**Group Two Development Team Project**

**Symmetric Cryptography and Password Hashing**

**Installation of tools**

Python provides a number of symmetric cryptography and password hashing choices within it’s hashlib library which encodes data and makes it visible to intended recipients only. symmetric cryptography utilizes a single key to encode and decode plain text. The key must be secured or else the capability to encode and decode is lost (King, 2018). Prior to executing cryptography source codes created by Group Two, Installation of the following software is paramount:

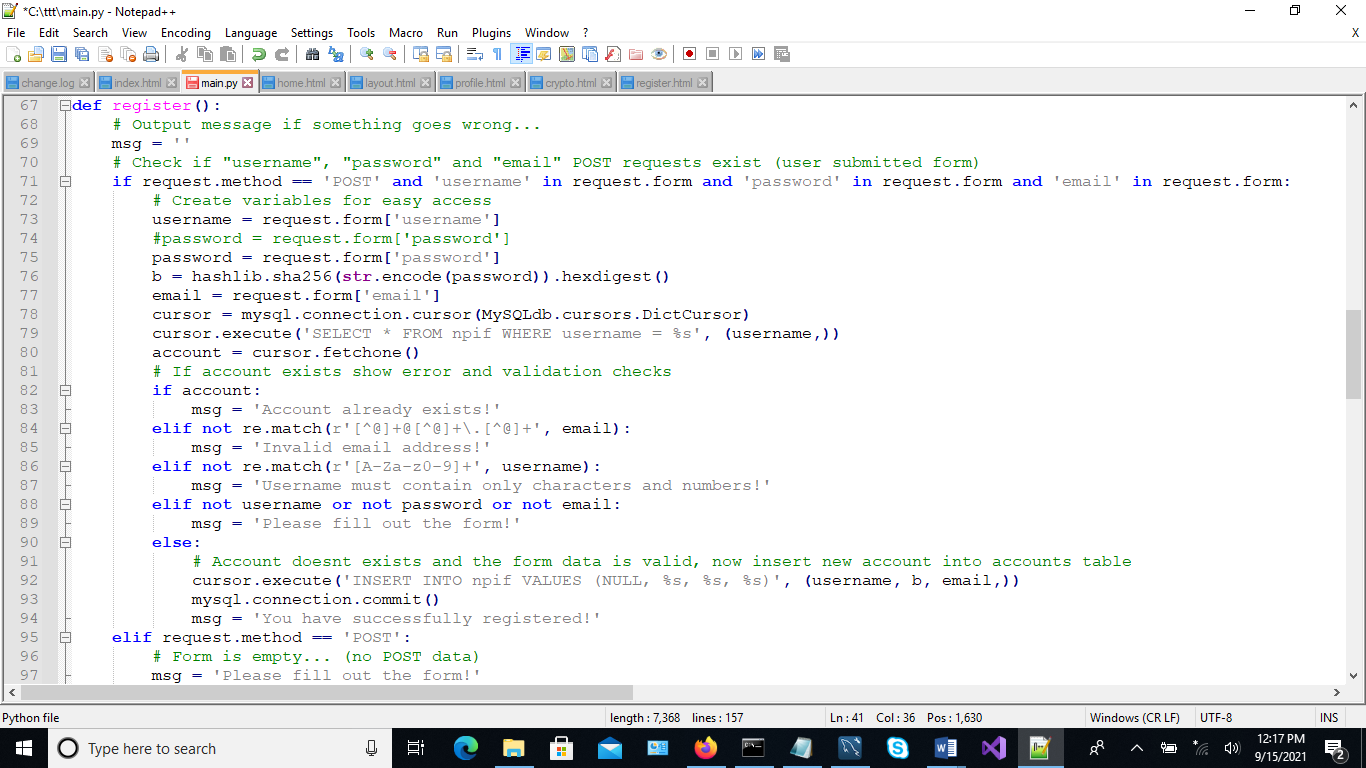
1. MySQL Community Server 8.0, MySQL WorkBench CE & MySQL Tutorial. You can download MySQL installer which contain these packages from <https://dev.mysql.com/downloads/file/?id=506568> and install the above named software. Configure the **root password** as **SSD123456p**
2. Python 3.9.6 for Windows. You can download Python 3.9.6 installer from <https://www.python.org/ftp/python/3.9.6/python-3.9.6-amd64.exe>. After the download is complete, proceed with the installation of the software mentioned above.

After installing Python 3.9.6 it is important for you to install all the necessary libraries that support Flask, cryptography and MySQL Database. To install the library, go to the command line and type the following commands then press enter key:

1. pip install mysql-connector-python
2. pip install cryptography
3. pip install Flask
4. pip install hashlib

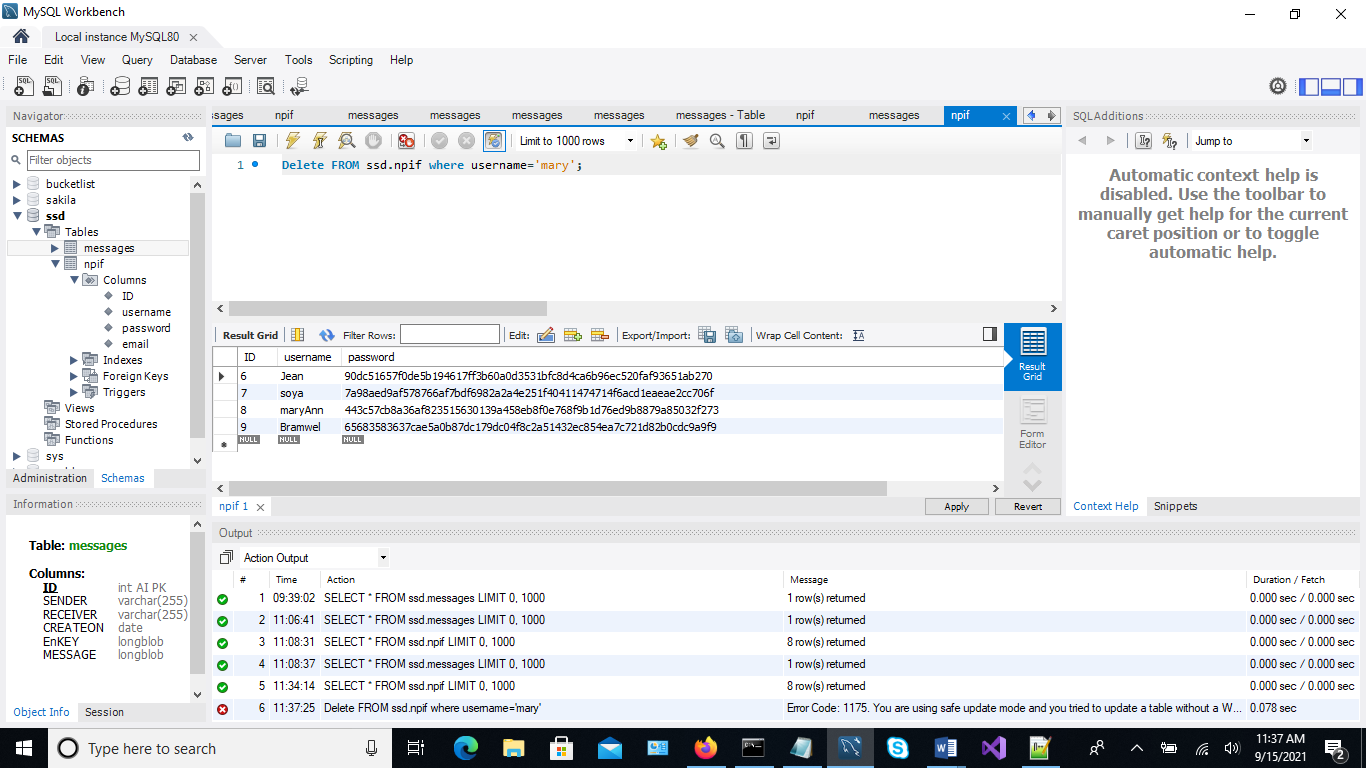
**Securing Passwords Using Hashlib Library**

