Utilization of data security and privacy tools.
Anyone working with data has a responsibility to make sure that sensitive data is not at-risk from being stolen. We have seen data leaks all over, user details are exposed in such situations making it very risky for user details landing in the wrong hands.
We will see how to properly implement encryption on database systems and provide better security of such data. Passwords are the most common encrypted in modern application databases. However, if one uses a weak algorithm this would create a risk of hackers cracking the password from a data leak

The idea of encryption is simple: data can be scrambled by a sender with an encryption key, and then unlocked by the receiver with a decryption key. Symmetric encryption means that the encryption key and decryption key are the same. We will use this method to create a key which we will use to encrypt data and store it to the database.

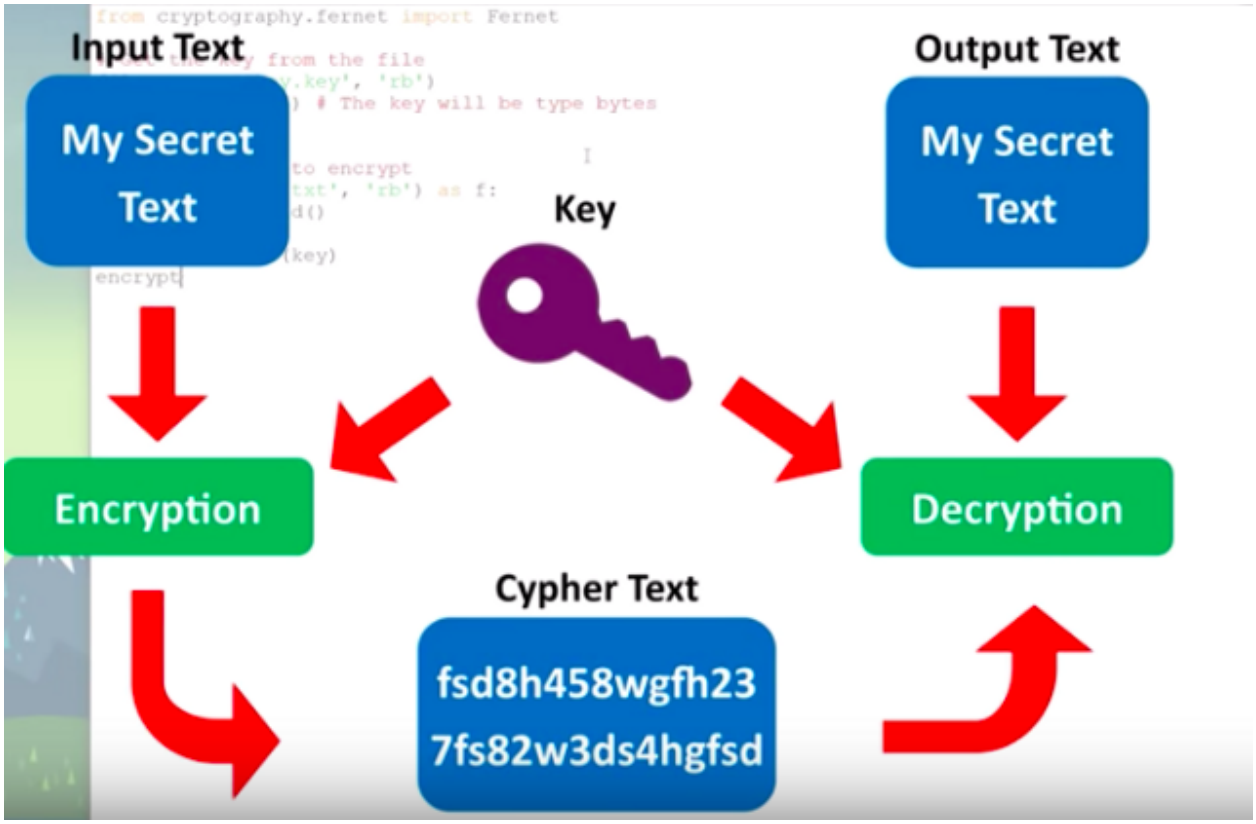
Asymmetric encryption is a form of cryptography where keys come in pairs, consisting of a public key and a private key. The private key is kept secret, while the public key can be shared with anyone. Anything encrypted with the public key can be decrypted with the private key.

