CADT
IDT វិទ្យាស្ថានបច្ចេកវិទ្យាឌីជីថល
Institute of Digital Technology

SECURE PASSWORD STORAGE

Department of Telecoms and Networking
Specialized in Cyber Security

Course: Cryptography
Term 1 | Year 3
Lecturer: Mr. Meas Sothearath
Written by: Ms.Chhay Lymean
Date of submission: 13/12/2025

Table of Content

I. Introduction

1. Overview of Project

This project implements a secure password management system using cryptographic techniques to protect user credentials. The system addresses the growing challenge of managing multiple online accounts securely while maintaining ease of use.

2. Problem Statement

Users typically have 70+ online accounts but struggle to remember strong, unique passwords for each. This leads to:

- Password reuse across multiple sites
- Weak password creation
- Insecure storage methods
- Increased vulnerability to credential theft

3. Solution

The system provides:

- Secure Storage: AES-256 encryption for password confidentiality
- User Authentication: Bcrypt hashing for login credentials
- Two-Step Verification: Additional security layer using personal questions
- Key Management: Option to generate or provide custom AES keys

4. Cryptographic Concept

- Password Encryption: Use AES-256 (EAX mode) to ensure confidential storage and integrity.
- User Authentication: Use Bcrypt hashing for secure credential verification
- Key Management: Use 32-byte AES keys per password for encryption control

5. Motivation

Current password management solutions often compromise between security and usability. This project demonstrates how cryptographic principles (AES, bcrypt) can be practically applied to create a secure yet user-friendly password manager that protects against common threats like brute-force attacks and credential leaks.

II. Implementation Components

1. Tools and Libraries

1.1. Tools

A lightweight, extensible code editor with built-in debugging, task management, and version control support, ideal for rapid development workflows.

Fig 01.
Visual Studio Code

Python is a high-level, easy-to-learn programming language used for building many kinds of software. It's popular because the code is simple, readable, and powerful.

Fig 02.
Python Language

Fig 03. GitHub

A cloud-based platform for version control and collaboration using Git, used to host, share, and manage project source code. I use it to store and publish my source code.

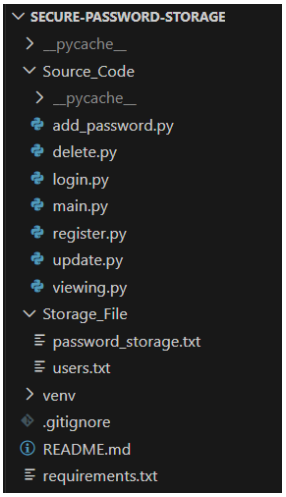
1.2. Libraries

Libraries are a collection of related modules. It contains bundles of code that can be used repeatedly in different programs. It makes Python programming simpler and more convenient for the programmer. I use nine libraries in this project, including

- **Re:** use to enforce rules for usernames and passwords by matching input against regular expression patterns. It ensures that users cannot create weak or invalid usernames/passwords.
- **OS:** In Python, the os library provides functions for interacting with the operating system, such as creating directories and files and managing the files or directories in many ways, and it makes it easy for us to find the path to any file.
- **Time:** used to handle time-related operations such as working timestamps, measuring time intervals, and manipulating time in various ways. We use it for calculating our project with clear time. For example, we use time.sleep() to skip the flow of the project for a moment at the time we set as seconds.
- **Getpass:** prompts the user for a password without echoing. It provides a secure way to handle the password prompts where programs interact with the users via the terminal. We use it to prevent shoulder surfing cyberattacks in real environments.
- **Bcrypt:** is a popular library in Python used for password hashing and secure password storage. It provides an implementation of the bcrypt hashing algorithm, which is a widely used and secure password hashing technique. I use it to encrypt users' information, including passwords and two-step verification information.
- **Msvcrt:** stands for Microsoft Visual C Runtime Library, is a built-in Python module that provides access to Microsoft Visual C Runtime functions. It is only available on Windows. I use it to mask input so that the first letter is visible and the rest are not when the user inputs their information.
- **Pyfiglet:** a Python library that lets you turn normal text into big ASCII art text. I use it for styling the welcome interface for the user.
- **PyCryptodome:** a third-party Python library that provides cryptographic functions, including AES encryption, decryption, and key generation to securely protect data.
- **Base64:** A built-in Python library that converts encrypted bytes into a text format for storage and decodes it back to bytes when decrypting. Without base64, storing AES-encrypted data as text in a password storage file could break or be unreadable.