Hashing is so widely used in systems to conceal passwords. It has been used in this system to hide passwords in the database. The system can be used by multiple users with different ranks within the Dutch Police Internet Forensics for example, when a user registers, the password will be hashed using sha256 as shown in line 76 of the source codes in **Figure 1.** The data is then validated and inserted in MySQL database.



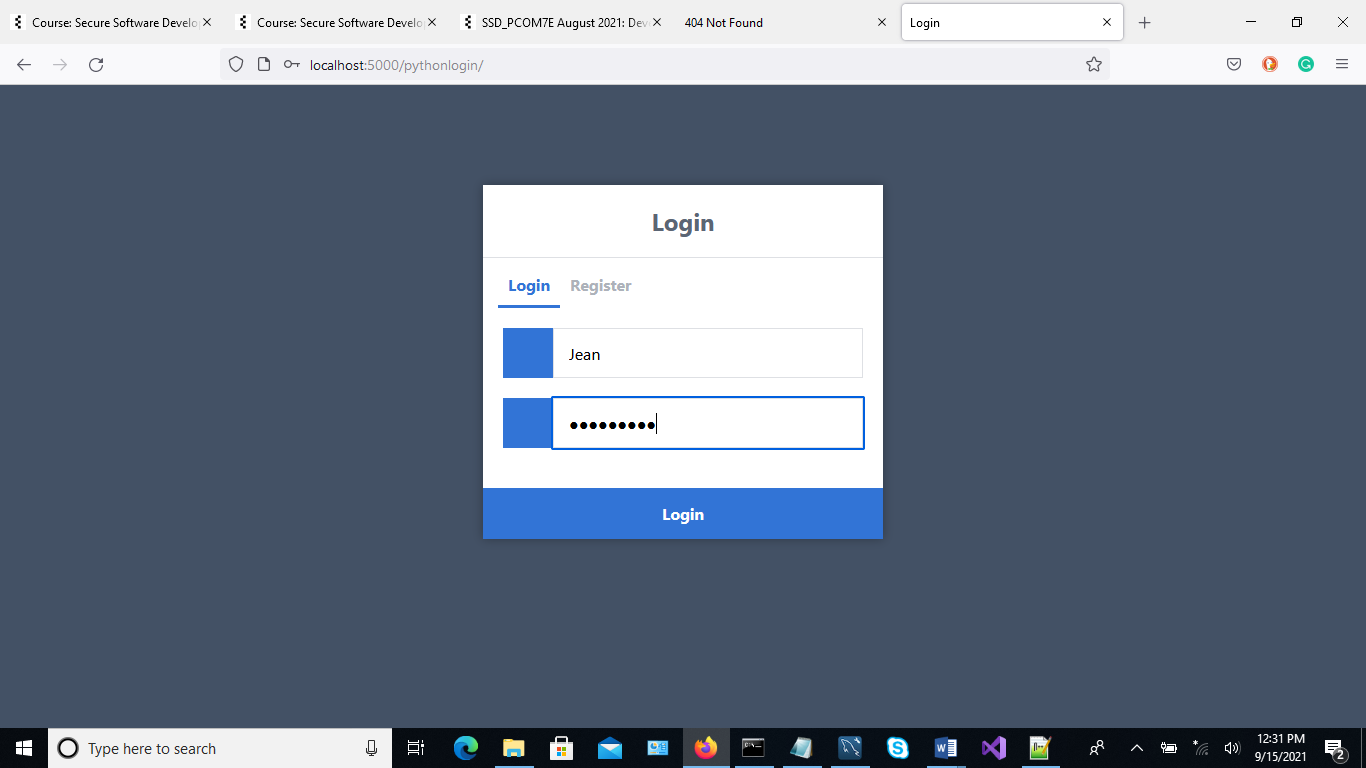
**Figure 1:** Password hashing with SHA 256 adopted and modified from David (2020)