The most common form of asymmetric encryption is public-key encryption, which is a type of encryption that uses two keys: a public key, which anyone can use to encrypt a message, and a private key, which only the intended recipient can use to decrypt it.

Good encryption keys should be no smaller than 2048 bits which is estimated to take a computer over 10 billion years to crack it.

In this project, we created a sample sign up and login python application which uses mariadb, a variant of MySQL to store user data.

Symmetric encryption is a type of encryption where only one key is used to both encrypt and decrypt data. This key must be kept secret, or else anyone who has it will be able to read the encrypted data. The image below shows how this type of encryption works.



The following tools were used in designing the project.

- **Python3 -** Python security is to always sanitize data (remove sensitive information) from external sources whether the data originates from a user input form, scraping a website, or a database request.
- Python cryptography module. pip3 install cryptography

```
~/data-enc git dev ❯ pip3 install cryptography
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: cryptography in /usr/lib/python3.10/site-packages (38.0.1)
Requirement already satisfied: cffi>=1.12 in /usr/lib/python3.10/site-packages (from cryptography) (1.15.1)
Requirement already satisfied: pycparser in /usr/lib/python3.10/site-packages (from cffi>=1.12->cryptography) (2.21)
~/data-enc git dev !1 ❯
```

Overall architecture;

1. **MySQL**

   MySQL is a relational Database Administration framework which is a free Open Source Program Beneath GNU Permit. It is additionally supported by Prophet Company i.e. Oracle .It is quick , versatile, simple to utilize database administration Framework. MYSQL bolster numerous operation frameworks like Windows, Linux, MacOS etc.

   MySQL is an Organized Inquiry Dialect which is utilized to control, oversee and recover information with the assistance of different Queries. MySQL is created and backed by MySQL AB which could be a Swedish Company and composed in C and C++ programming dialect. It

was created by Michael Widenius and David Hughes .It is frequently used to say that MYSQL is named after the girl of the co-founder MIchael Widenius whose title is ‘My’.

2. **MariaDB**
   MariaDB is a fork of MySQL, and that is why the two share many similarities. It was created by the original MySQL developers. Its DBMS comes with data processing capabilities for small and enterprise tasks.
   You can think of MariaDB as an improved MySQL version. It is shipped with many powerful inbuilt features and usabilities, performance, and security improvements that are not available in MySQL.

   MariaDB is fully GPL licensed and it has been made available to everyone. It is now in the top 10 list of the most widely used Database management systems worldwide. Some of the top companies using this database include Wikipedia, Google, Tumblr, Ubuntu, Amazon Web Services, RedHat, and others. You can use it on Windows, Linux, and Mac OS. It is safe, easy to master, and convenient, and this can be attributed to its growing popularity.

   MariaDB uses the client/server architecture with the main Database and many clients that request and manipulate the data. Clients use SQL statements to interact with the server and to retrieve data and present the results of manipulations on the client-side. It uses similar security measures as MySQL.

3. **Python3**
   Python3 is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

   Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0.Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.Python consistently ranks as one of the most popular programming languages.

4. **Linux**
   Linux is one of the most popular versions of the UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.
   Linux Operating System has primarily three components

- Kernel − Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.

- System Library − System libraries are special functions or programs using which application programs or system utilities access Kernel's features. These libraries implement most of the functionalities of the operating system and do not require kernel module's code access rights.

- System Utility − System Utility programs are responsible to do specialized, individual level tasks.

5.Visual Studio

Visual Studio Code is a lightweight open source text editor developed under Microsoft and can be contributed to through the GitHub repository vscode. Extra functionality for Visual Studio Code is provided by means of extensions, which can also be developed by third party developers.

