 Class Diagram

6.5 Communication Diagram:

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node. It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

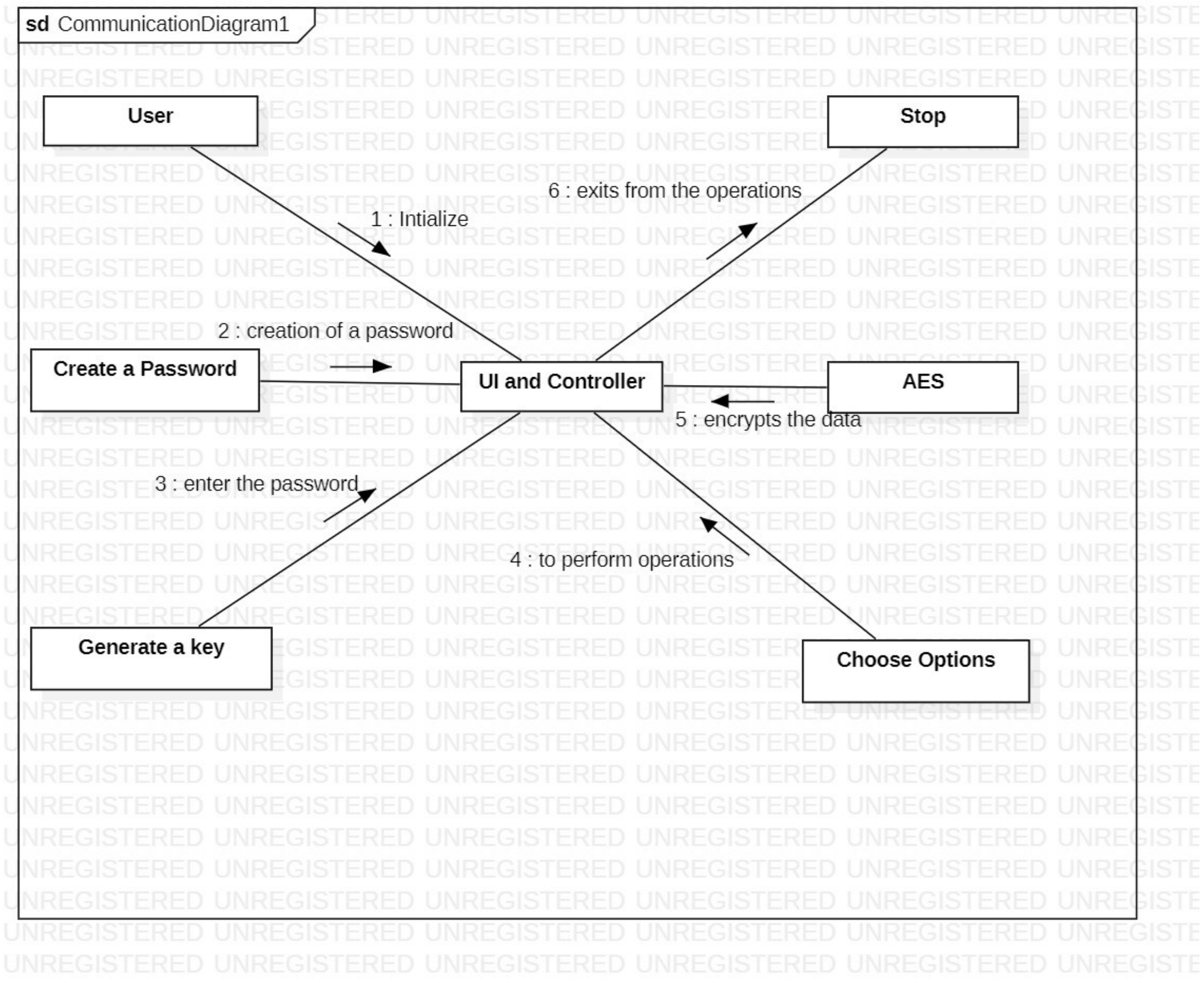


Figure 6.5 Communication Diagram

6.6 Component Diagram:

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node. It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