**Figure 2**: shows the registered users and their hashed password in MySQL database



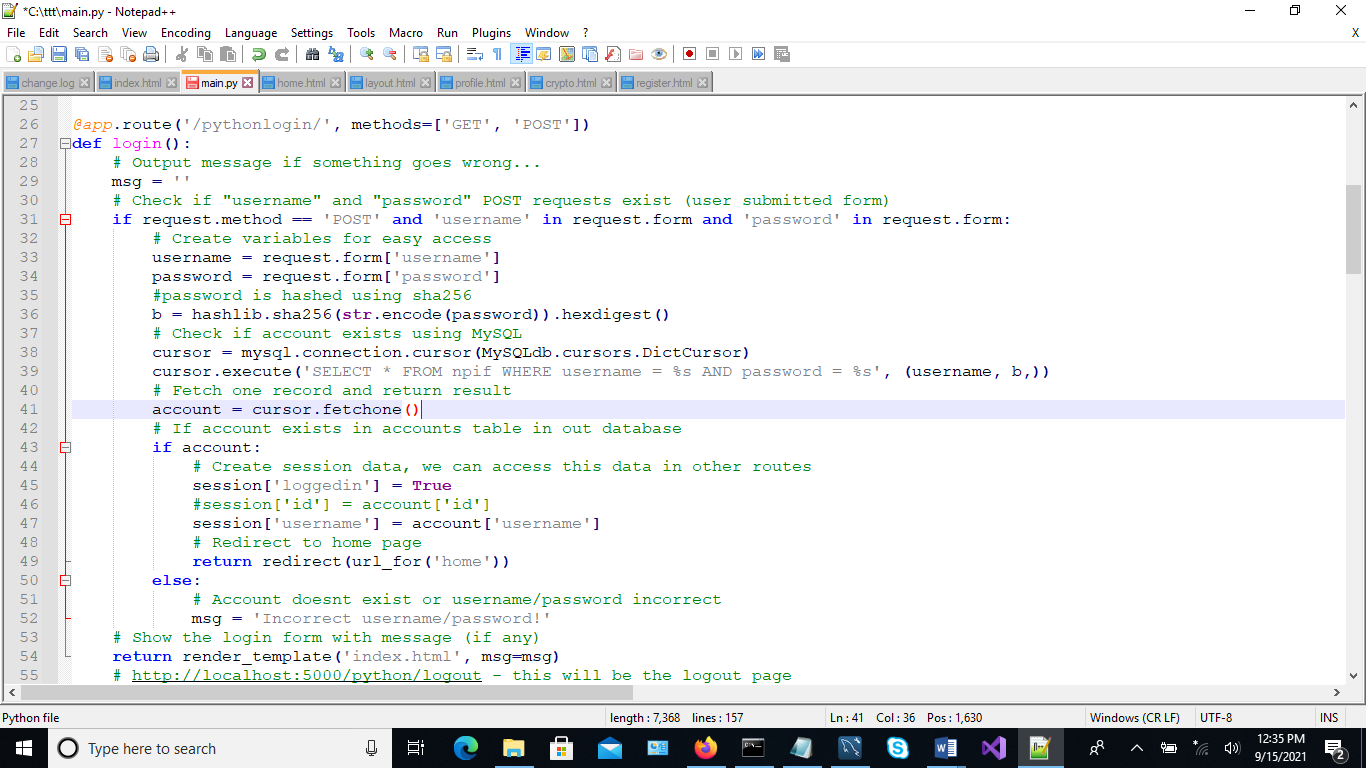
**Figure 2: Hashed User passwords in Mysql database**

When the user attempts to login, the password that they type into the system is hashed and compared with that stored in MySQL database during the registration process. If they are equal, the user is allowed to login. **Figure 3** shows the source codes.



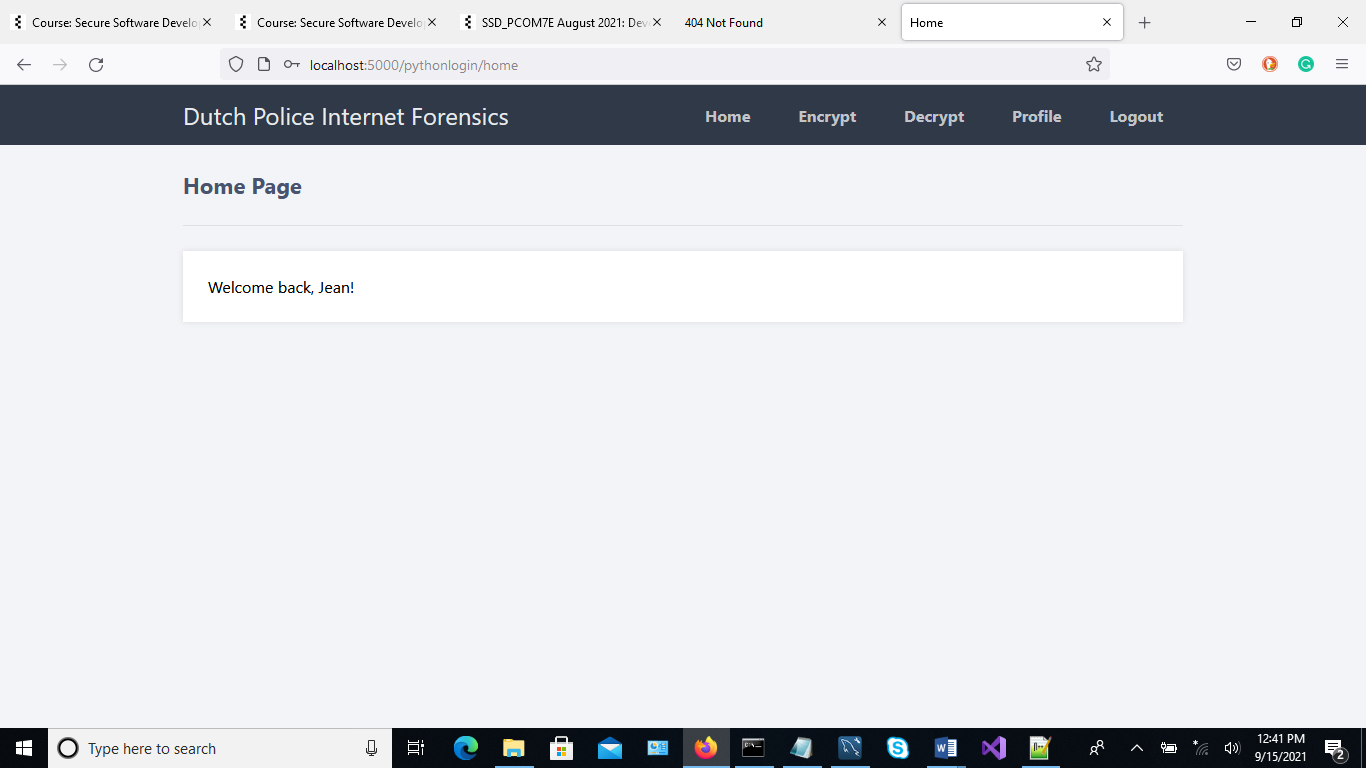
**Figure 3: User Login**

Line 36 of the source codes in **Figure 4** shows the password being hashed.