FUNCTIONALITIES

1. **Generate key file.** We have a function that uses cryptography module and specified password to generate the encryption decryption key.

```python
# Generate a key
@app.command(short_help='Generate a key and salt it. ')
def generate_key(password):
    """Generate a key from the password and salt and return it   """
    password = password.encode('utf-8') # Convert to bytes
    os.urandom(16) # Generate a random salt
    salt = b'\x8b\x9d\x8f\x9e\x8b\x9d\x8f\x9e\x8b\x9d\x8f\x9e\x8b\x9d\x8f\x9e'    # Hardcoded salt

    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        length=32,
        salt=salt,
        iterations=100000,
        backend=default_backend()
    )
    key = base64.urlsafe_b64encode(kdf.derive(password)) # Can only use kdf once
    console.print('🔐 Key generated successfully \n', style='bold green')
    return key
```

2. **Insert data**

INSERT statement in SQL Server is used for adding records in a table within the specified database. SQL Server performs insertion operation in two ways within a single query: Add data in a single row. Add data in multiple rows.
We insert encrypted data to the database, its binary encoded to preserve its structure and ensure that there won't be modification. The user name is not encrypted because there is minimal risk of someone knowing only your name.
Email and password is encrypted and the various checks are done to ensure user confirm the password.

3. **Get data**
   This function is used to retrieve data in the database and print it to show that all the data in the database is encrypted symmetrically using the key.

ID	Name	Email	Password
5	Jane Doe	gAAAAABjdqc_Va_GO9eA3JH2YqZQCSzKAls5mYhPJMR7HvK2e01KNdqynjuVBEzi2…	gAAAAABjdqc_LYRdAR3TSmOX3UHmk1HRmVHI4WxEINCYVjlqpPcOvLPRaCJ5WwYhey…
6	user one	gAAAAABjdqzGkSYfIEtm41hUGdk-EOuB8VPZlRJIjHjzYmQY_xWxD7YgSS0m8YgxV…	gAAAAABjdqzG3lfirUVTbqpLSY3lUTdMaD_sDJgRPdeaoqp81mUsXTmSPG1vT106y0…

4. **Regenerate key** if you feel insecure of the other key being compromised, you can create a new key and replace your old one. NOTE. Any old data won't be accessible if you lose the key.
5. **Save key file** We need to save a key so that we are able to actually use it to encrypt. To save the key we can write it into a file. The code below will save it in a file within the folder you are working in currently.

```
# Save the key
@app.command(short_help='Save the key to a file. ')
def save_key(key):
    """Save the key to a file called key.key  """
    with open('credentials/key.key ', 'wb') as key_file:
        key_file .write(key)
    key_file .close()
    console .print(f'🔑 Key: {key}\n', style='italic green ')
    console .print('✅ Encryption Key saved to file at credentials/key.key   ', style='
bold green ')
```

6. **Load env variable**
   This section lists environment variables that are used directly or indirectly by MySQL. Most of these can also be found in other places in this manual.

7. **Retrieve the key for use during the decryption process.**
   If you want to retrieve the key, we can retrieve the key from the file we saved with read bytes.

```
# use the key to encrypt user details
with open('credentials/key.key ', 'rb') as file:
    enc_key  = file.read()
    file.close

fernet  = Fernet(enc_key )
```

SECURITY FEATURES

1. **OpenSSL**
   OpenSSL is a software library for applications that secure communications over computer networks against eavesdropping or need to identify the party at the other end. It is widely used by Internet servers, including the majority of HTTPS websites. It contains an open-source implementation of the SSL and TLS protocols.
2. **The Cryptography module** provides a number of cryptographic services to Python applications. These services include:
   - Encryption and decryption
   - Message signing and verification
   - Password hashing
   - Key generation
   - Random number generation

```
# imports go here
import hashlib
import os
import sys

import cryptography
import typer
from click import confirm, prompt
from config import Database as DB
from cryptography.fernet import Fernet
from dotenv import load_dotenv
from rich.console import Console
```

3. **Crptography module Fernet class**
   Fernet is a symmetric encryption method that makes use of an authentication tag in order to ensure that messages cannot be tampered with. It is based on the concept of a "web of trust", in

which each user has their own set of keys that they use to encrypt and decrypt messages. Other users in the network can then use these keys to verify the authenticity of the message.