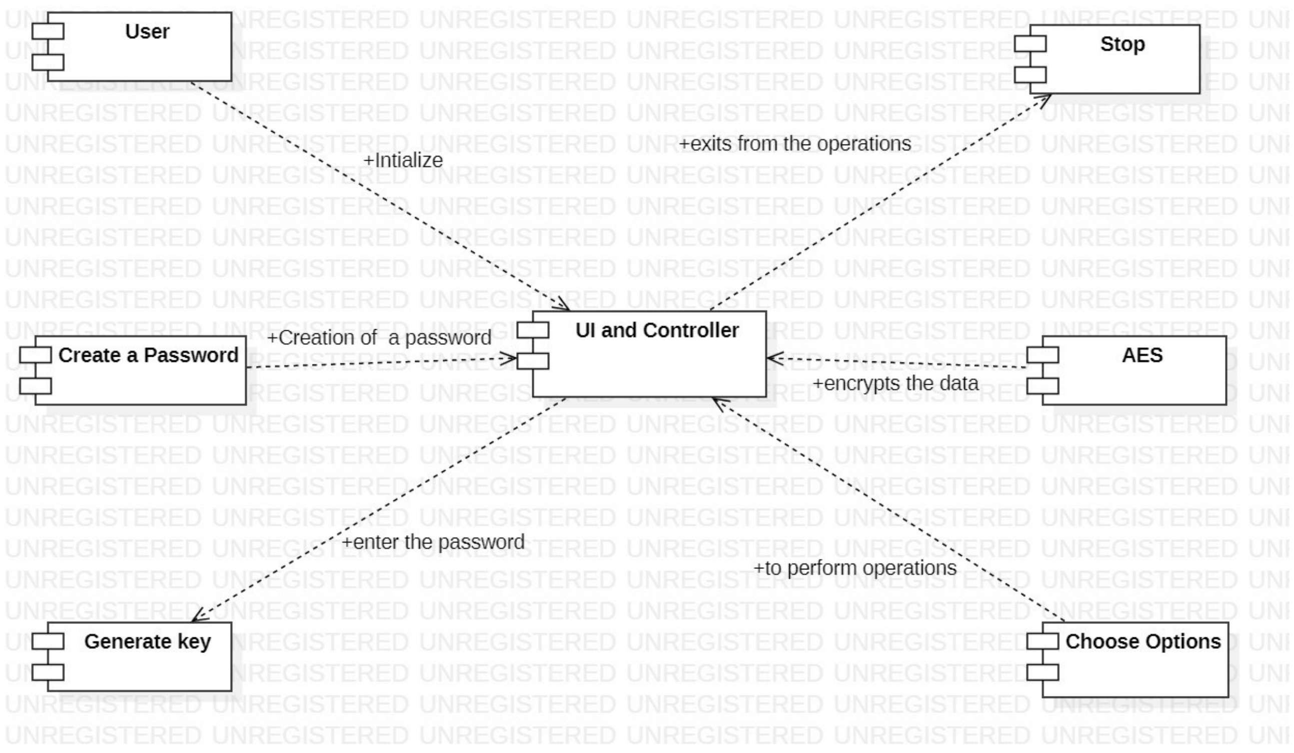


Figure 6.6 Component Diagram:

6.7 Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.It shows how data enters and leaves the system, what changes the information, and where data is stored.The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