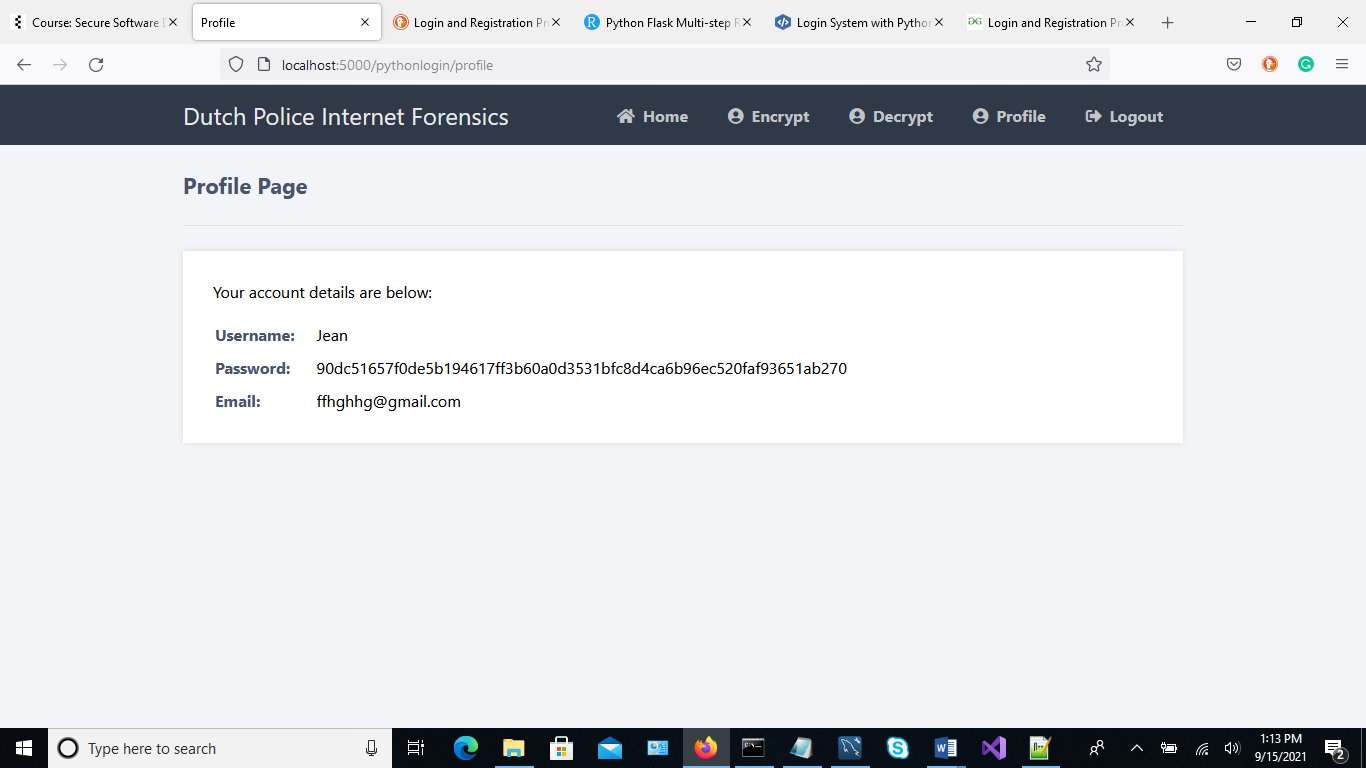


**Figure 4: User Login Source Code** adopted and modified from David (2020)

For example, **Figure 5**: shows the home page once Jean logs in the system. When Jean clicks on profile, **Figure 6** shows that her password is hashed.