Fernet is designed to be simple to use and easy to integrate into existing applications. It uses the AES encryption algorithm and supports key sizes of 128, 192, and 256 bits. Fernet also supports key rotation, which allows users to change their encryption keys on a regular basis without having to re-encrypt all of their data.

Fernet is open source and is available under the MIT license.

4. **Password salting** A salt is a random string of characters used to secure passwords. Salts are added to passwords to make them more difficult to crack.

```
salt = b'\x8b\x9d\x8f\x9e\x8b\x9d\x8f\x9e\x8b\x9d\x8f\x9e\x8b\x9d\x8f\x9e'    # Hardcoded salt

kdf = PBKDF2HMAC(
    algorithm=hashes.SHA256(),
    length=32,
    salt=salt,
    iterations=100000,
    backend=default_backend()
)
```

Limitations of security features used in the project.

- The main limitation of encryption is that it can only be used to protect data at rest, not data in transit. Additionally, encryption is only as strong as the key that is used to encrypt the data; if the key is weak, the data can be easily decrypted.

- There are a few potential limitations to using password salting:
  1. It can be difficult to manage salts across multiple systems and ensure that they are kept synchronized.

  2. If a salt is compromised, all passwords that use that salt are also compromised.

  3. Password salting does not necessarily protect against all types of attacks, such as dictionary attacks or brute force attacks.

- There are a few limitations to database encryption:
  1. Encryption can impact performance.

  2. Encryption can add complexity to the database design and implementation.

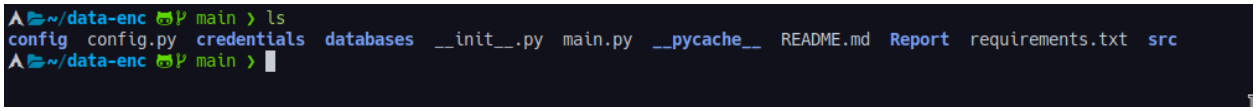
  3. Encrypted data can be difficult to query and analyze.

- Openssl is based on data transport and provides encryption during data transit. Some information in the network is not encrypted and thus proves difficult in encrypting some data.
- Form fields such as encrypting user names does not prove efficient. The only options may be encrypting user email, password, credit card information etc.
- Some features in encryption prove to be difficult to implement in your code with some requiring entirely importing modules or frameworks.
- Data encryption can introduce security risks if not implemented properly. For example, if data is encrypted with a weak encryption key, it can be easily decrypted by attackers.

Project Running and Installation.

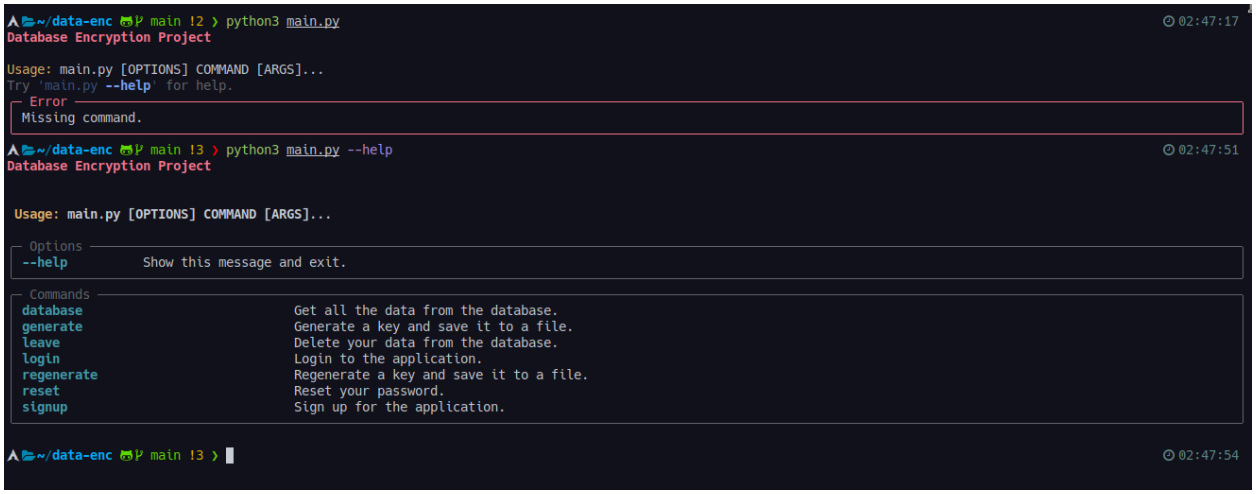
Step 1. Unzip the file and open the new folder from the zip. Or clone the project from github and navigate to the repository.