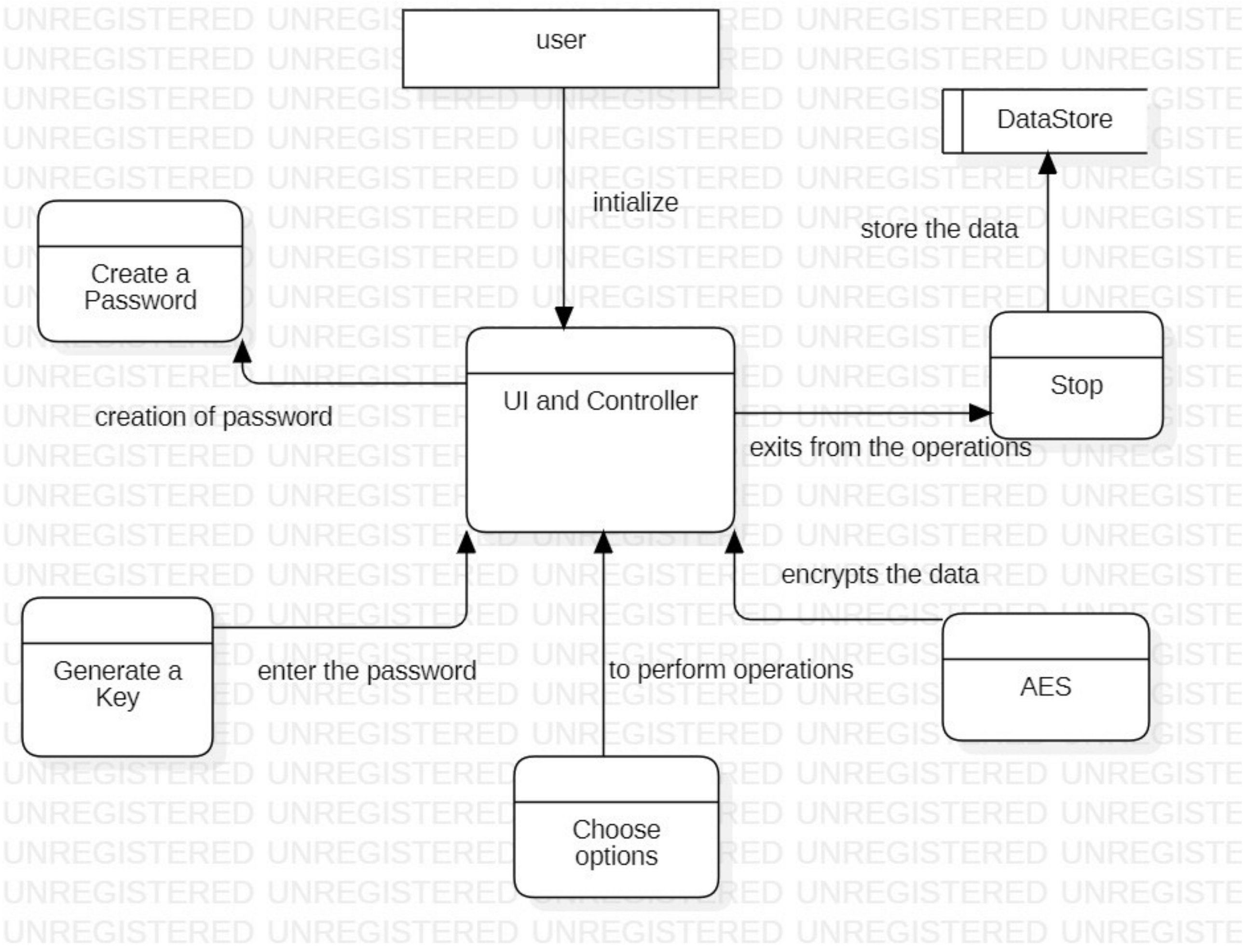


Figure 6.7 Data Flow Diagram

6.8 Deployment Diagram:

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships. It as certains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths.

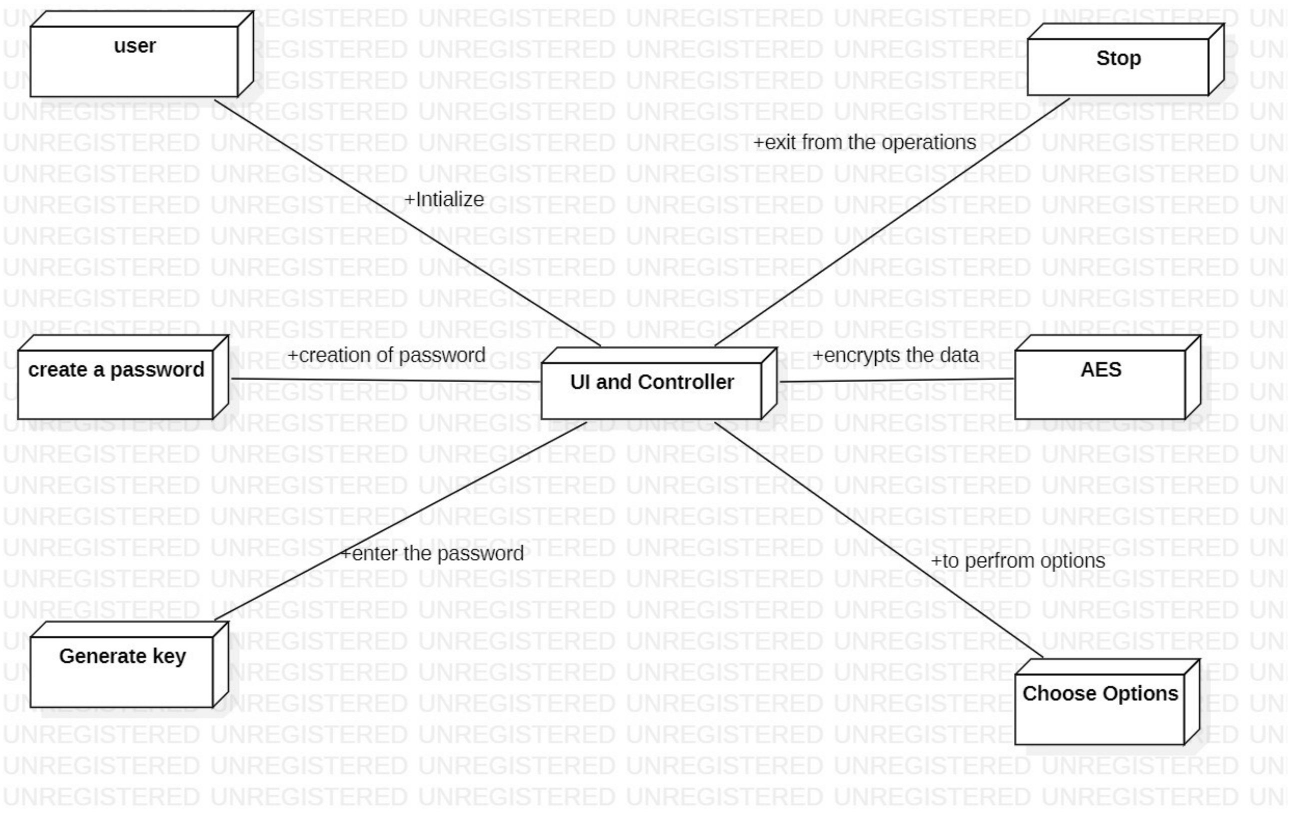


Figure 6.8 Deployment Diagram

6.9 Objective Diagram:

Object diagrams are dependent on the class diagram as they are derived from the class diagram. It represents an instance of a class diagram. The objects help in portraying a static view of an object-oriented system at a specific instant. Both the object and class diagram are similar to some extent; the only difference is that the class diagram provides an abstract view of a system. It helps in visualizing a particular functionality of a system.