2. File Handling

- **Secure Password Storage:** a folder that stores the whole project and also the name of the project.
- **Source Code:** Contains all source code files of the project, organized by functionality, including authentication, password management, and cryptographic operations.
- **.gitignore:** A configuration file that tells Git which files or folders to ignore and not track in the repository.
- **README.md:** A markdown file that provides an overview of the project. It contains the project description, installation instructions, usage guide, and other important information for users or developers.
- **Requirement.txt:** This file contains the library for use for this project.
- **Storage File:** Contains all persistent data files, including user credentials and encrypted passwords, separate from application source code.



3. Error Handling

- **Try :** I use try to cover the block of code that need to execute.
- **Exception:** I use exceptions to prevent errors so that the project still works as normal even if it is an error. Exceptions also help us to know what kind of error and alert the message. There are five types of except that we use including :
  - **ValueError as e:** Raised when a function receives a value of the correct type but an inappropriate value. Example: when user leaves username blank and when user enters a menu option that is not 0, 1, or 2
  - **keyboardInterrupt:** Raised when the user interrupts program execution using Ctrl+C. I use it to ensure the program exits gracefully rather than showing a Python error trace.
  - **PermissionError:** Raised when a file or resource cannot be accessed due to insufficient permissions.
  - **FileNotFound:** Raised when a file operation fails because the file does not exist.
  - **ImportError:** Raised when Python fails to import a module. Example: the user forgot to install the library, so the system will send the message instead of keeping the user in error mode.

4. Data Structure

- **Users.txt:** one line for one person. I use ":" to separate each piece of data for the user, including ID, username, password, favorite color, favorite animal, and zodiac sign. All user data are stored in hash format using bcrypt to ensure security.
  - Format

user_id:username:password[hashed]:fav_color[hashed]:fav_animal[hashed]:zodiac_sign[hashed]

- **Password_storage.txt:** one line for one account. It means in one line, there are user_ids, which can connect users.txt with password_storage.txt, passwords, and descriptions in hashed format for one user. So I use ":" to split each piece of data for one password and use "|" to separate passwords and descriptions so that one user can store many passwords in the system.
  - Format

user_id:password_stored(1)[hashed]:description(1)[hashed]|password_stored(2)[hashed]:description(2)[hashed]

5. Function In Project

- **Main module**
  - Register(): This function is the main function to execute the register feature.
  - login(): This function is the main function to execute the login feature for the user.
  - add_password(): This function is the main function to execute the add password feature so that users can store any password they want
  - normal_view(): This function is created to display all passwords in an encrypted and masked format.
  - special_view(): This function is created to display all passwords in plain text format (readable).
  - delete_password(): This function is created to allow users to delete the password that they do not want to store anymore.
  - update_password(): This function is created to allow users to update their stored password.
- **Authentication module**
  - masked_input(): This is a custom input function for sensitive information like favorite color, favorite animal, and zodiac sign, where the first letter is visible but the rest is masked by *.
  - two_step_verification(): This function is like the second security layer that asks the user three questions to confirm the user's identity.
- **Cryptographic module**
  - encrypt_aes(): This function is created to operate AES encryption.
  - decrypt_aes(): This function is created to operate AES decryption.
  - hash_data(): This function is created to hash sensitive data like password, color, pet, and zodiac sign before saving to users.txt.

6. Algorithm/Cryptographic Method

6.1. AES (Advanced Encryption Standard)

A symmetric key encryption algorithm that uses the same key is used for both encryption and decryption.

- **Purpose:**

  My service is to remember password for user, so I use AES-256 to encrypt password and note for user because AES ensures confidentiality (Ensures that only someone with the correct key can read the data) and integrity & Authentication **(**ensures the data hasn't been modified by producing a tag for verification.)

- **Mode:**

  I use EAX mode because it prevents tampering and ensures confidentiality. It:

  - Generates a nonce (random value) to make each encryption unique.
  - Produces a tag to verify integrity.