Step 2. Open Terminal and list directory contents if you are on Linux, its simply ***ls*** command and on windows ***Get-ChildItem***

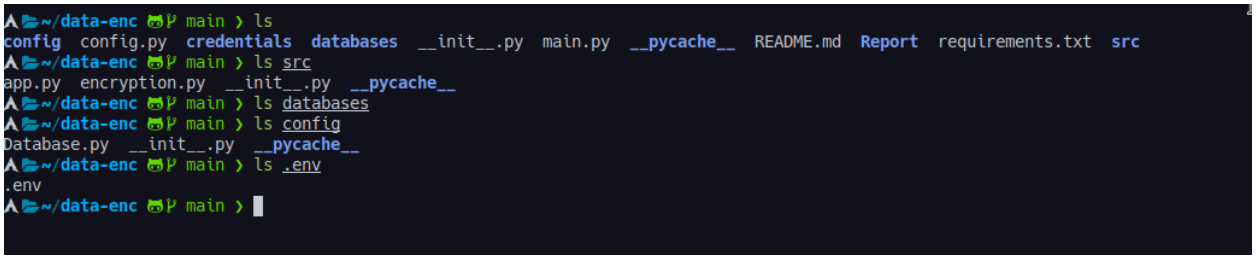


Run this command to ensure you have all required modules for the project. You also need to have **python 3.10** installed to your environment.

Step 3. Run the file main.py using python3.
python3 main.py
python3 main.py –help
This will display all available commands possible for the application



The **class file app.py** and **encryption** module are located at **src** directory.



These files are imported where necessary in the **main.py** file and are not supposed to be executed independently.

For colors in certain output messages and also the command feature we use the **Rich module** and **Typer module** for the beautiful formatting and table representation of **get_data** function.

1. **Sample data processing. Inserting encrypted data to the database**

```
~/data-enc  main ❯ python3 main.py signup
Database Encryption Project

Enter your details to signup

Enter your name: : Mark John
Enter your email: : markjohn@gmail.com
Enter your password: :
Confirm your password: :
Data inserted successfully
~/data-enc  main !2 ❯
```

2. **Log in and failed login**

```
~/data-enc  main !2 ❯ python3 main.py login
Database Encryption Project

Enter your username: : john
Enter your email: john@gmail.com
Enter your password:
Wrong password!
```

3. **Key Generation**

```
~/data-enc  main !2 ❯ python3 main.py regenerate
Database Encryption Project

Enter your password to regenerate your key

⚠WARNING: This will delete your old key and replace it with a new one. Make sure you have a backup of your old key before proceeding.

Enter the password: :
Confirm the password: :
🔐 Key generated successfully

🔑 Key: b'GVjEnTz0RC97EV3NXVpmMvx5p34fFB1-0bsBAq5mAzQ='

✅ Encryption Key saved to file at credentials/key.key
b'GVjEnTz0RC97EV3NXVpmMvx5p34fFB1-0bsBAq5mAzQ='
~/data-enc  main !3 ❯
```