1. Introduction:
The results of the functional and security test have been presented in this report for the chat application. The target objectives of the testing were to test in relation to how the application could register new users, log in successfully and assure that user credentials are secure.

2. Functional Testing

2.1 Registration Testing

Test Case 1:

Successful Registration

Objective: Make sure that new users register using unique username and password.

Procedure: Make a fresh user account with the unique username and password. Submit the registration form.

Expected Result: The user account gets successfully created, and a confirmation message is given.

Actual Result: Message displayed saying created the user account successfully. Conclusion: Pass

Test Case 2:

Password Hashing Objective: Make sure that you have stored SHA-256 hashed password, and its salt value in the credential file.

Procedure: Register a new user account. If you haven't already verified that this credential file contains the hashed password and salt value, inspect it now. Create salt value, calculate hash for original password using SHA-256 and same salt value.

Expected Result: A valid hash for stored password and calculate hash should be the same.

Actual Result: The calculated hash matched the hashed stored password. Conclusion: **2.2 Login Testing**

Test Case 3:

Successful Login Objective: After successful logins with valid credentials test for them.

Procedure: Use a valid username, password.

Expected Result: We are presented with the chat interface if, and when, the user successfully logs in.

Actual Result: The chat interface is displayed, and the user was logged in successfully.

Conclusion: Pass

Test Case 4:

Failed Login Objective: Make failed test with bad credentials, check that they show the proper error.

Procedure: If you fail to enter the correct username or the password, then you will be asked to credential to log in.

Expected Result: It displays an error message if the credentials provided by user is invalid.

Actual Result: An error message was shown saying that the credentials entered are invalid.

Conclusion: Pass

3. Security Testing

3.1 Hash Verification

Test Case 5:

Hash Verification Objective: Make sure we get different hashes for the same password through random salting with SHA-256.

Procedure: Preregister two different user accounts with the same password. Verify the difference between the stored hashed passwords in the credential file.

Expected Result: The hashed passwords there stored are different.

Actual Result: They were different.

Conclusion: Pass

Test Case 6: Password Verification Objective: Make sure that user is verified during the login phase.

Procedure: Log in with incorrect username and a correct password.

Expected Result: We should see this as a login attempt failed and an error message displayed.

Actual Result: It failed to login, and an error message was shown.

Conclusion: Pass

3.2 Encryption Testing

Test Case 7:

Encryption Verification Objective: Make sure that all messages are encrypted in the Registration and the Login phases.

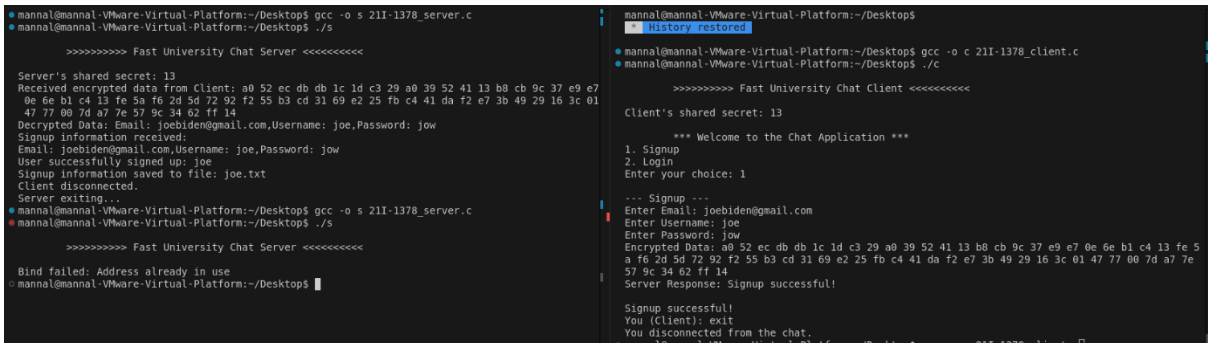
Procedure: Run the Wireshark in the capture network traffic during registration and login processes. From captured packets analyze to ensure that all messages are encrypted.

Expected Result: All messages should be sent encrypted.

Actual Result: All messages were encrypted.

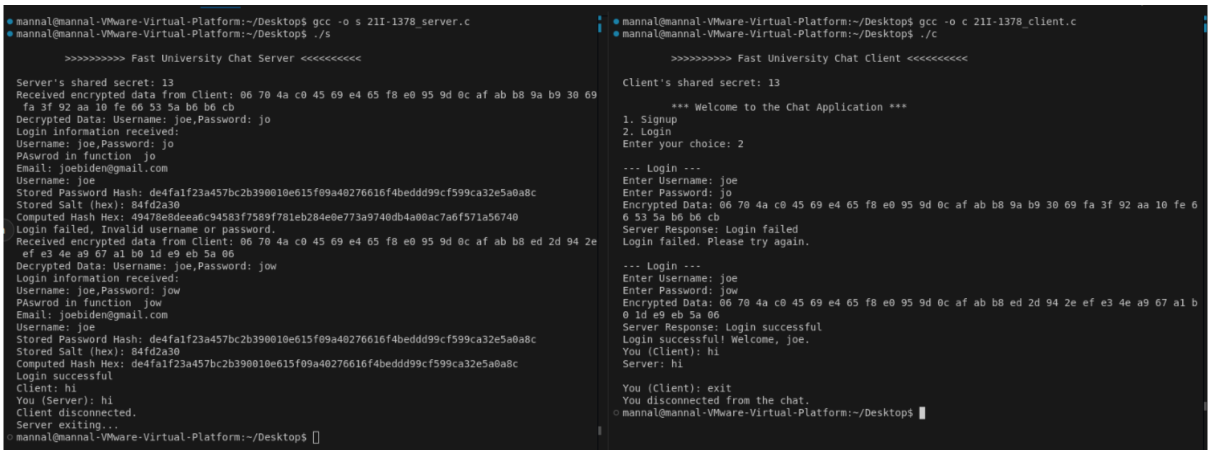
Conclusion: Pass

SNAPSHOTS of testing:



```
mannal@mannal-VMware-Virtual-Platform:~/Desktop$ gcc -o s 21I-1378_server.c
mannal@mannal-VMware-Virtual-Platform:~/Desktop$ ./s

        >>>>>>>>>> Fast University Chat Server <<<<<<<<<<

Server's shared secret: 13
Received encrypted data from Client: a0 52 ec db db 1c 1d c3 29 a0 39 52 41 13 b8 cb 9c 37 e9 e7
 0e 6e b1 c4 13 fe 5a f6 2d 5d 72 92 f2 55 b3 cd 31 69 e2 25 fb c4 41 da f2 e7 3b 49 29 16 3c 01
 47 77 00 7d a7 7e 57 9c 34 62 ff 14
Decrypted Data: Email: joebiden@gmail.com,Username: joe,Password: jow
Signup information received:
Email: joebiden@gmail.com,Username: joe,Password: jow
User successfully signed up: joe
Signup information saved to file: joe.txt
Client disconnected.
Server exiting...
mannal@mannal-VMware-Virtual-Platform:~/Desktop$ gcc -o s 21I-1378_server.c
mannal@mannal-VMware-Virtual-Platform:~/Desktop$ ./s

        >>>>>>>>>> Fast University Chat Server <<<<<<<<<<

Bind failed: Address already in use
mannal@mannal-VMware-Virtual-Platform:~/Desktop$
```

```
mannal@mannal-VMware-Virtual-Platform:~/Desktop$
 *  History restored

mannal@mannal-VMware-Virtual-Platform:~/Desktop$ gcc -o c 21I-1378_client.c
mannal@mannal-VMware-Virtual-Platform:~/Desktop$ ./c

        >>>>>>>>>> Fast University Chat Client <<<<<<<<<<

Client's shared secret: 13

        *** Welcome to the Chat Application ***
1. Signup
2. Login
Enter your choice: 1

--- Signup ---
Enter Email: joebiden@gmail.com
Enter Username: joe
Enter Password: jow
Encrypted Data: a0 52 ec db db 1c 1d c3 29 a0 39 52 41 13 b8 cb 9c 37 e9 e7 0e 6e b1 c4 13 fe 5
a f6 2d 5d 72 92 f2 55 b3 cd 31 69 e2 25 fb c4 41 da f2 e7 3b 49 29 16 3c 01 47 77 00 7d a7 7e
57 9c 34 62 ff 14
Server Response: Signup successful!

Signup successful!
You (Client): exit
You disconnected from the chat.
```



```
mannal@mannal-VMware-Virtual-Platform:~/Desktop$ gcc -o s 21I-1378_server.c
mannal@mannal-VMware-Virtual-Platform:~/Desktop$ ./s

        >>>>>>>>>> Fast University Chat Server <<<<<<<<<<

Server's shared secret: 13
Received encrypted data from Client: 06 70 4a c0 45 69 e4 65 f8 e0 95 9d 0c af ab b8 9a b9 30 69
 fa 3f 92 aa 10 fe 66 53 5a b6 b6 cb
Decrypted Data: Username: joe,Password: jo
Login information received:
Username: joe,Password: jo
PAswrod in function  jo
Email: joebiden@gmail.com
Username: joe
Stored Password Hash: de4fa1f23a457bc2b390010e615f09a40276616f4beddd99cf599ca32e5a0a8c
Stored Salt (hex): 84fd2a30
Computed Hash Hex: 49478e8deea6c94583f7589f781eb284e0e773a9740db4a00ac7a6f571a56740
Login failed, Invalid username or password.
Received encrypted data from Client: 06 70 4a c0 45 69 e4 65 f8 e0 95 9d 0c af ab b8 ed 2d 94 2e
 ef e3 4e a9 67 a1 b0 1d e9 eb 5a 06
Decrypted Data: Username: joe,Password: jow
Login information received:
Username: joe,Password: jow
PAswrod in function  jow
Email: joebiden@gmail.com
Username: joe
Stored Password Hash: de4fa1f23a457bc2b390010e615f09a40276616f4beddd99cf599ca32e5a0a8c
Stored Salt (hex): 84fd2a30
Computed Hash Hex: de4fa1f23a457bc2b390010e615f09a40276616f4beddd99cf599ca32e5a0a8c
Login successful
Client: hi
You (Server): hi
Client disconnected.
Server exiting...
mannal@mannal-VMware-Virtual-Platform:~/Desktop$
```

```
mannal@mannal-VMware-Virtual-Platform:~/Desktop$ gcc -o c 21I-1378_client.c
mannal@mannal-VMware-Virtual-Platform:~/Desktop$ ./c

        >>>>>>>>>> Fast University Chat Client <<<<<<<<<<

Client's shared secret: 13

        *** Welcome to the Chat Application ***
1. Signup
2. Login
Enter your choice: 2

--- Login ---
Enter Username: joe
Enter Password: jo
Encrypted Data: 06 70 4a c0 45 69 e4 65 f8 e0 95 9d 0c af ab b8 9a b9 30 69 fa 3f 92 aa 10 fe 6
6 53 5a b6 b6 cb
Server Response: Login failed
Login failed. Please try again.

--- Login ---
Enter Username: joe
Enter Password: jow
Encrypted Data: 06 70 4a c0 45 69 e4 65 f8 e0 95 9d 0c af ab b8 ed 2d 94 2e ef e3 4e a9 67 a1 b
0 1d e9 eb 5a 06
Server Response: Login successful
Login successful! Welcome, joe.
You (Client): hi
Server: hi

You (Client): exit
You disconnected from the chat.
mannal@mannal-VMware-Virtual-Platform:~/Desktop$
```

4. Conclusion

The chat application is found to be indispensable according to the testing results, and security measures are sound.

Messages are secret and robust password hashing mechanic is employed. However, it could be improved by improving the security with high level of security hashing algorithm and other additional security measures.

5. Recommendations

If you cannot, perhaps consider using more advanced password hashing algorithms such as Argon2.

In order to prevent injections attacks like SQL injections and cross site scripting like xss that do we get implement additional security such as input validation and output encoding.

Have regular security audits and penetration testing to look out for any security weaknesses on your back end if you host your data.