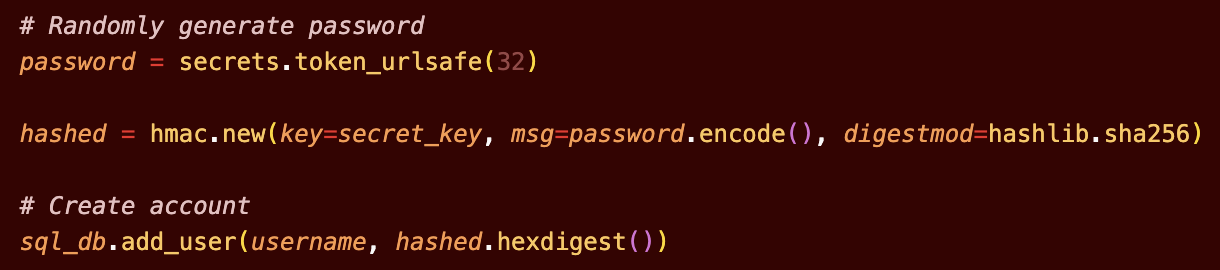
Each of the examination criteria is outlined below.

**Storing passwords.**

Credentials are entered into the form and securely transmitted to the server. Passwords are then stored in the database using a keyed hash. This key was initially randomly generated for security purposes.

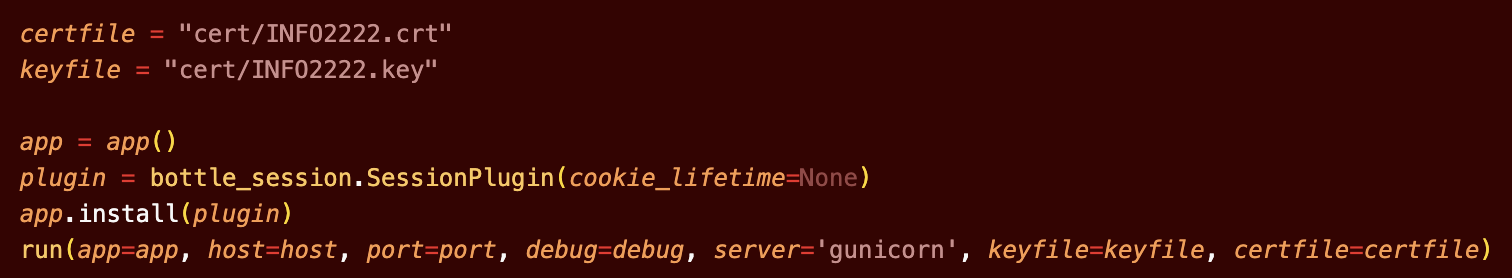
This method of storing passwords is secure as it helps to prevent data compromisation. Even if the server is breached and the database is stolen, the data can not be easily read. Even a precomputed rainbow table would be unlikely to work due to the secret key used in the hashing process.



**Checking the server certificate.**

Following the tutorial instructions, I have created a self-signed certificate and key. Through the implementation of the ‘gunicorn’ server, certificate verification becomes easy. If the certificate is not verified through the keyfile, a PEM Error is triggered, and the server will not run.

Additionally, by adding the certification to my keychain, the details of the certificate are automatically checked against the website when opening (shown in the next criteria through the integration of https) ensuring that it is the correct certificate for the correct website.



**Securely transmitting a password to server.**

The ‘gunicorn’ server allows for https connections to be established (shown through the lock symbol in the screenshot below). For ease of access I have defined the IP to work with <https://temp.website> using /etc/hosts.

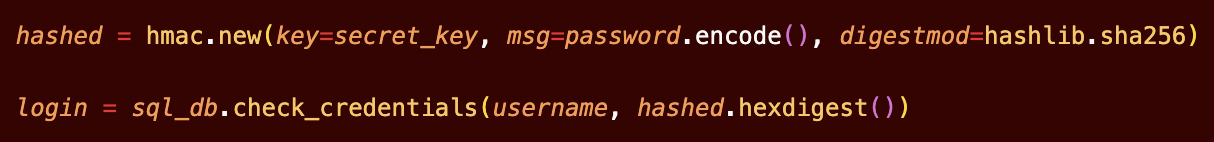
By requiring https connections, the client operates on the basis that there is a valid certificate in place. This ensures that the server is who they say they are. Additionally, this verification integrates Transport Layer Security (TLS) protocols into transmission, protecting data from eavesdropping through encryption and creating message authentication codes to ensure that transmissions remain unaltered.

Graphical user interface, application, Teams

Description automatically generated

**Checking whether password is correct:**

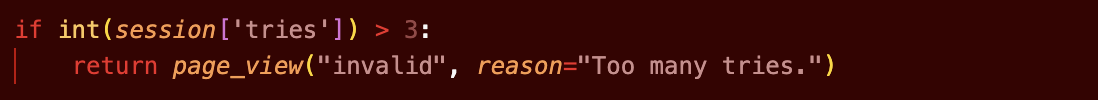
After passwords are securely transmitted to the server, we check if they are correct. As stored passwords have had transformations applied to them (i.e. been through a keyed hash), I do the same for the newly received password.



After identifying the row in the database containing the associated username (also sent through the form), the hashed password is checked against the stored hash, returning a correct login if there is an exact match.

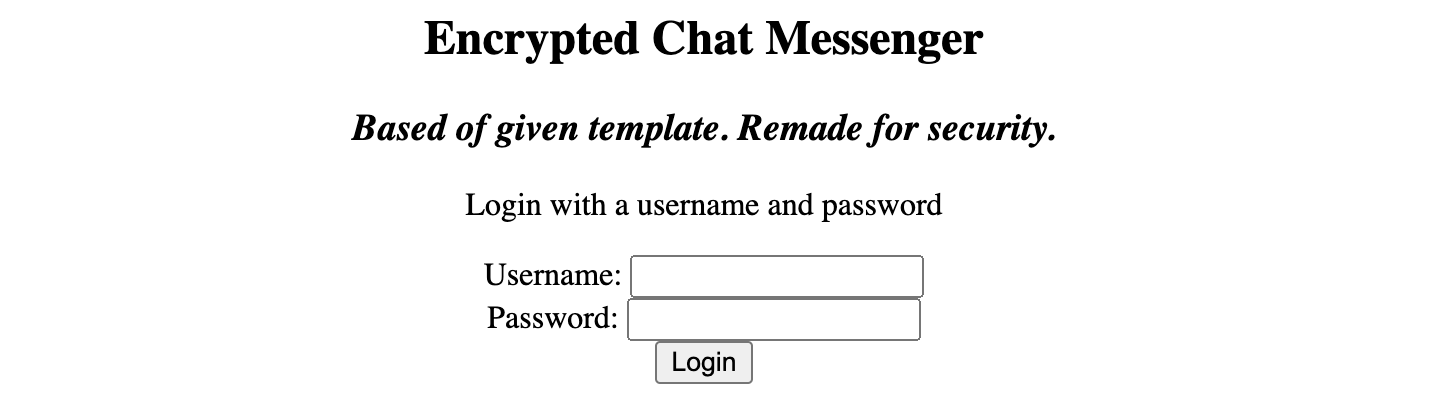
If the user does not exist or the password is wrong, login returns false, and a webpage is shown stating invalid credentials.

Pre-computation methods such as rainbow tables are defended against using keyed hashing. Additionally, there is a maximum number of allowed tries (3) before which the user is locked out, this is kept track of using session cookies.



**Securely transmitting messages:**

To access the chat screen, users must first log in and be verified.



Following this, their friends list is shown. Users can select a friend to begin a conversation with by typing their name. The user is also shown there to initialise a private conversation.

Graphical user interface, text, application

Description automatically generated

Messages are transmitted between two users using end-to-end encryption. This is to secure messages from the server. This is implemented using symmetric key encryption.

Transmitting the key from one client to the other would result in a vulnerability, so a Diffie-Hellman key exchange is used, allowing both sides to agree on a secret key without the server knowing what it is.

This exchange works as each client has secrets a and b respectively. Client A sends ga to client B and client B sends gb to client A. Both sides are then able to compute the secret key as (ga)b=(gb)a=gab.

The prime and base are predetermined, with the secret being randomly generated. The size of the prime is the maximum value a standard Javascript number can hold, although BigInt was used for computation in case numbers got too large.

The implementation of the exchange is shown below:

Initial key creation in handshaking process.

Text

Description automatically generated

Secret key computation.



There is also a webpage check each time the page is refreshed before returning the chat window. Using session cookies, the currently logged in user is verified to be user A in the chat window. If not, an error screen is returned.



To prevent against computation attacks, this key is then hashed using SHA-256 to create a byte array suitable for key generation through the Javascript crypto library.

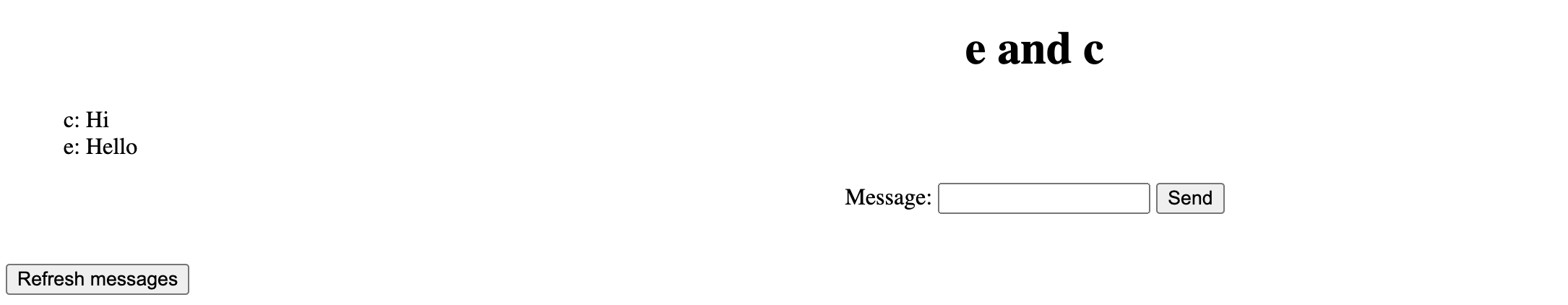
This key is used to encrypt messages client side before sending to the server, resulting in the server storing encrypted messages and the associated salted message authentication code. When these messages are passed to the client, they are decrypted before being displayed, checking the message authentication code in the process. If the message has been tampered with, its contents are redacted and the line [Modified Message] is shown in its place.

Text

Description automatically generated

Text

Description automatically generated



**Division of tasks:** Ali did all of the work while Abhishek did nothing, not attending class, completing work or responding to messages.