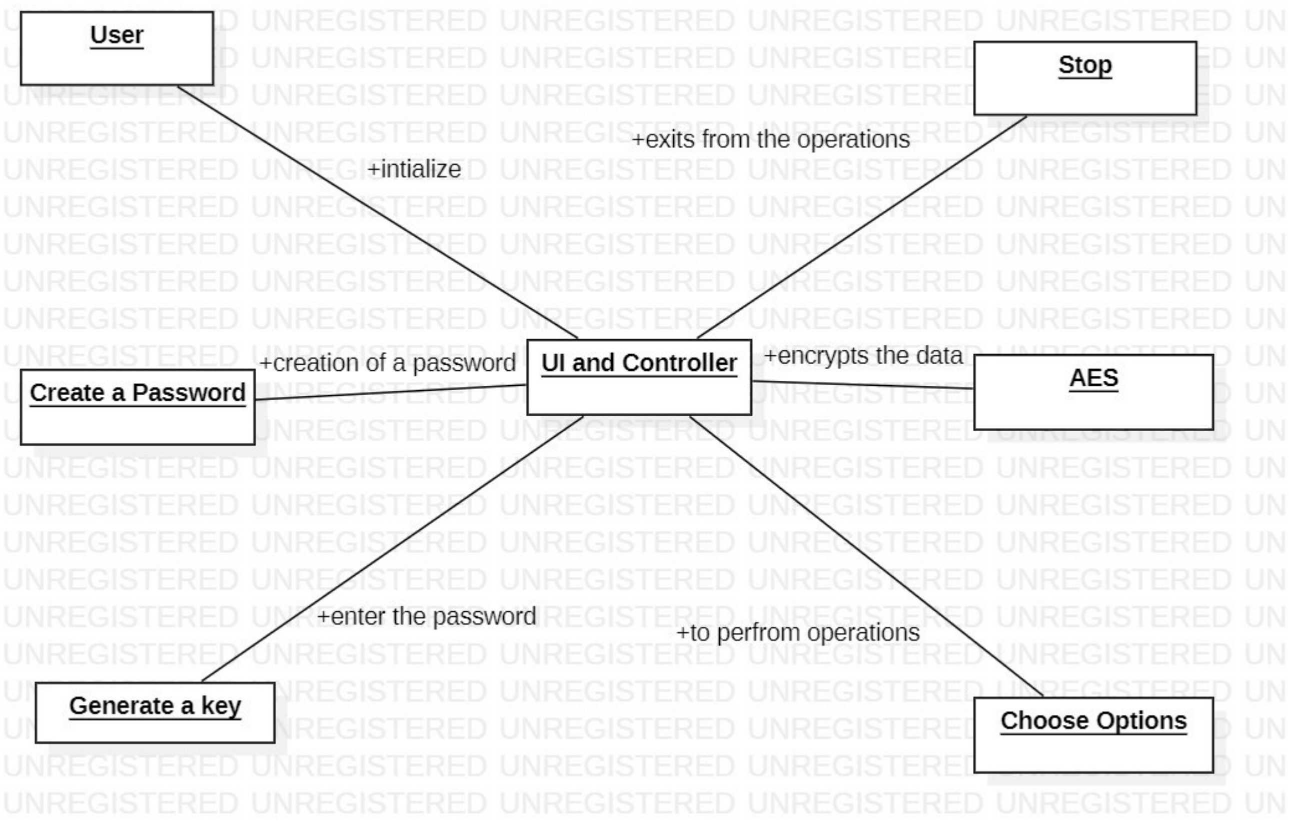


Figure 6.9 Objective Diagram

7. Implementation

```
import os
import base64
import shutil
import pyAesCrypt
from subprocess import call
from getpass import getpass
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC

class secret_vault:

    buffer_size = 64 * 1024

    def __init__(self, masterpwd):
        self.masterpwd = masterpwd

    def add_file(self, path, encrypt):
        if encrypt:
            filenameWithExt = os.path.basename(path) + '.aes'
            vaultpath = self.hid_dir + filenameWithExt
            pyAesCrypt.encryptFile(path, vaultpath, self.key.decode(), self.buffer_size)
        else:
            shutil.copy(path, self.hid_dir)

    def del_file(self, index):
        filenameWithExt = self.files[index]
        vaultpath = self.hid_dir + filenameWithExt
        if filenameWithExt.endswith('.aes'):
            filename = filenameWithExt[:-4]
            pyAesCrypt.decryptFile(vaultpath, filename, self.key.decode(),
            self.buffer_size)
            os.remove(vaultpath)
        else:
            shutil.copy(vaultpath, filenameWithExt)
            os.remove(vaultpath)

    def list_files(self):
        self.get_files()
        if not self.files:
            print("\nVault is empty!!!")
            return
        maxlen = max([len(x) for x in self.files])
        print('')
        print('-'*(maxlen+10))
        print("index\t|files")
        print('-'*(maxlen+10))
        for i, file in enumerate(self.files):
            print("{}\t|{}".format(i, file))
            print('-'*(maxlen+10))

    def generate_key(self, salt=b"\xb9\x1f|}'S\xa1\x96\xeb\x154\x04\x88\xf3\xdf\x05",
    length=32):
      password = self.masterpwd.encode()

      kdf = PBKDF2HMAC(algorithm = hashes.SHA256(),
            length = length,
            salt = salt,
            iterations = 100000,
            backend = default_backend())

      self.key = base64.urlsafe_b64encode(kdf.derive(password))
```

```
    def get_files(self):
        self.files = os.listdir(self.hid_dir)

    def set_hid_dir(self):
        path = '~/.vault'
        hid_path = os.path.expanduser(path)
        self.hid_dir = hid_path + '/'

def main():
    print("Welcome to the secret vault!!!")
    path = os.path.expanduser('~/.vaultcfg')
    if os.path.exists(path):
        masterpwd = getpass("Enter your Master Password : ")
        vault = secret_vault(masterpwd)
        vault.generate_key()
        fernet = Fernet(vault.key)
        with open(path, 'rb') as f:
            actual_mpwd = f.read()
            try:
                fernet.decrypt(actual_mpwd)
                print('Welcome Back')
            except:
                print("Wrong Master Password!")
                exit()
    else:
        masterpwd = getpass("Create a Master Password : ")
        vault = secret_vault(masterpwd)
        vault.generate_key()
        fernet = Fernet(vault.key)
        enc_mpwd = fernet.encrypt(masterpwd.encode())
        with open(path, 'wb') as f:
            f.write(enc_mpwd)
            vault.set_hid_dir()
        try:
                os.makedirs(vault.hid_dir[:-1])
        except FileExistsError:
                pass

        if os.name == 'nt':
                call(["attrib", "+H", vault.hid_dir[:-1]])
                call(["attrib", "+H", path])

        print("Welcome")

vault.set_hid_dir()

choice = 0
while choice != 4:
        print("\nEnter 1 to hide a file\nEnter 2 to unhide a file\nEnter 3 to view hidden files\
        nEnter 4 to Exit\nEnter 5 to Reset the vault and delete all of its contents\n")
        try:
                choice = int(input("Enter your choice : "))
        except:
                print("\nUnknown value!")
                continue

        if choice == 1:
                print("\nTip : Drag and Drop the file")
                filepath = input("Enter the path of the file to hide : ")
                filepath = filepath.replace('\\', '')
                if filepath.endswith(' '):
                        filepath = filepath[:-1]
                if os.path.exists(filepath):
                        if os.path.isfile(filepath):
                                while True:
```

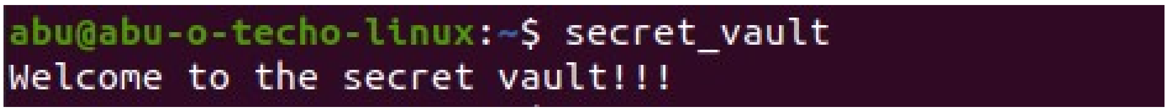
```
                                enc_or_not = input("Do you want to encrypt the file?
                                (Y or N) : ")
                                if enc_or_not == 'y' or enc_or_not == 'Y':
                                        print('\nAdding file to the vault...')
                                        vault.add_file(filepath, 1)
                                        print("\nFile successfully added to the vault")
                                        print("You can now delete the original file if
                                        you want")
                                        break
                                elif enc_or_not == 'n' or enc_or_not == 'N':
                                        print('\nAdding file to the vault...')
                                        vault.add_file(filepath, 0)
                                        print("\nFile successfully added to the vault")
                                        print("You can now delete the original file if
                                        you want")
                                        break
                                else:
                                        print("Type Y or N")
                else:
                        print("\nGiven path is a directory and not a file!")
        else:
                print('\nFile does not exists!')


elif choice == 2:
        print('')
        try:
                file = int(input("Enter the index of the file from view hidden files : "))
                vault.del_file(file)
                print('\nFile unhided successfully')
                print('The file will be present in {}'.format(os.getcwd()))
        except:
                print("\nInvalid index!")
```

```
elif choice == 3:
    vault.list_files()


elif choice == 5:
    while True:
        confirm = input("\nDo you really want to delete and reset the vault?(Y
        or N) : ")
        if confirm == 'y' or confirm == 'Y':
            pwdCheck = getpass("\nEnter the password to confirm : ")
            reset = secret_vault(pwdCheck)
            reset.generate_key()
            resetFernet = Fernet(reset.key)
            path = os.path.expanduser('~/.vaultcfg')
            with open(path, 'rb') as f:
                actual_mpwd = f.read()
                try:
                    resetFernet.decrypt(actual_mpwd)
                    print('Removing and resetting all data...')
                except Exception as e:
                    print(e)
                    print("\nWrong Master Password!")
                    print("Closing program now...")
                    exit()
            os.remove(path)
            shutil.rmtree(vault.hid_dir[:-1])
            print('\nReset done. Thank You')
            exit()
        elif confirm == 'n' or confirm == 'N':
            print("\nHappy for that")
            break
        else:
            print("Type Y or N")
```

```
if __name__ == '__main__':
        main()
```