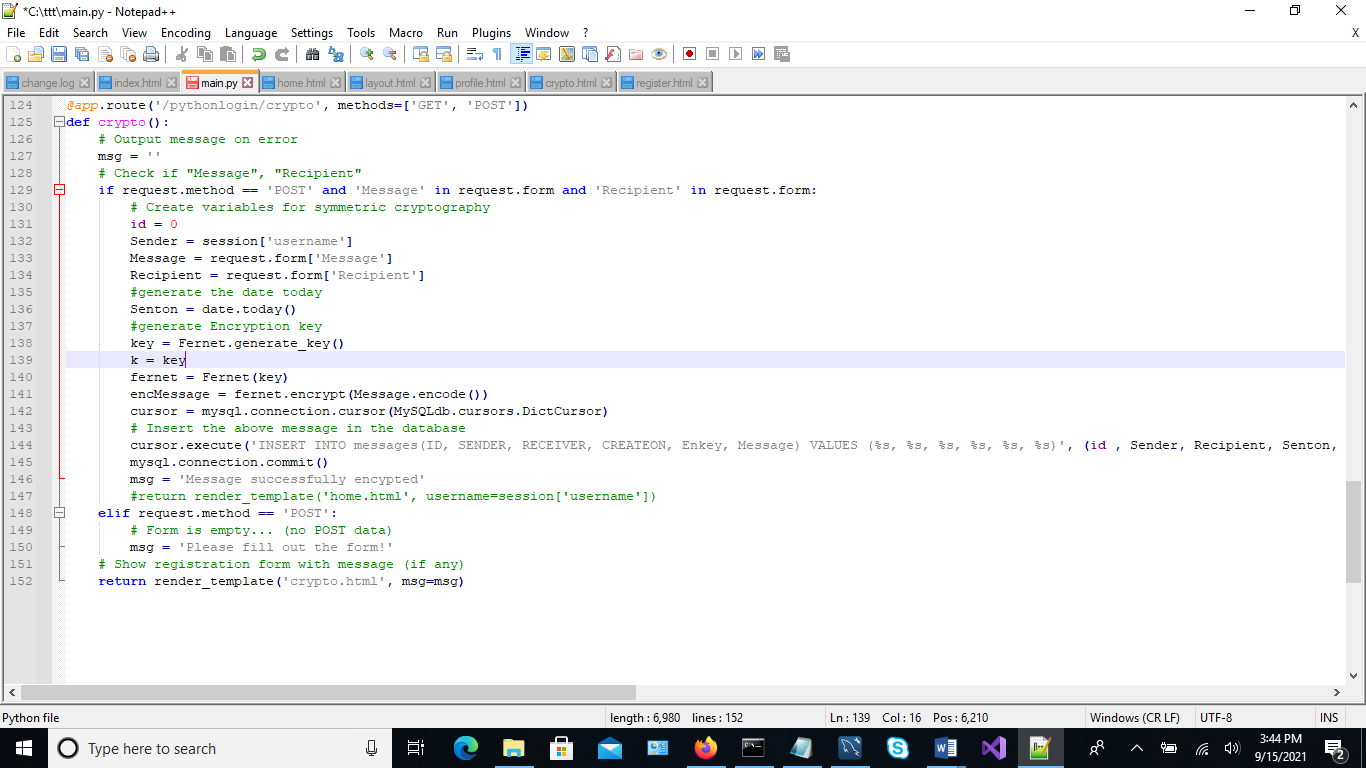
**Figure 5: User Login Source Code**



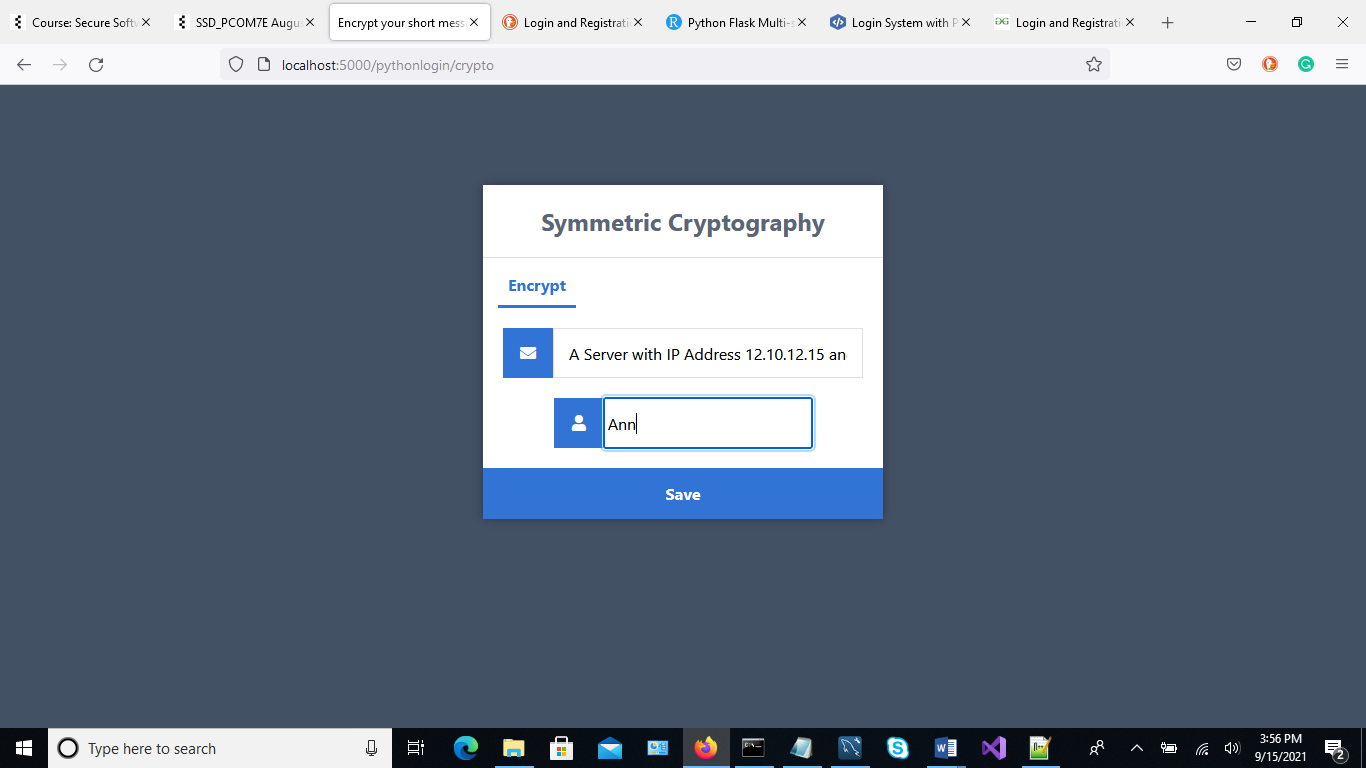
**Figure 6: Hashed password**

**Symmetric Encryption**

The symmetric cryptography prototype has been inspired from King (2018) and modified to be able to encrypt and decrypt data while the output is stored in MySQL database. Every time a user creates a new message, a key is auto generated and used to encrypt the message. The encoded message and the key are then stored in the database. The key is then used on a later date and time to decrypt the message by the intended recipient. **Figure 7** shows symmetric encryption source codes used to encrypt the messages. To encrypt a message, On the menu click on Encrypt to encrypt the message and the form shown in **Figure 8** will open. Enter the message and the recipient’s name then click save.

