### Use Case Diagrams:

**Figure 1.1 (Login):** This diagram shows the login process. It assumes that the user is valid and will use their credentials to log in. The credentials introduced by the user are then validated by the database.

**Figure 1.2 (Send Message):** A user attempts to send a message, the system will first need to validate their connection to ensure that the target user is online and connected to the broker.

* If the user is online:
  + The source message is sent, and the source chat history is updated
  + The target user is notified, and the target user’s chat history is also updated
  + The chat histories on both ends are updated within the database
* If the user is offline:
  + The source message is stored within the database temporarily
  + The source user’s connection is then repeatedly checked until they are confirmed to be online
  + Once the source user is online, their message is then sent to the target user, and the following stages mentioned above occur

**Activity Diagrams:**

**Figure 2.1 (Login):** In this scenario, the User attempts to login into the application by inputting the User credentials (i.e. Username and Password). The credentials are then authenticated against the existing credentials within the database.

* If the password is valid, the user gets access to their user space
* If the password is invalid, the user is notified of their credentials being incorrect and are asked to try again.

**Figure 2.2 (Send Message):** The activity diagram covers the flow of events required to send a message. The user triggers an event when sending a new message using the application. The source user’s status is validated to ensure they are connected to the broker. If the source user is offline, the message is temporarily stored in the database. The application keeps checking the status of the users every 10 seconds. Once the source user's connection is established as being online, the system retrieves any messages that are stored in the database. A queue of messages to be sent is created and messages are ordered accordingly in a first in, first out fashion. A fork then occurs for several activities to run concurrently, such as notifying the target user that a new message has been received, updating chat logs, and updating the chat history itself. Once these are all completed, they merge back to end the 'send message' activity.