8. TESTING

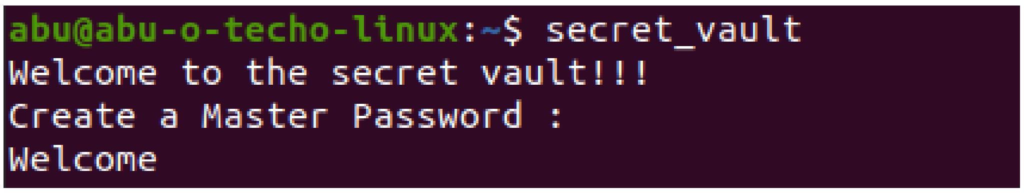
TestCase 8.1: Initialization.



```
abu@abu-o-techo-linux:~$ secret_vault
Welcome to the secret vault!!!
```

TestCase 8.1 : Initialization by the user or root user

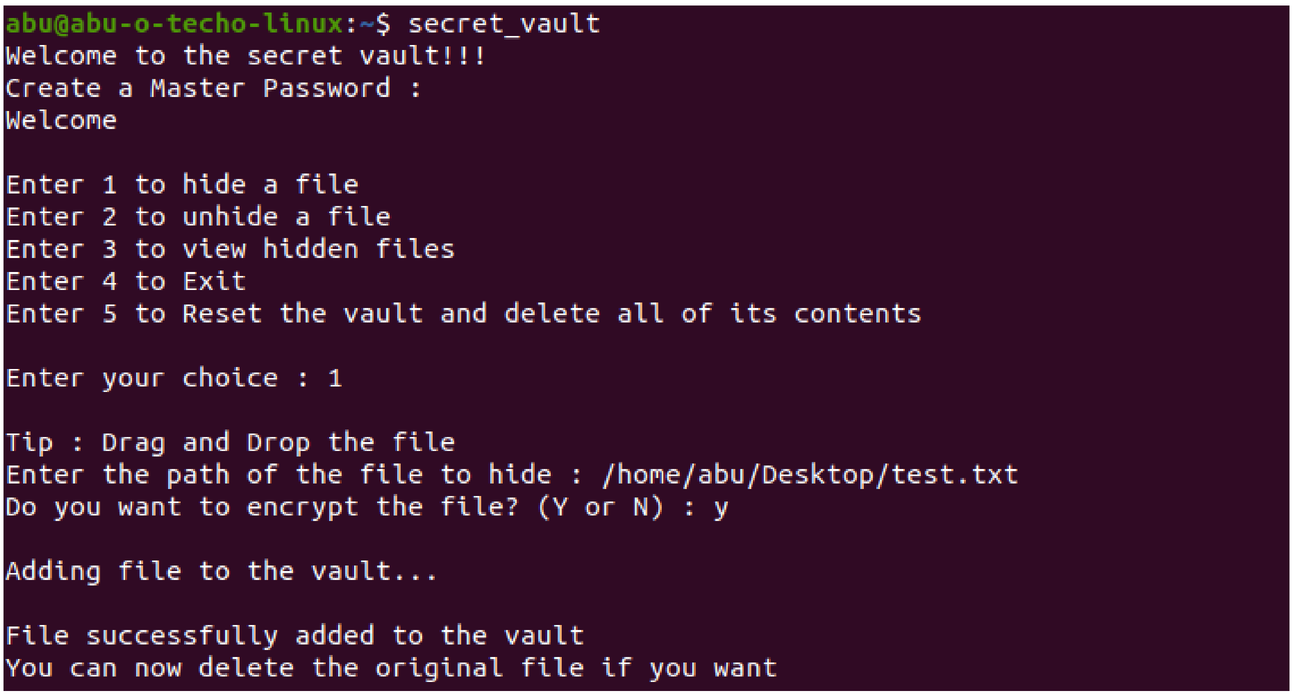
TestCase 8.2: Creating the master password



```
abu@abu-o-techo-linux:~$ secret_vault
Welcome to the secret vault!!!
Create a Master Password :
Welcome
```