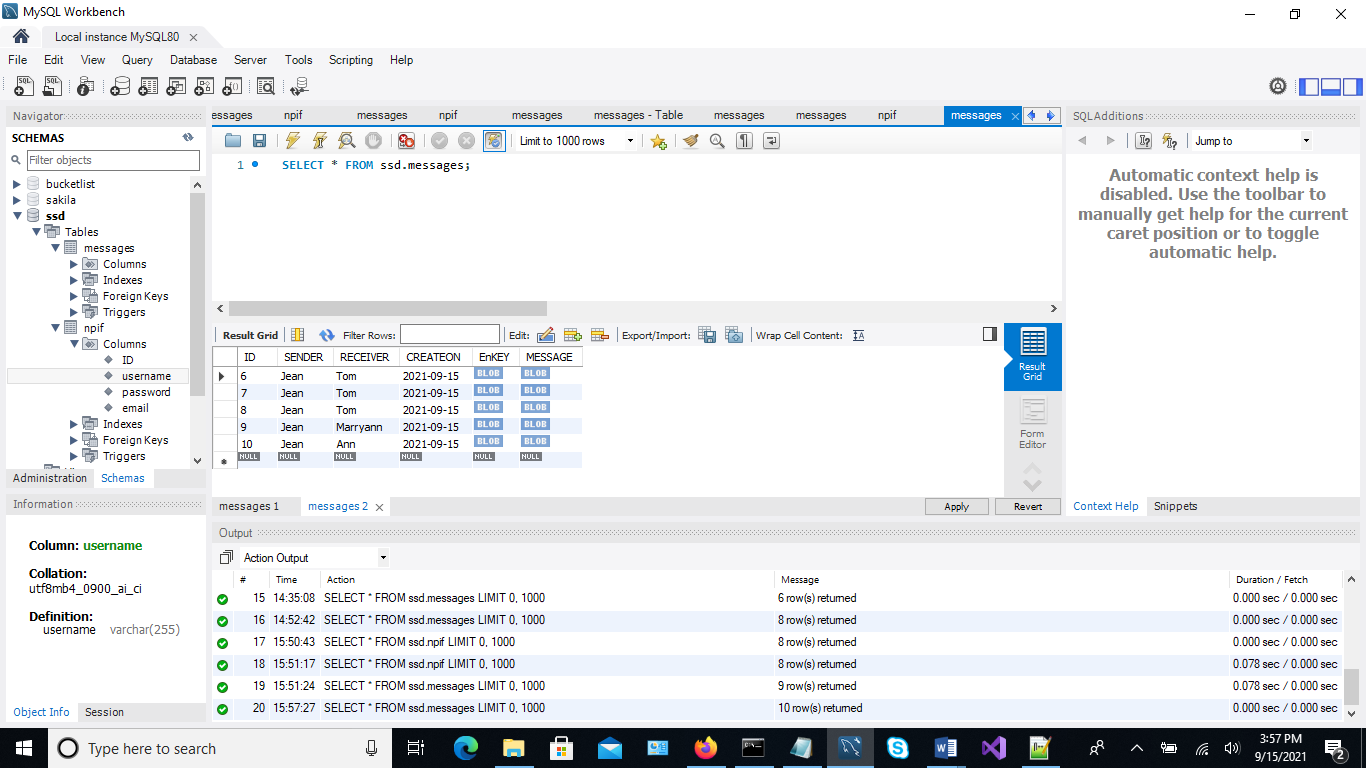


**Figure 7: Message Encryption Form**



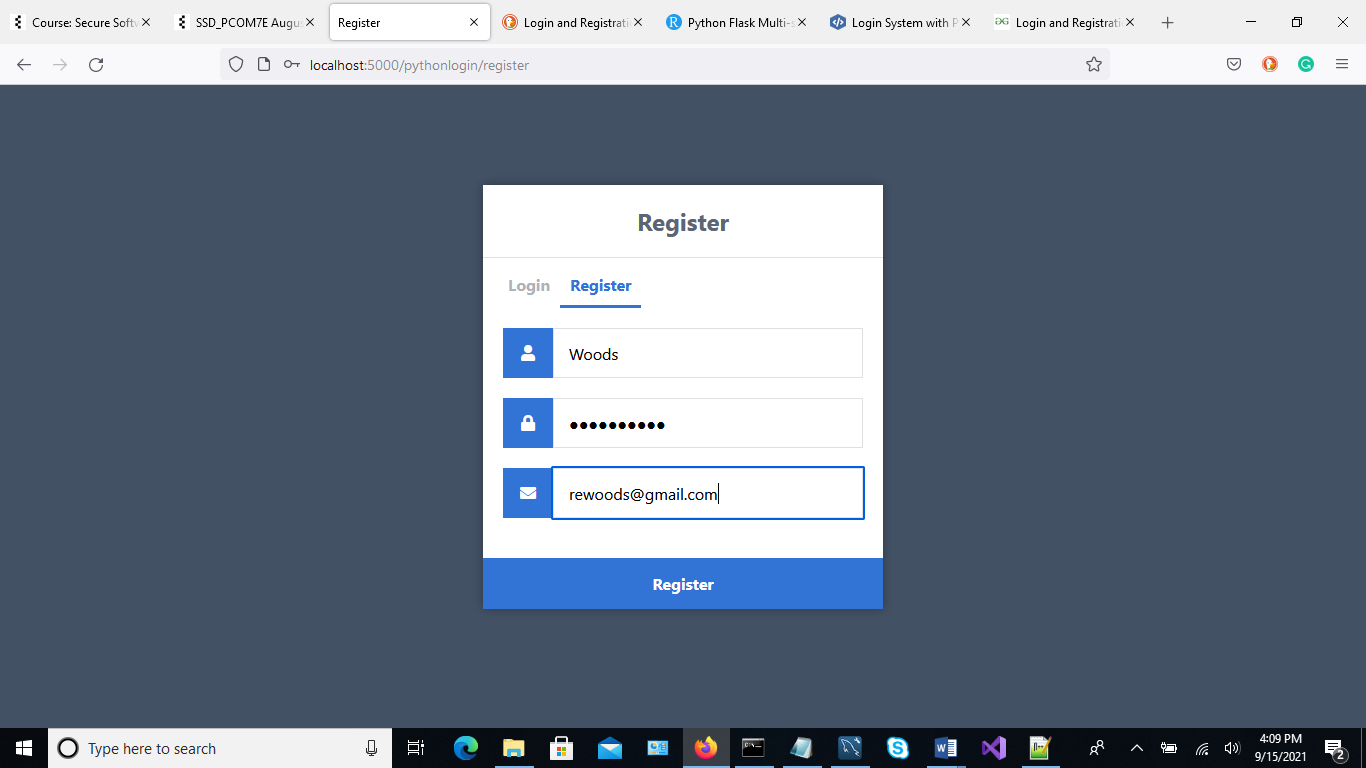
**Figure 8: Message Encryption Form**

**Figure 9** shows how the records look like in MySQL WorkBench. The keys and messages are seen as blobs. This helps to secure the data from unauthorized users. A point to mention is that in a real life application, it would be a better option for the encryption keys to be stored in a separate server. The datatypes can also be renamed to conceal what they store.



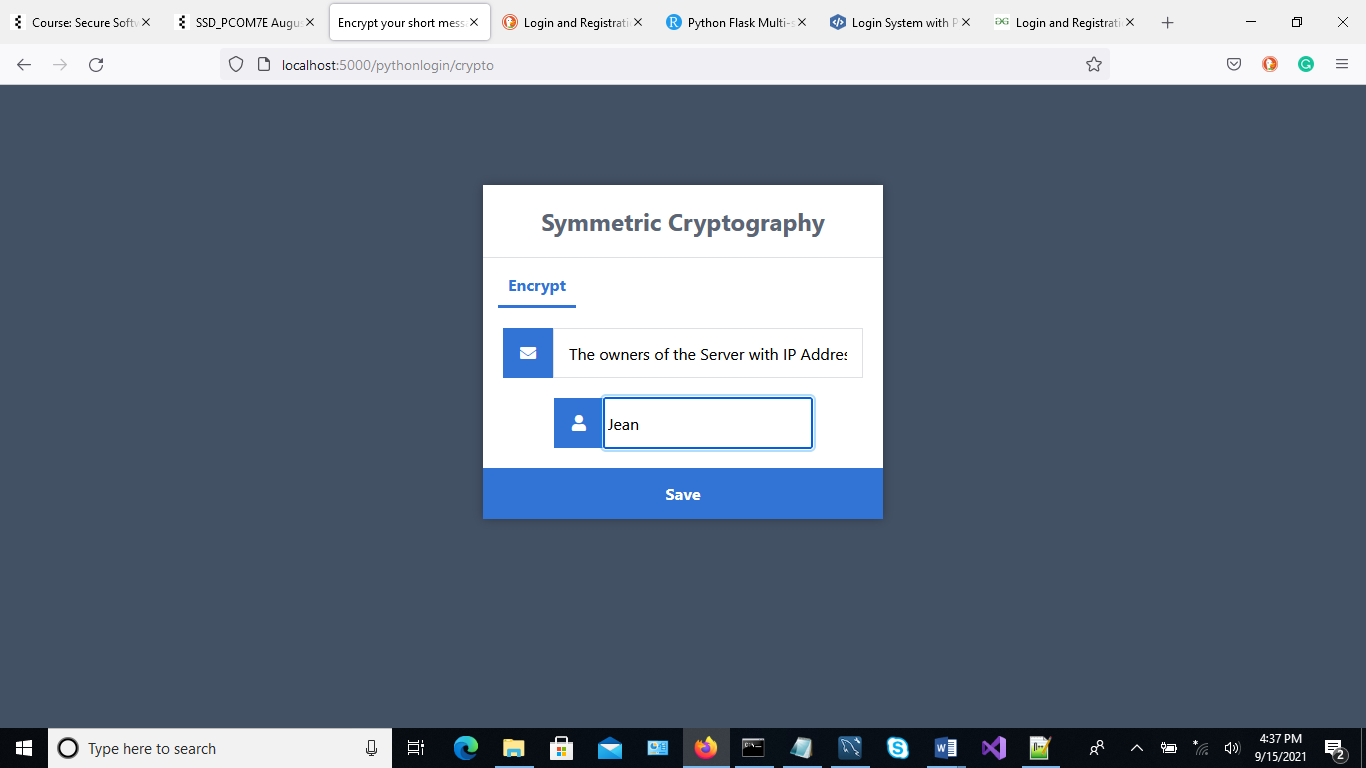
**Figure 9: Encrypted Messages**

We can now register a new user who will now send a message to Jean. On the menu click on logout. On the login form click on register. Enter the Username as **Woods**, password as **Dunlop#88!** and email address as [**redwoods@gmail.com**](mailto:redwoods@gmail.com)as shown **in Figure 10 .** Click on register and login as Woods.



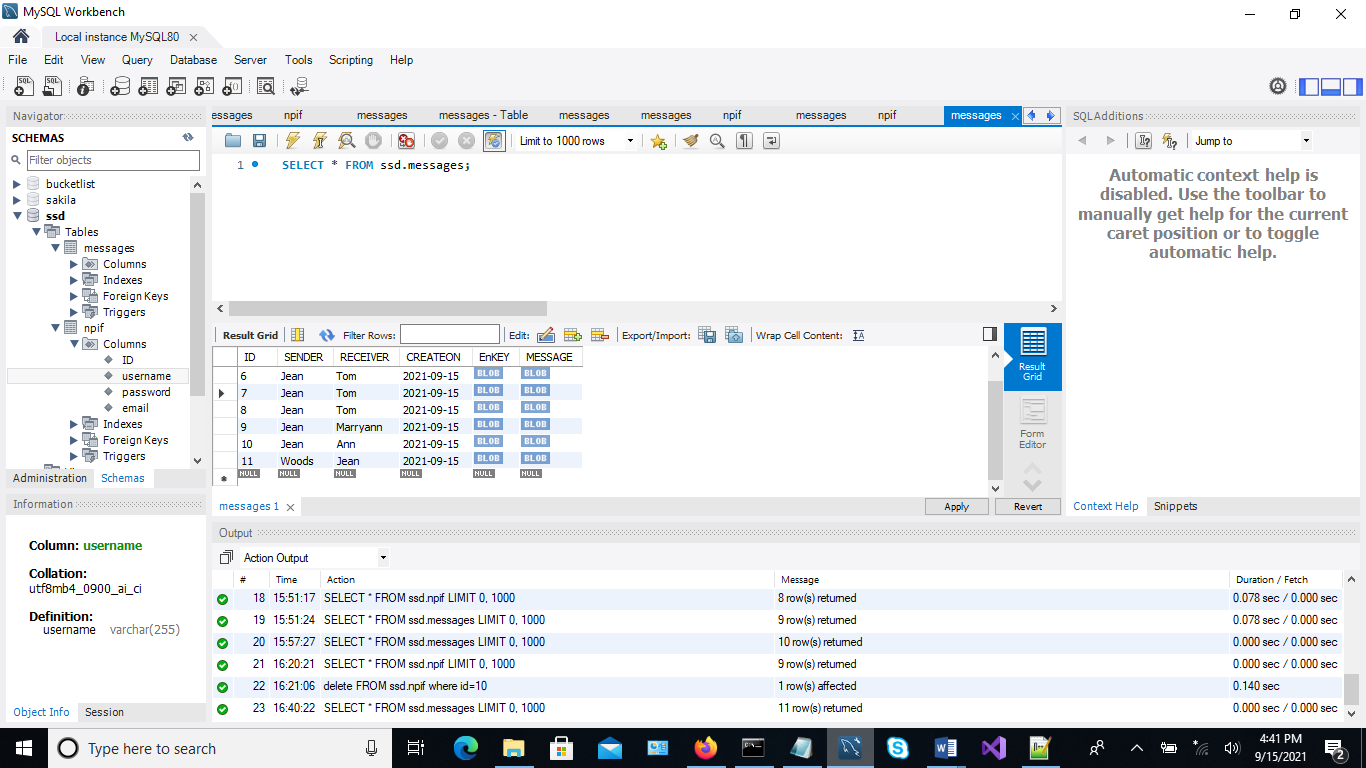
**Figure 10: New User Login**

On the menu click on Encrypt, enter a message that you intend to send to Jean as shown in **Figure 11** and click on save.



**Figure 11: New message for Woods**

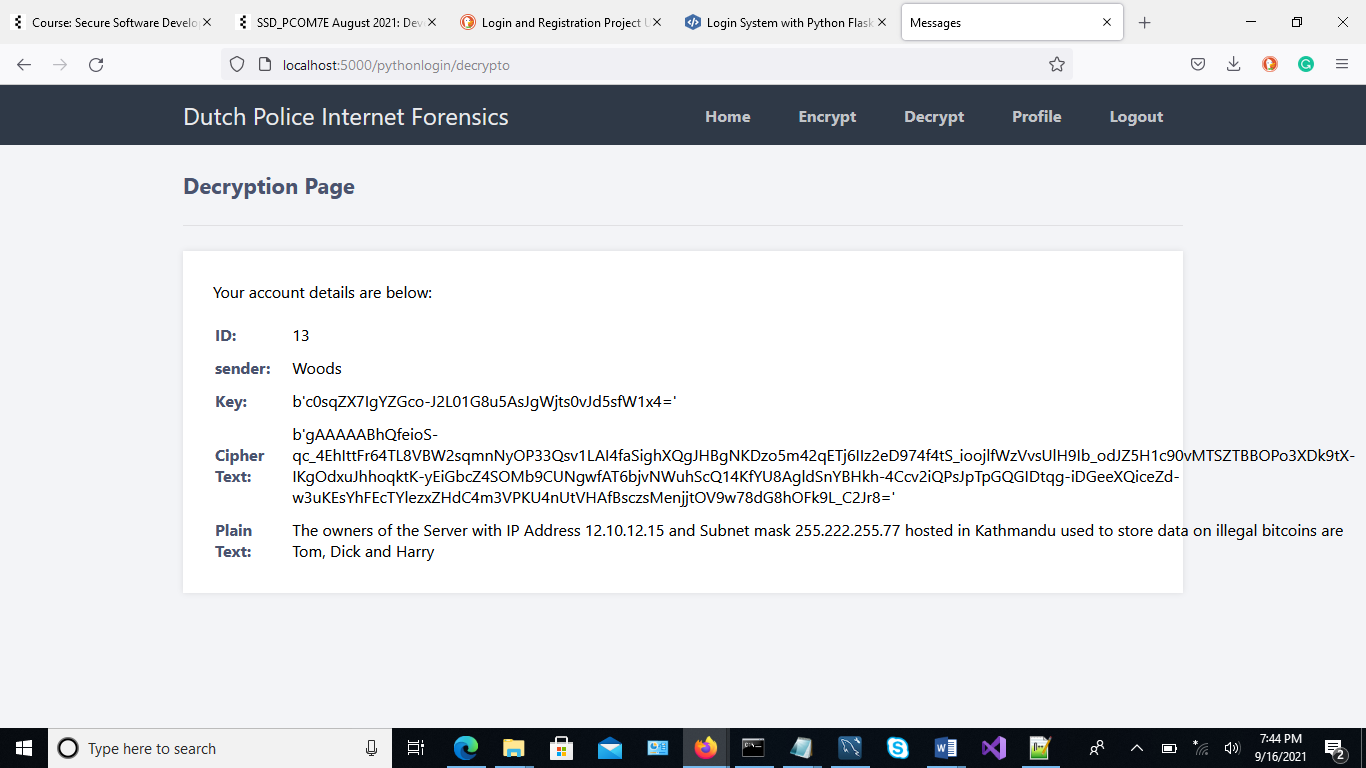
From MySQL workbench, the output now looks as follows:



**Figure 11: Saved Message for Woods**

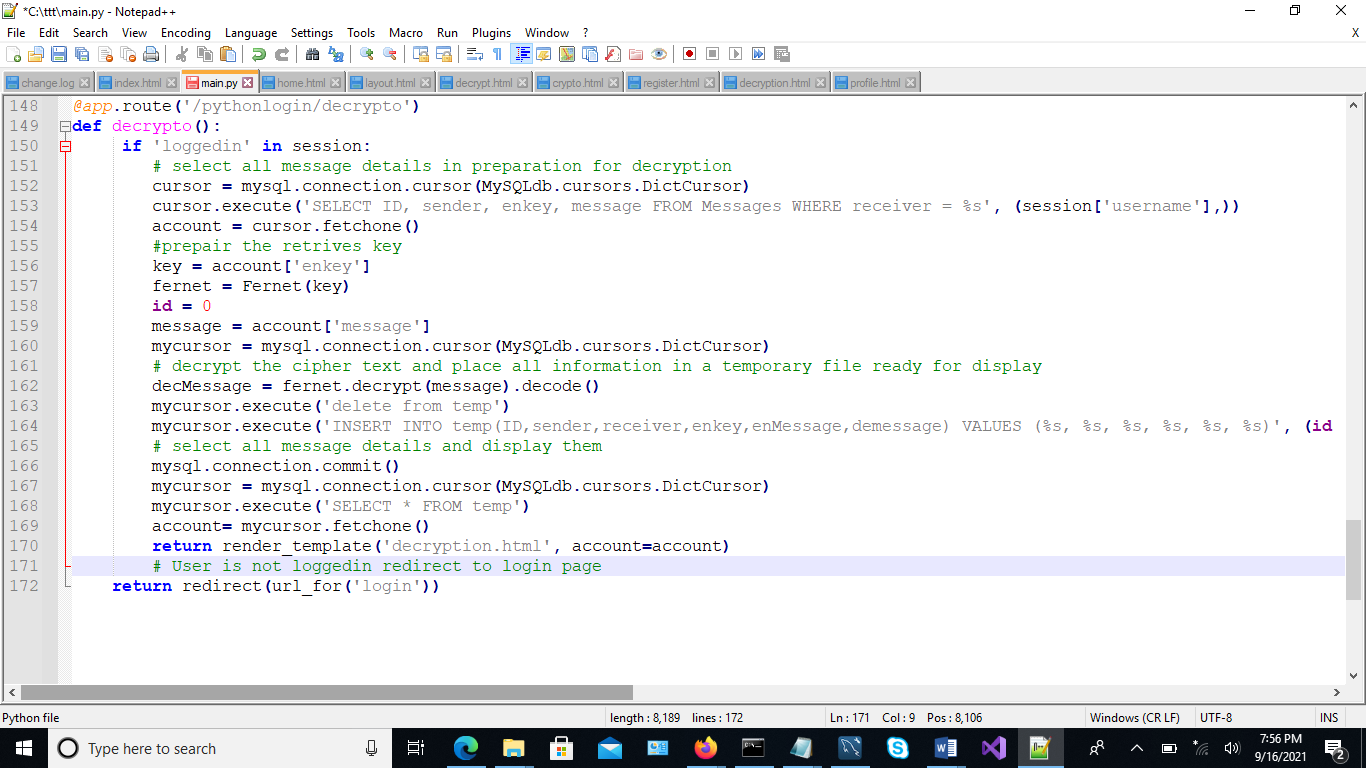
**Symmetric Decryption**

When decrypting documents, the same key that was used for encryption is used to decrypt the same message. The system retrieves both the message and the key then decrypts the message and finally displays the same. **Figure 12** show the message that was sent to Jean by Woods



**Figure 12: sent to Jean by Woods**

**Figure 13** shows symmetric decryption source codes used in the system



**Figure 13: Symmetric Decryption Source Code**

**Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Number** | **Test Type** | **Working?**  **Yes/No** | **Evidence** |
| 1 | Anonymity and unlikability of the users of the system through symmetric key cryptography | Yes | * In the MSQL workbench the keys and messages are seen as blobs to secure the data from unauthorized users. * The system is made so that the data can be encrypted and decrypted. * The passwords are hashed * User login - username and password ensures security |
| 2 | Perfect forward secrecy maintained to prevent adversaries from obtaining encrypted information |
| 3 | System protects the data against potential attacks/vulnerabilities | Yes | The system aims to protect against the potential attacks and vulnerabilities, although we cannot deploy an attack on the system purposely to check if an attacker can gain the encrypted information through other malicious ways-nonetheless we have tried our best to ensure that the security we have implemented works |
| 4. | System stores encrypted information in a database | Yes | All data is successfully stored into MySQL database |
| 5. | System stores the copy of the encrypted information that is in the database in the c drive in binary files | No | This was a requirement in the design document of the project, however, due to time constraints and merged resources the authors could not cooperate/collaborate with another server to manage the keys. |
| 6. | System should be understandable to use for the end users | Yes | Instructions on how to use the system is clearly demonstrate in this document |
| 7. | System should include registered users login | Yes | This is an extra feature we have added onto the system to ensure utmost security. |

**Conclusion**

In conclusion, the prototype showed how messages could be encrypted, decrypted and stored in a database safely. In real-life applications it is advisable to save the keys in another separate server. Unfortunately, due to time constraints and limited resources amongst team members, the authors could not cooperate/collaborate with another server to manage the keys. Nonetheless, the prototype inspires confidence that symmetric cryptography can guarantee users a secure and better cyber experience.

**References**

1. David, A (2020) Login System with Python Flask and MySQL Available from

https://codeshack.io/login-system-python-flask-mysql/

[Accessed 9 September 2021].

1. King, B (2018) Encrypting passwords for use with Python and SQL Server. Available from

<https://www.mssqltips.com/sqlservertip/5173/encrypting-passwords-for-use-with-python-and-sql-server/>

[Accessed 9 September 2021].