6.2. Bcrypt

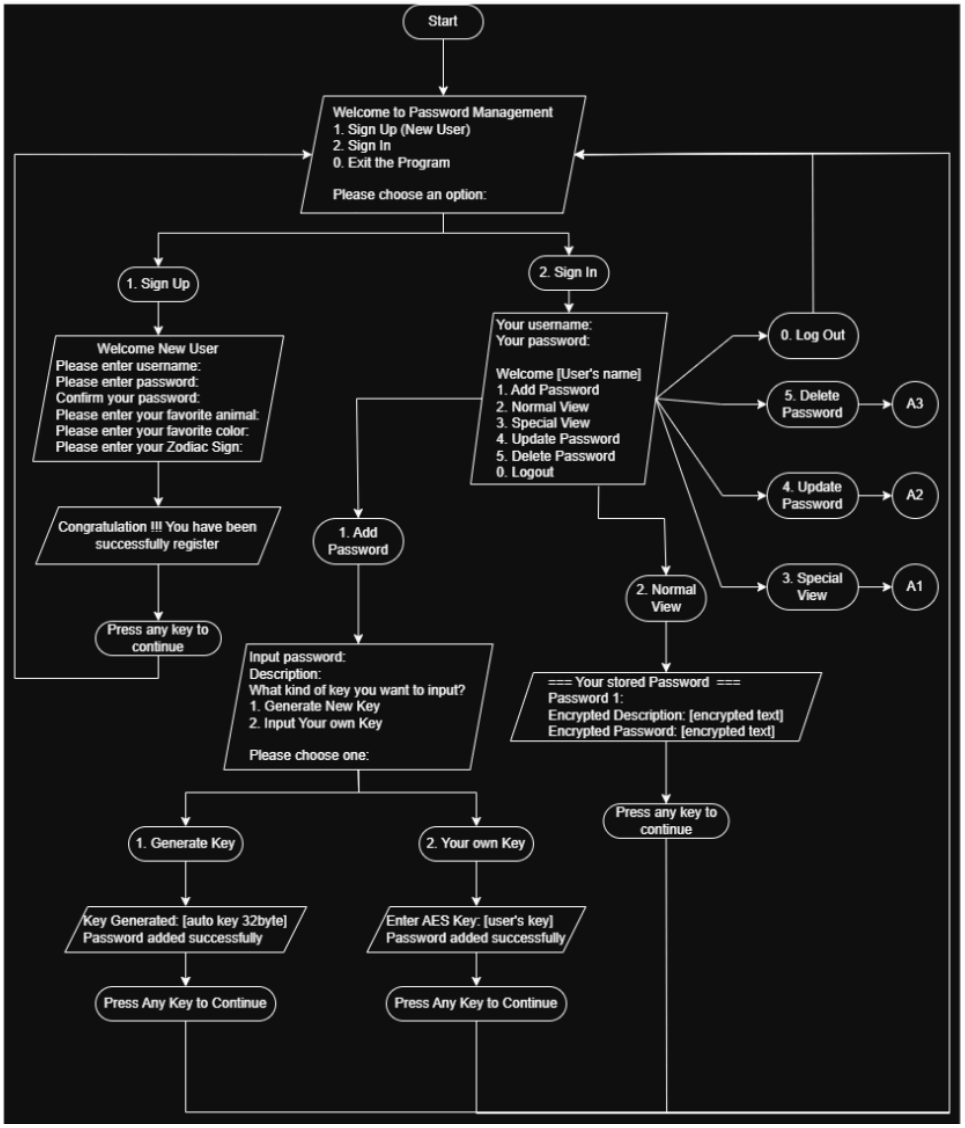
A strong, adaptive password-hashing function used to securely store passwords by converting them into unreadable hashes, making them resistant to brute-force attacks.