TestCase 8.2 : Creating the master password

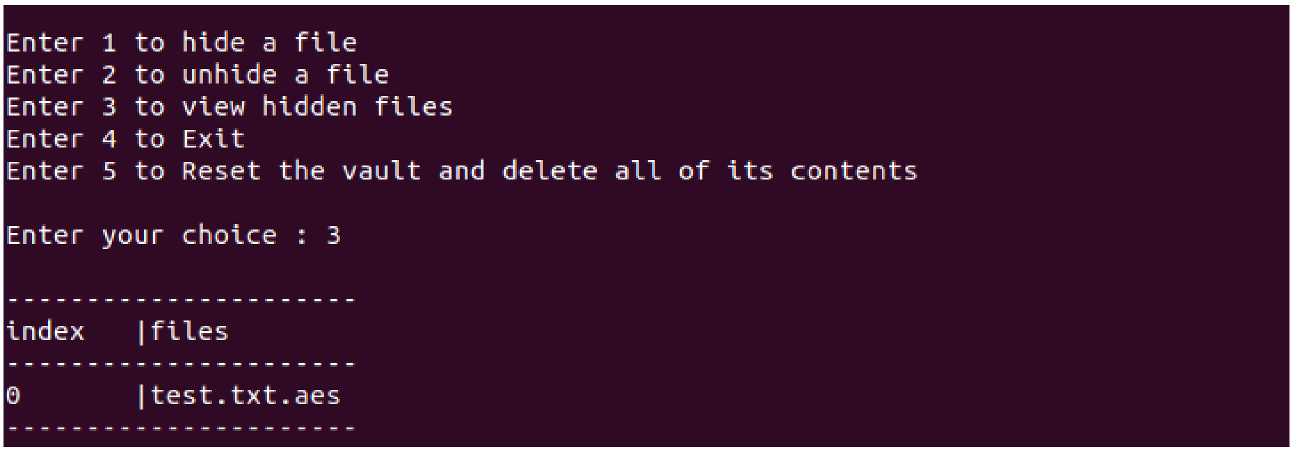
TestCase 8.3: Hiding the file where the user needs to locate the file address and paste or drag and drop it.



```
abu@abu-o-techo-linux:~$ secret_vault
Welcome to the secret vault!!!
Create a Master Password :
Welcome

Enter 1 to hide a file
Enter 2 to unhide a file
Enter 3 to view hidden files
Enter 4 to Exit
Enter 5 to Reset the vault and delete all of its contents

Enter your choice : 1

Tip : Drag and Drop the file
Enter the path of the file to hide : /home/abu/Desktop/test.txt
Do you want to encrypt the file? (Y or N) : y

Adding file to the vault...

File successfully added to the vault
You can now delete the original file if you want
```

TestCase 8.3 : Hiding the file

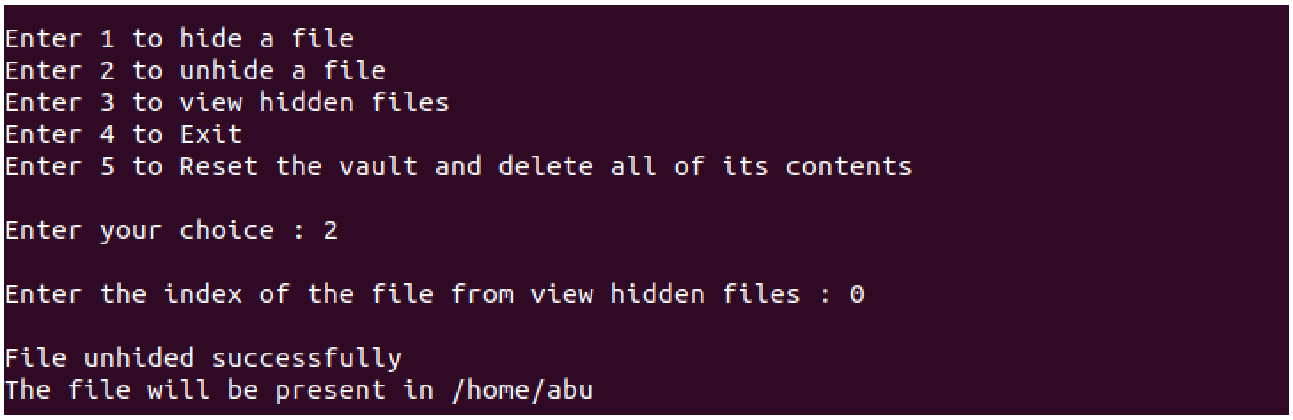
TestCase 8.4: Viewing the hidden files in the terminal for the user specific and it wont be saved in any logs .



```
Enter 1 to hide a file
Enter 2 to unhide a file
Enter 3 to view hidden files
Enter 4 to Exit
Enter 5 to Reset the vault and delete all of its contents

Enter your choice : 3

----------------------
index   |files
----------------------
0       |test.txt.aes
----------------------
```

