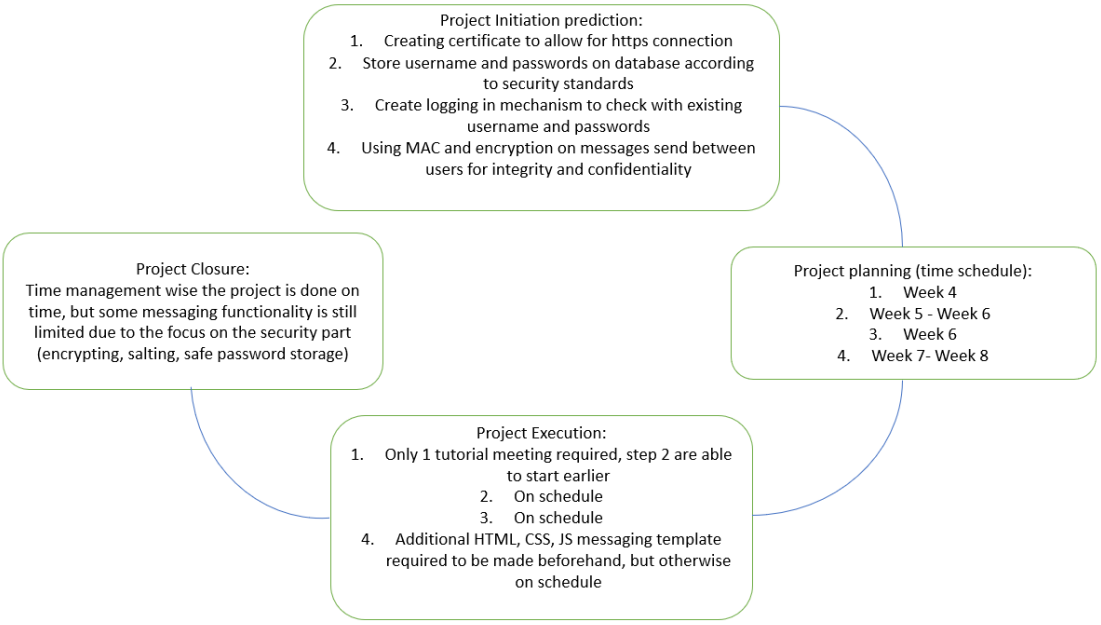
INFO2222 Report

Group name: CC08_Team7

Name: Fox Barancewicz, Wilson Husen

Summary



1. Username, password (encrypted with salt value) and randomly generated salt value, implemented by running password_storage.py and then storing it in storage.csv
2. Certificates are created manually with the certificate named as Assignment 1 and cert file located at /certificates/localhost.crt, used gunicorn as the server for integrating https to the website
3. Same certificate in (2) allows password to be stored securely through https.
4. Username and Password is checked for correctness when logging in by hashing the password and looping through the database to check if there is any match.
5. (Come back to this after done with message sending protocols)

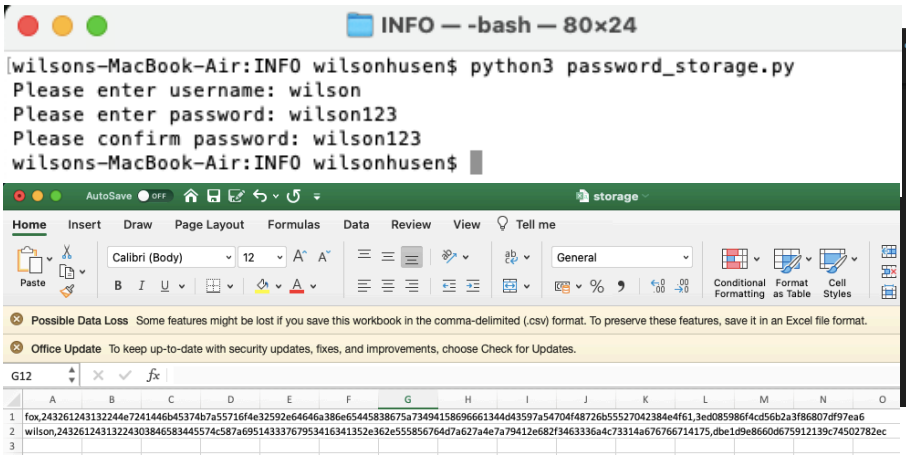
Member Contribution percentage:

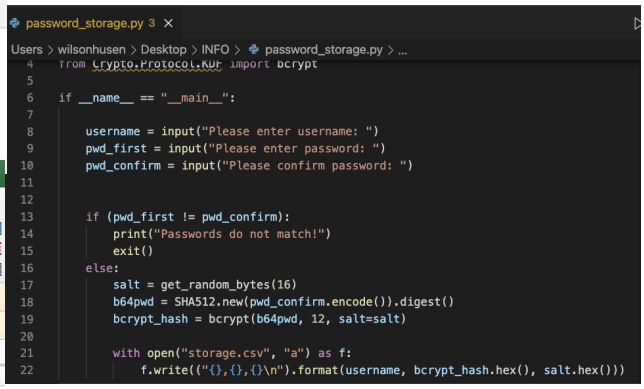
Tasks are mainly discussed and done together through git, refining the code progressively after each implementation of a function.