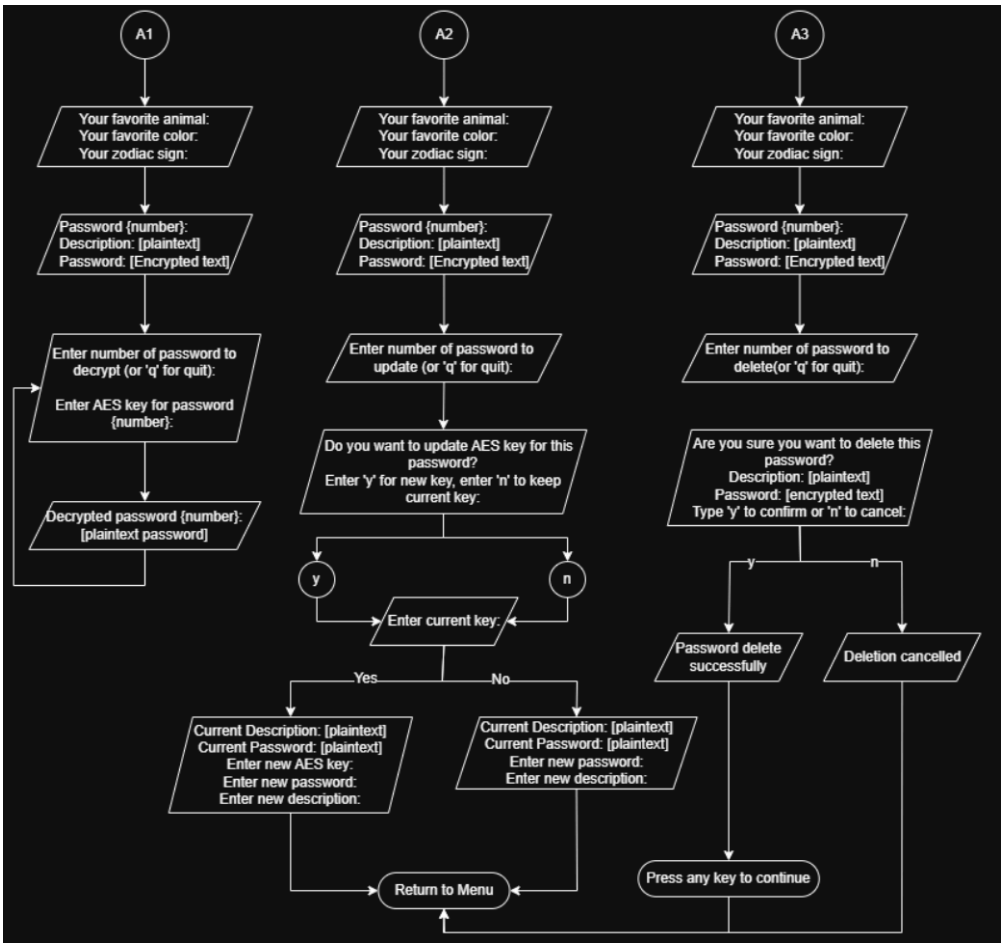
- **Purpose:**

  My system also stores user information when they create their account. So I use Bcrypt to securely store passwords and other sensitive data so that even if my database is leaked, an attacker cannot reverse or decrypt that information.

III. System Design/Architecture

1. System Architecture





2. System Structure Description

The project implements **Secure Password Storage** to offer users a simple and reliable way to manage their passwords. It supports a single user role with two main functions, including **Sign-Up** and **Sign-In**, designed to protect user confidentiality and data integrity. An exit option is also included to safely terminate the program.

- **Home Interface:** This interface pops up when the user starts running the program. This shows a user-friendly menu for users to choose from.
- **Sign-Up Option:** This is the first option for users who have not had an account. In this option, the system requires the user to fill in their information, including their username, password, favorite animal, favorite color, and zodiac sign.
  - **Username:** Their username must be 4-8 characters, unique (only one in user storage), and only include uppercase letters, lowercase letters, and numbers.
  - **Password:** Their password must be more than 8 characters and include uppercase letters, lowercase letters, numbers, and special characters. The password will be invisible when the user is typing to prevent shoulder surfing. It will be stored in hashing format to ensure security for users.
  - **Favorite Animal:** At this point, the user must start the word with a capital letter and then lowercase letters. When users are typing, the first letter is visible while the rest are replaced with stars (*). It will be stored in hashing format to ensure security for users.
  - **Favorite Color:** At this point, the user must start the word with a capital letter and then lowercase letters. When users are typing, the first letter is visible while

the rest are replaced with stars (*). It will be stored in hashing format to ensure security for users.

- **Zodiac Sign:** At this point, the system will display the list of all zodiac signs, and the user needs to fill in with one of them starting with a capital letter. When users are typing, the first letter is visible while the rest are replaced with stars (*). It will be stored in hashing format to ensure security for users.

After the user fills in all requirements, the system will display a congrats message and allow the user to press any key so that they can return to the home interface.

❖ **Sign-In Option:** This is the second option for users who already have an account in the system. In this option, the user will be required to verify authentication with their username and password, and then the system will display the menu for the user once they successfully log in. The menu includes

- **Add Password:** This option is for the user to store or write in their password. It allows the user to input the password that they want to store and then the description, which is like the note for the user to identify the password. The last stage is the user must choose between "Generate New Key" and "Input Your Own Key."
    - **Generate New Key:** will automatically generate the 32-byte key and display it for the user to keep for themself.
    - **Input Your Own Key:** will require the user to input whatever they want. But the user also needs to keep that key for themself.

  After the user adds a password successfully, it will allow the user to press any key to return back to the menu interface.

  *All the keys that the user inputs or generates are used to encrypt and decrypt the password only. For the description, I use a fixed key to encrypt and decrypt.
- **Normal View:** For this option, the user can view their own passwords that are stored in the system with the note they wrote (description), but all data will be displayed in a hashed, masked, and clear format that makes it easy for the user to know how many passwords they stored or the password order.
- **Special View:** This option is the advanced version of Normal View; the user can see their own password that is stored in the system with the note they wrote (description) in plaintext format by passing two layers of security.
    - **Two-Step Verification:** The system requires the user to answer a few questions, including favorite animal, favorite color, and zodiac sign. After successful verification, it will display the plaintext of the description and password order number.
    - **Password Revealing:** The user can choose the number of the password that they want to see. Then the system will require the user to input their key so that they can see the plaintext of the password in a readable format.
- **Update Password:** Not much different from Special View, the user needs to pass two-step verification and choose the password that they want to update by seeing the data display on screen. Then the system will ask if the user wants to change their key along with the password or keep the same key.

- **"y":** if the user inputs "y," it means "yes," and then the user needs to input the current key to prevent unauthorized users from modifying it. The user can update their password by inputting a new password, a new description, and a new key. After the process is complete, the user can return to the menu by pressing any key.
- **"n":** if the user inputs "n," it means "no," and then the user needs to input the current key to prevent unauthorized users from modifying. After that, the user can fill in their new password and description to complete the process and be ready to go back to the menu interface.

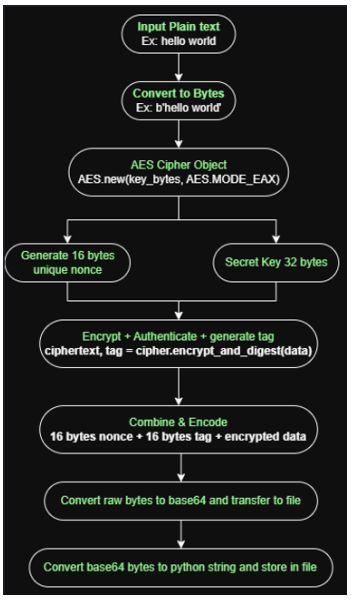
- **Delete Password:** Not much different from Update Password, the user needs to pass two-step verification and choose the password that they want to delete via the screen display. After that, it will ask the user again to confirm their decision if they really want to delete the password or cancel the process.
  - **"y":** if the user inputs "y," it means "yes," and then the password and the related data will be deleted, and the system will send the message that says "Password deleted successfully."
  - **"n":** if the user inputs "n," it means "no," and then the deleting process will be canceled and the data will stay safe from deletion. The system will display the message "Deletion cancelled."

3. Encryption/Decryption/Hashing Step

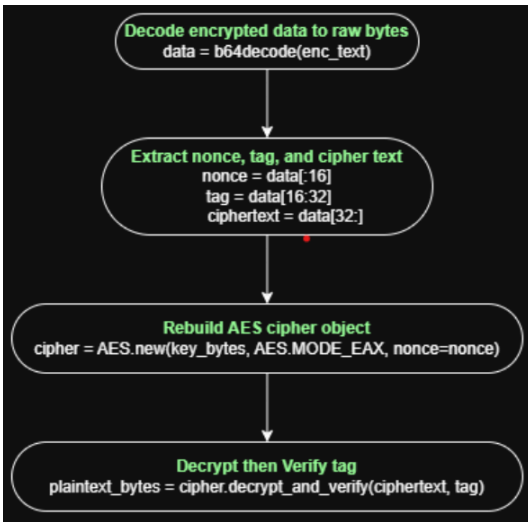
3.1. AES (Advanced Encryption Standard)

a/. Encryption

- First, the password or description is converted into bytes because AES cannot encrypt raw strings.
- Next, an AES cipher object is created using a secret key and AES in EAX mode, which automatically generates a unique 16-byte nonce.
- After that AES start to encrypt while producing a tag for integrity
- After encryption, the nonce, tag, and ciphertext are combined into a single sequence of raw bytes.
- Then it convert the raw bytes to base64 to ensure security for data in transit from program to file
- Finally, it decode the base64 to a Python string so that it can be stored in a file password_storage.txt.

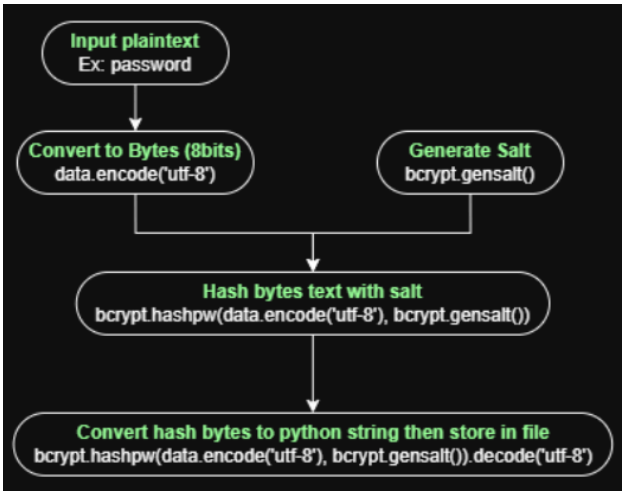


b/. Decryption



- First, the encrypted password is read from the file and decoded from Base64 into raw bytes.
- The system then extracts the first 16 bytes as the nonce, the next 16 bytes as the authentication tag, and the remaining bytes as the ciphertext.
- Using the same secret key, a new AES cipher object is created in EAX mode with the extracted nonce to rebuild the cipher object
- The ciphertext is decrypted and verified against the tag to ensure integrity.
- Once verification is correct, it will show the readable password to the user.

3.2. Bcrypt



- The user's input (e.g., password) is first converted from a Python string into bytes using UTF-8 encoding.
- bcrypt generates a random salt automatically using .gensalt(). The salt come with cost(work factor) which is the amount of round that bcrypt hash the value, by default it is 12.
- After that, the password bytes are hashed with the generated salt.
- The output of combination is bytes string of bcrypt hashing. It come with bcrypt algorithm which can identify the hash format ($2b$)
- Finally, it is converted to python string to store in users.txt file

IV. Usage Guideline

1. Installation

❖ Step 01: Install VSCode

- Open a web browser (Chrome, Firefox,...).
- Go to VSCode website: https://code.visualstudio.com/download
- Choose between window and Mac based on device you use
- Follow their guideline step-by-step, and then you will get VSCode on your device.

- ❖ Step 02: Install Python Language
  - Open a web browser (Chrome, Firefox,...).
  - Go to the official Python website: https://www.python.org/downloads
  - Go to the download option and click on the latest version, then it will be downloaded.
  - Or you can follow this guideline in detail for both window and MAC: https://code.visualstudio.com/docs/python/python-tutorial
  - In VSCode, you also need to install the Python language. Go to the Extension option, which is on the left sidebar, and search for Python; then you will see the Python that is owned by Microsoft. In the end, click install.
- ❖ Step 03: Install My Project
  - Open VSCode. On the top bar, click on Terminal, then New Terminal
  - Type: " git clone https://github.com/KCLP1995/Secure-Password-Storage.git "
  - Wait a second, and you will get my project on your device.
- ❖ Step 04: Execute the project
  - On the top bar, click on Terminal, then New Terminal
  - Type: " python .\Source_Code\main.py "

2. Dependencies

You need to install four required libraries, including **setuptools, pycryptodome, bcrypt, and pyfiglet,** so that you can use or run this project. I have 2 ways to install them; you can use either one of them.

- ❖ Option 01: you can install all libraries by install them one by one using
  - pip install [library_name]. Example: pip install setuptools
- ❖ Option 02: you can install all libraries in one time because I already create one file that contains all needed libraries. Using this command: pip install -r requirements.txt

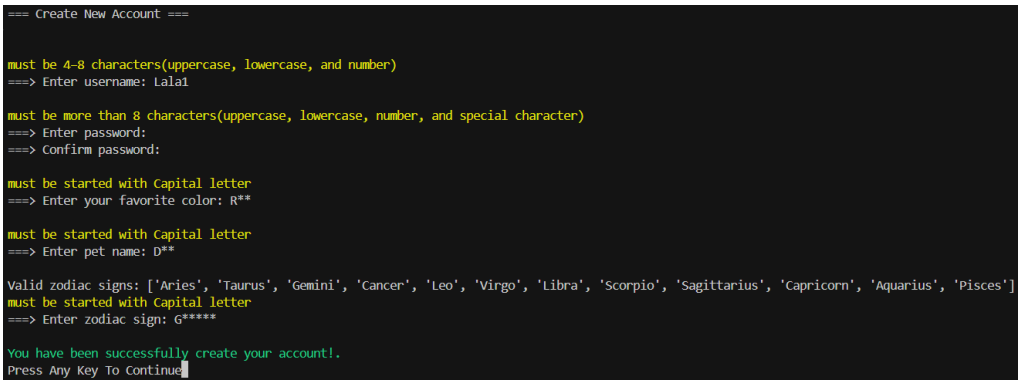
V. Results and Testing

1. Home Interface

Once user run this program, it will show the welcome interface and 3 options for use (sign-up, sign-in, and exit)

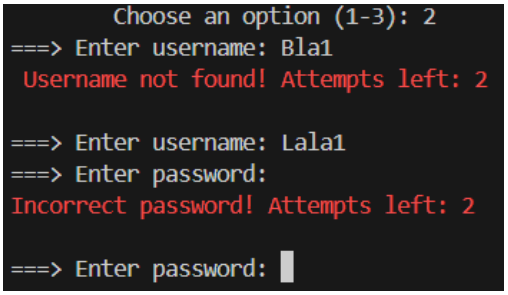
2. Sign-Up Interface

The user can fill in all the validated information that is required and displayed on screen. Once it is done, their account will be created successfully and stored in a file in a secure format. Each data has 3 attempts for the user to input. If the user runs out of attempts, the system will bring the user back to the home interface, and the account will not be created.

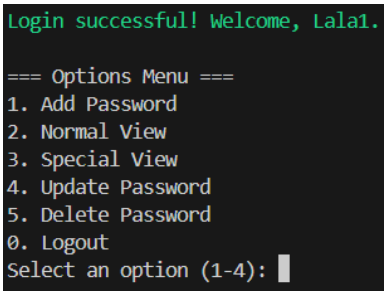


3. Sign-In Interface

The user has 3 attempts to login to their account and be ready to use the services in the system by choose number 1-5 and 0. Input outside the list will be error.



⇒

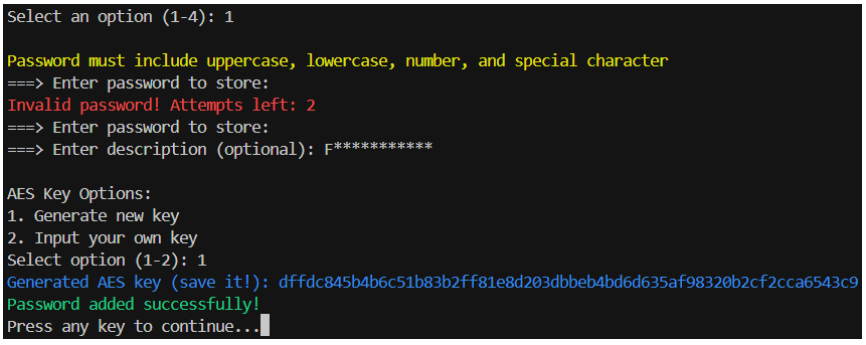


4. Services

4.1. Add Password

The user can input the password and note that they want to store within 3 attempts. If the user leaves the description blank, the system will store it as “No Description.” Most importantly, users need to use the key to secure the password they store by inputting “1” or “2.” For the key, the user needs to keep it safe so that they can use it to decrypt the password later. Recommendation: keep the key on a physical device or in physical material.

a/. Generate New key



b/. Input Your Own Key

```
Password must include uppercase, lowercase, number, and special character
===> Enter password to store:
Invalid password! Attempts left: 2
===> Enter password to store:
===> Enter description (optional):

AES Key Options:
1. Generate new key
2. Input your own key
Select option (1-2): 2
===> Enter AES key (your own key): l*****
Password added successfully!
Press any key to continue...
```

4.2. Normal View

After the user inputs “2” from the menu, the system will show all data that the user stored in encrypted and masked format to prevent unauthorized access and shoulder surfing.

```
=== Your Stored Passwords (Encrypted Preview) ===

Password 1:
  Encrypted Description: o******************************************************
  Encrypted Password: 0*************************************************

Password 2:
  Encrypted Description: D******************************************************
  Encrypted Password: L*************************************************


Press Any Key To Continue
```

4.3. Special View

a/. Two-Step Verify

The user needs to pass this verification with a few security questions before accessing the restricted area.

```
== Two-Step Verification Required ==
Enter your favorite color: R**
Enter your favorite pet: D**
Enter your zodiac sign: G*****
Verification successful!
```

b/. Showing Password

The user can choose one password from their password storage by inputting the order number of the password (password 1, password 2). Then they can use the key that they kept to show the password in readable format.

```
=== Your Stored Passwords ===

Password 1:
  Description: For facebook
  Password: 0*************************************************

Password 2:
  Description: No description
  Password: L*************************************************

Enter number of password to decrypt (or 'q' to quit): 1
Enter AES key:
Wrong AES key! Cannot decrypt this password.

Enter number of password to decrypt (or 'q' to quit): 2
Enter AES key:

Decrypted Password: Lala@123

Enter number of password to decrypt (or 'q' to quit):
```

4.4. Update Password

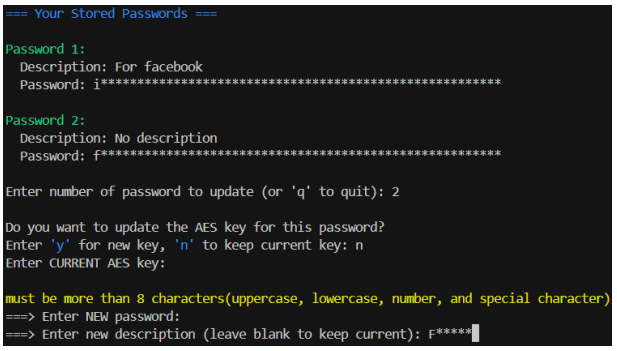
a/. Two-Step Verification

The user needs to pass the verification stage before they can update their storage.

b/. Updating

The user can choose one of the passwords from their storage to update by inputting the order number. The user can decide if they want to use the same key or a new key to store their new password.

- If the user inputs "y," they need to input the old key to ensure security and then input the key. After that they can write the new password and description. The description will stay the same if the user left it blank.
- If the user inputs "n," they still need to input the old key; then they can write the new password and description.



4.5. Delete Password

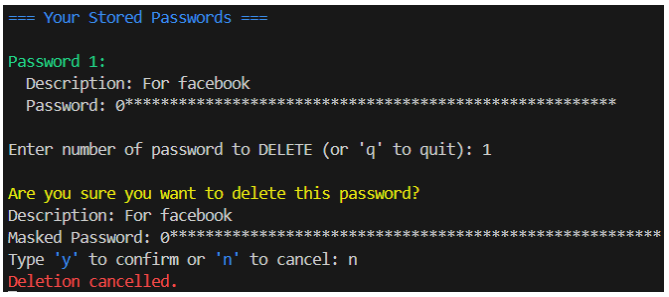
a/. Two-Step Verification

The user needs to pass the verification stage before they can delete their storage.

b/. Deleting

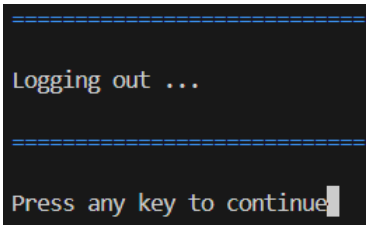
The user can choose one of the passwords from their storage to delete by inputting the order number. The system will ask the user again to confirm their decision.

- If the user inputs "y," the system will automatically delete that password from user's storage and tell user through the message
- If the user inputs "n," the system will cancel all the process of deleting, and the password will stay safe

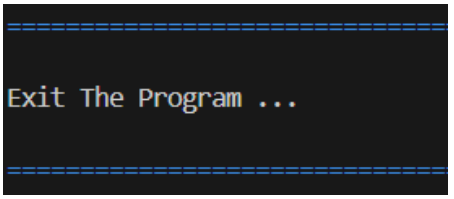


5. Exit Interface

5.1. Account Logout (option 0)



5.2. Exit Program



VI. Conclusion

This project successfully implements a secure password management system designed to protect user credentials against prevalent security threats. By integrating bcrypt hashing for authentication data and AES-256 encryption for stored passwords, the system provides robust defense mechanisms against brute-force attacks, shoulder surfing, and potential data breaches. The layered security approach combining strong encryption, two-step verification, and user-controlled key management ensures both data confidentiality and system reliability, offering users a practical and trustworthy solution for managing their digital identities securely.

VII. Future Work

This project lays the groundwork for several meaningful extensions that could enhance both its security and practical utility. Future development may include

- **Multi-Factor Authentication (MFA):** adding an additional layer of authentication, such as SMS or email OTPs, to enhance security and provide a more realistic user experience, including secure password recovery.
- **Real-World Implementation:** Developing the system into a web or mobile application would increase accessibility and provide a more polished, real-world user experience, making secure password management available to a broader audience.
- **Database Integration:** Transitioning from file-based storage to a secure database system would improve scalability, performance, and data integrity for larger user bases.

VIII. References

1. W3School: Learn Python Language
2. Geeks For Geeks: Learn about AES encryption and decryption
3. Doverunner: Learn about AES-256
4. Geeks For Geeks: Learn about Bcrypt
5. The Smart Dev Koder: Hashing Bcrypt in Python