TestCase 8.4 : View of the hidden files

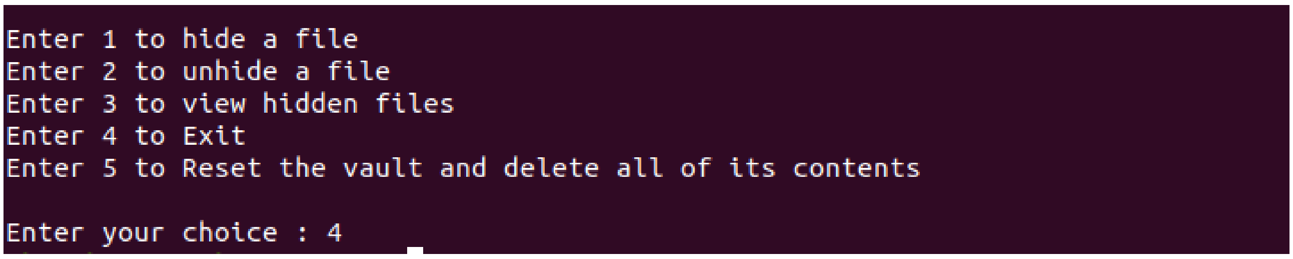
TestCase 8.5: Unhide the file/files from the vault and it will be saved in the same directory as the terminal opened for the intialization of the secret_vault.



```
Enter 1 to hide a file
Enter 2 to unhide a file
Enter 3 to view hidden files
Enter 4 to Exit
Enter 5 to Reset the vault and delete all of its contents

Enter your choice : 2

Enter the index of the file from view hidden files : 0

File unhided successfully
The file will be present in /home/abu
```

TestCase 8.5 : Unhide the file or files

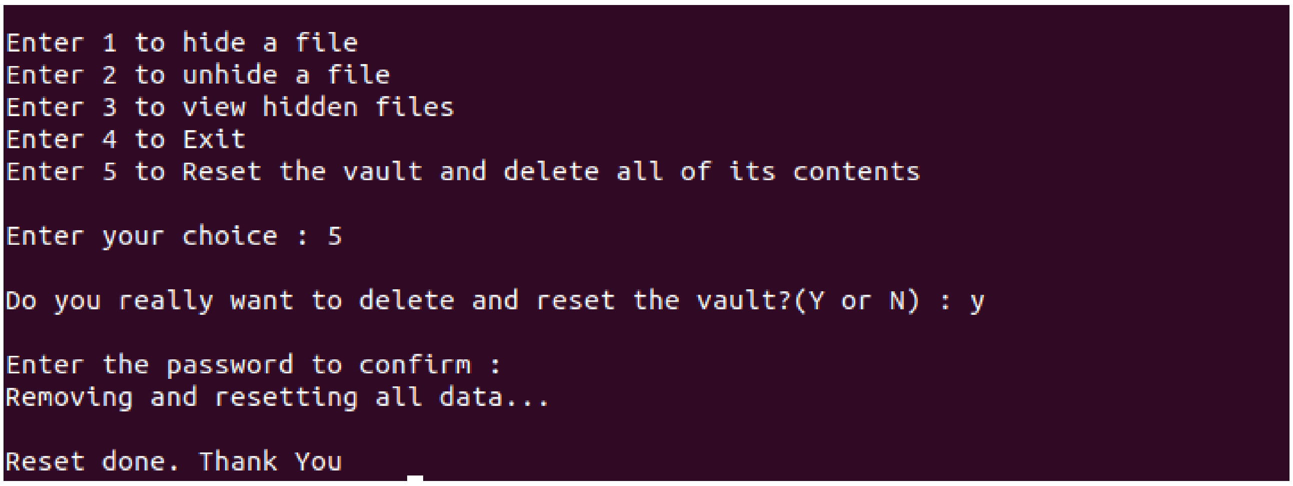
TestCase 8.6: Exit from the program



```
Enter 1 to hide a file
Enter 2 to unhide a file
Enter 3 to view hidden files
Enter 4 to Exit
Enter 5 to Reset the vault and delete all of its contents

Enter your choice : 4
```

TestCase 8.6 : Exit from the program

TestCase 8.7: Deleting and Resetting the vault by user wish and need master password to do this operation which was created by the user.



```
Enter 1 to hide a file
Enter 2 to unhide a file
Enter 3 to view hidden files
Enter 4 to Exit
Enter 5 to Reset the vault and delete all of its contents

Enter your choice : 5

Do you really want to delete and reset the vault?(Y or N) : y

Enter the password to confirm :
Removing and resetting all data...

Reset done. Thank You
```

TestCase 8.7 : Delete and Reset the files in the vault

9. CONCLUSION

This project conclude the system which will hide the file where the user needs to first setup the master password and after creating the password the users has four operations, it is shown in the screen and can select anyone operation by entering the desired number.

10. FUTURE WORK

In near future we will be adding the whole folder or multiple files encryption in one go unlike looping the every single file in a multiple process.

11. REFERENCES

- https://dock2learn.com/tech/how-to-store-your-secrets-in-vault-using-python/
- https://www.contino.io/insights/hashicorp-vault
- https://github.com/marketplace/actions/vault-secrets
- https://www.youtube.com/watch?v=VYfl-DpZ5wM

12. BIBLIOGRAPHY:

[1]	Advanced Encryption Standard
[2]	Python Advanced Encryption Standard Cryptography
[3]	Operating System
[4]	Unified Modeling Language