Fox: 50%, Wilson: 50%

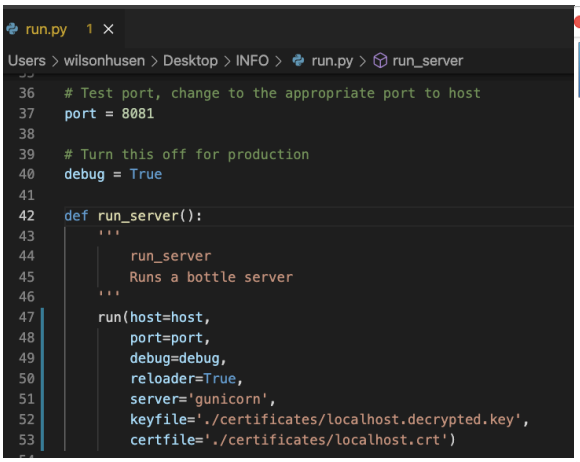
Body:

1. The screenshot below shows a snip of the code and commands executed in the terminal in order to store the passcode. After confirming whether the password and password confirmed matches, the password entered by the user will be stretched to a fixed size using SHA512 and it will be hashed together with a random salt to avoid attackers from matching common passwords in other database.

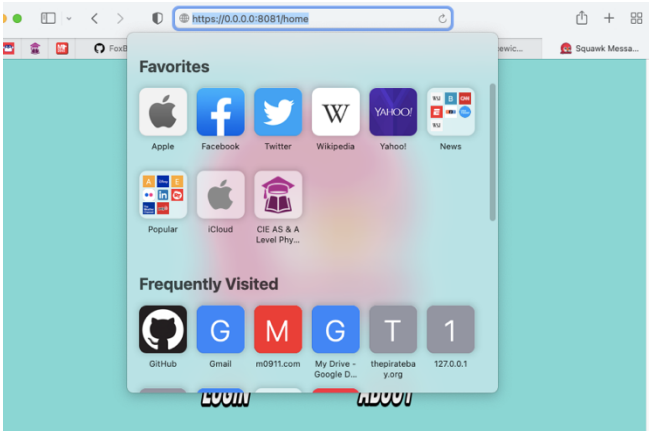




2. Server certificate is created through terminal and stored in file called localhost.crt, with the key used to verify the certificate stored in localhost.decrypted.key. Gunicorn is used to support the https connection directly to the server.



3. The certificate in (2) that allows https connection will give the user secure communication with the website, thus password will be safely transmitted to the server due to it being encrypted with TLS beforehand.



4. Password and username will be verified by comparing the username and designated password to the one in database, and as described in (1) salt will be used to defend against offline precomputation attacks.

5. When taking part in a two-way chat, the sender will encrypt messages with their private key (accessed through local dictionary variable, referenced with sender's authentication cookie. The receiver of said message must have previously logged in for the current instance of the messaging webapp (i.e. receiver must have a unique log-in every time the server restarts). If authenticated, the receiver will be able to access the encrypted messages through access-control to the messaging class for any chat they are involved in. For messages sent by the receiver, these are decrypted locally using the receiver's private key (accessed in same way as for the sender, see above), and for messages received from the sender, these are decrypted (also locally), however using the sender's private key, with the effectively being given access to the private key of the sender for the purpose of decrypting the messages. The receiver does not ever see the senders' private-key, nor will the sender ever be able to see the receiver's private key.

```python
for message in message_array:

    # Handles first 4 messages in message_array, which represent messages sent
    # to be decrypted using said users' key
    if message != '' and (count < 4):
        # Inits new fernet class using logged-in users' key
        fernet = Fernet(cookie_dict[cookie][1])
        new_message_arr.append(fernet.decrypt(message).decode())

    '''
    Access control: other users' key cannot be physically viewed, and can only
    the other party in the messaging class between two parties.
    '''
    # Handles last 4 messages in message_array, representing messages sent from
    # to be decrypted using the other users' key.
    if message != '' and (count >= 4):
        for user_cookie in cookie_dict:
            if cookie_dict[user_cookie][0] == send_to:
                fernet = Fernet(cookie_dict[user_cookie][1])
                new_message_arr.append(fernet.decrypt(message).decode())

    elif message == '':
        new_message_arr.append('')
```

Achievement beyond basic requirements:

- List of friends are created and stored in the website
- Main webpage and messaging interface design

Appendix:

- Example of messaging between two users: