**Testcase 1 - Login incorrect password**  
*Instructions:*1. Create a server with a username-password combination

2. Create a client with that username

3. Attempt to login to the server with an incorrect password  
*Expected result:*

1. Connection does not succeed

2. A message is printed alerting the user that the password did not match the username provided

**Testcase 2 - Login correct password**  
*Instructions:*1. Create a server with a username-password combination

2. Create a client with that username

3. Attempt to login to the server with the correct password  
*Expected result:*

1. Connection succeeds, welcome message is displayed, and client may send messages to the server

**Testcase 3 - Login invalid username**  
*Instructions:*1. Create a server with a username-password combination file

2. Create a client with a username that isn’t in the file

3. Attempt to login to the server  
*Expected result:*

1. Connection does not succeed

2. A message is printed alerting the user that the username is not a member of that server

**Testcase 4 - Private chat insufficient arguments**  
*Instructions:*1. Create a server and a client connected to that server

2. Type ‘#private’  
*Expected result:*

1. Message displayed indicating that you must include a user id when sending private messages

**Testcase 5 - Private chat invalid user**  
*Instructions:*1. Create a server and a client connected to that server

2. Type ‘#private <invalid\_user\_id>’ with an invalid user id  
*Expected result:*

1. Message displayed indicating that <invalid\_user\_id> is not a valid user id to send messages to

**Testcase 6 - Private chat userid identical to sender**  
*Instructions:*1. Create a server and a client connected to that server

2. Type ‘#private <own\_user\_id> <message>’ with the user id of the client sending the message  
*Expected result:*

1. Message is printed for the client

**Testcase 7 - Private message to another user**  
*Instructions:*1. Create a server and a 3 clients connected to that server

2. Type ‘#private <user\_id> <message>’ with the user id of another user into one of the clients  
*Expected result:*

1. Message is printed for the client with <user\_id>

2. No message displayed for the third client

**Testcase 8 - Create channel**  
*Instructions:*1. Create a server and a client connected to that server

2. Type ‘#channel <name>’ into the client  
*Expected result:*

1. A message is printed that indicates the channel has been created

2. A message is printed that indicates the client has joined the channel

**Testcase 9 - Create channel channel already exists**  
*Instructions:*1. Create a server and a client connected to that server

2. Type ‘#channel <name>’ into the client

3. Type ‘#channel <name>’ into the client with the same name  
*Expected result:*

1. A message is printed that indicates the channel has already been created and the attempt to create the channel has failed

**Testcase 10 - Create insufficient arguments**  
*Instructions:*1. Create a server and a client connected to that server

2. Type ‘#channel’ into the client  
*Expected result:*

1. A message is printed that indicates the client needs to include a name for the new channel

**Testcase 11 - Join channel**  
*Instructions:*1. Create a server and a 2 clients connected to that server

2. Type ‘#channel <name>’ into one of the clients

3. Type ‘#join <name>’ into the other client  
*Expected result:*

1. A message is printed for the first client that indicates the channel has been created and the first client has joined

2. A message is printed for both clients that indicates the second client has joined

**Testcase 12 - Join channel channel does not exists**  
*Instructions:*1. Create a server and a client connected to that server

2. Type ‘#join <name>’ into the client  
*Expected result:*

1. A message is printed to the client that indicates the channel with that name does not exist so the client could not join the channel

**Testcase 13 - Send message to channel**  
*Instructions:*1. Create a server and 3 clients connected to that server

2. Create a channel with the first client

3. Join the channel with the second client

4. Type ‘#send <channel> <message>’ into the first client  
*Expected result:*

1. The message is printed to the first and second clients with the channel name and user id of the first client before the message

3. The message is not printed for the third client

**Testcase 14 - Send message insufficient arguments**  
*Instructions:*1. Create a server and a client connected to that server

2. Create a channel with the client

3. Type ‘#send’ into the client  
*Expected result:*

1. An error message is displayed to the client indicating that they need to supply a channel to send their message to

**Testcase 15 - Send message to channel channel does not exist**  
*Instructions:*1. Create a server and a client connected to that server

2. Create a channel with the client

3. Type ‘#send <wrong\_channel> <message>’ into the client with a channel name that does not exist  
*Expected result:*

1. A message is displayed indicating that the channel does not exist and the message could not be sent

**Testcase 16 - Send message to channel when not a member of the channel**  
*Instructions:*1. Create a server and 2 clients connected to that server

2. Create a channel with the first client

3. Type ‘#send <channel> <message>’ into the second client  
*Expected result:*

1. A message is displayed for the second client indicating that the client must join the channel before they can send messages to the channel

**Testcase 17 - Leave channel insufficient arguments**  
*Instructions:*1. Create a server and a client connected to that server

2. Create a channel

3. Type ‘#leave’ into the client  
*Expected result:*

1. A message is displayed for the second client indicating that the client must include a channel to leave

**Testcase 18 - Leave channel channel does not exist**  
*Instructions:*1. Create a server and a client connected to that server

2. Type ‘#leave <channel>’ into the client  
*Expected result:*

1. A message is displayed for the second client indicating that the channel the client is attempting to leave does not exist

**Testcase 19 - Leave channel when not a member**  
*Instructions:*1. Create a server and 2 clients connected to that server

2. Create a channel with the first client

3. Type ‘#leave <channel>’ into the second client  
*Expected result:*

1. A message is displayed for the second client indicating that they cannot leave a channel they have not joined

**Testcase 20 - Availability set unavailable**  
*Instructions:*1. Create a server and a client connected to that server

2. Type ‘#unavailable’ into the client  
*Expected result:*

1. A message is displayed for the client indicating that they are now unavailable

**Testcase 21 - Availability set unavailable when already unavailable**  
*Instructions:*1. Create a server and a client connected to that server

2. Type ‘#unavailable’ into the client

3. Type ‘#unavailable’ into the client again  
*Expected result:*

1. A message is displayed for the client indicating that they are now unavailable after the first command

2. A message is displayed for the client indicating that they are already unavailable after the second command

**Testcase 22 - Availability set available**  
*Instructions:*1. Create a server and a client connected to that server

2. Type ‘#unavailable’ into the client

3. Type ‘#available’ into the client  
*Expected result:*

1. A message is displayed for the client indicating that they are now unavailable after the first command

2. A message is displayed for the client indicating that they are now available after the second command

**Testcase 23 - Availability set available already available**  
*Instructions:*1. Create a server and a client connected to that server

2. Type ‘#available’ into the client  
*Expected result:*

1. A message is displayed for the client indicating that they are already available

**Testcase 24 - Availability set idle**  
*Instructions:*1. Create a server and a client connected to that server

2. Wait five minutes  
*Expected result:*

1. A message is displayed for the client indicating that they are now idle because it has been five minutes since they interacted with the client

**Testcase 25 - Check status of user insufficient arguments**  
*Instructions:*1. Create a server and a client connected to that server

2. Type ‘#status’ into the client  
*Expected result:*

1. A message is displayed for the client indicating that they need to provide a user id when looking for a status

**Testcase 26 - Check status of user that does not exist**  
*Instructions:*1. Create a server and a client connected to that server

2. Type ‘#available <user\_id>’ into the client with a user id that does not exist on the server  
*Expected result:*

1. A message is displayed for the client indicating that the user id they provided does not exist

**Testcase 27 - Check status of user available**  
*Instructions:*1. Create a server and 2 clients connected to that server

2. Type ‘#status <user\_id>’ into the first client with the user id of the second client  
*Expected result:*

1. A message is displayed for the first client indicating that the second client is available

**Testcase 28 - Check status of user idle**  
*Instructions:*1. Create a server and 2 clients connected to that server

2. Don’t do anything on the second client for five minutes

3. Type ‘#status <user\_id>’ into the first client with the user id of the second client  
*Expected result:*

1. A message is displayed for the first client indicating that the second client is idle

**Testcase 29 - Check status of user unavailable**  
*Instructions:*1. Create a server and 2 clients connected to that server

2. Type ‘#unavailable’ into the second client

3. Type ‘#status <user\_id>’ into the first client with the user id of the second client  
*Expected result:*

1. A message is displayed for the first client indicating that the second client is unavailable

**Testcase 30 - Check status of user offline**  
*Instructions:*1. Create a server and a client connected to that server with two user ids

2. Type ‘#status <user\_id>’ into the client with the user id of the other user on the server  
*Expected result:*

1. A message is displayed for the first client indicating that the second client is offline