# Chat project

## TTM4100

Group 66:
Damir Anicic,
Leif Halvor Sunde,
Christian Fredrik Sætre,
Jostein Borgen Moe,
Nicolay Erlbeck

## *Textual description*

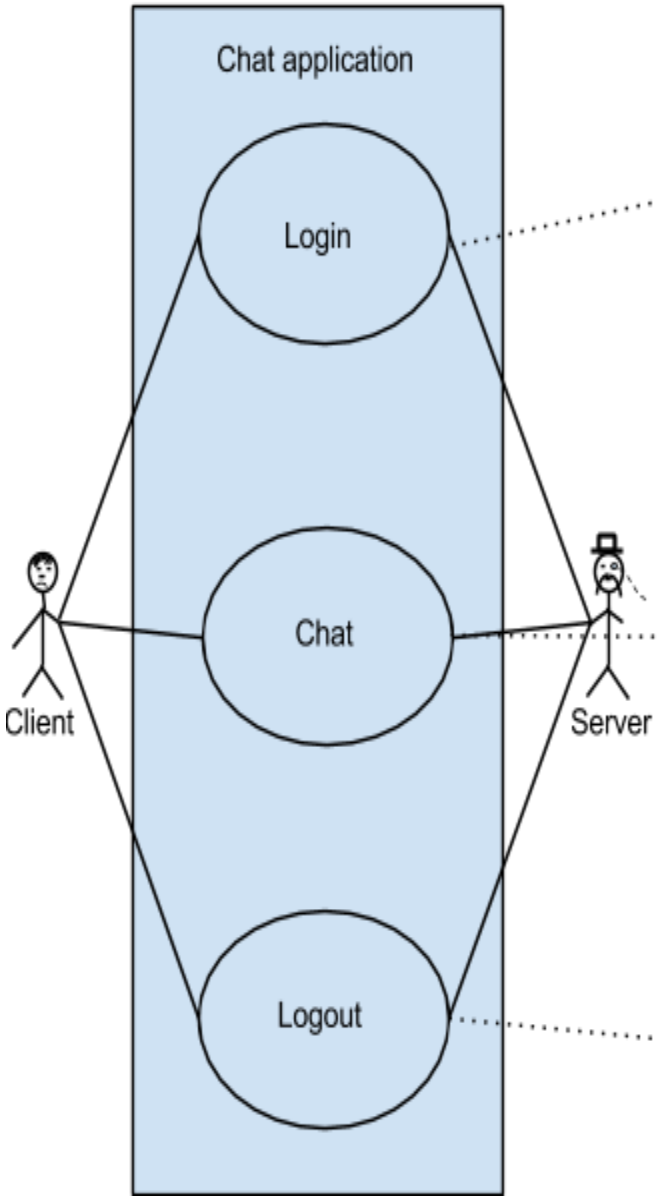
The system is divided into three modules, consisting of the classes and functions for the client, server and message worker. We have chosen to divide the system into three use-cases. First, we have the *login* case, where the user requests to login to the server, and the server responds with the correct response; Second, we have the chat case, where the client requests to send a message and the server sends the message out to all clients logged on; Third, we have the logout case, where the client requests to log out and the server responds with the correct response.

All these cases can have success-scenarios and extensions, as described in the use case diagram. The different scenarios depends on whether the client is in the correct state, that is logged in or not.

Our system is further divided into three layers: network, parsing and application. The network layer contains the protocol we are using to communicate between server and client. The parsing layer consists of the coding and decoding, JSON in our case, of the messages sent over the network layer protocol. The application layer consists of the general system functionality, taking care of user input and output to screen.

We have made 2 clients, one in Go and one in Python, where the python client uses the receiveMessageWorker to both receive and send message simultaneously (this is simpler in Go as we can use gorutines, thus the reason for two clients). The server is written in Python, like the client in python, it is heavily influenced by the skeleton code.

## Use cases

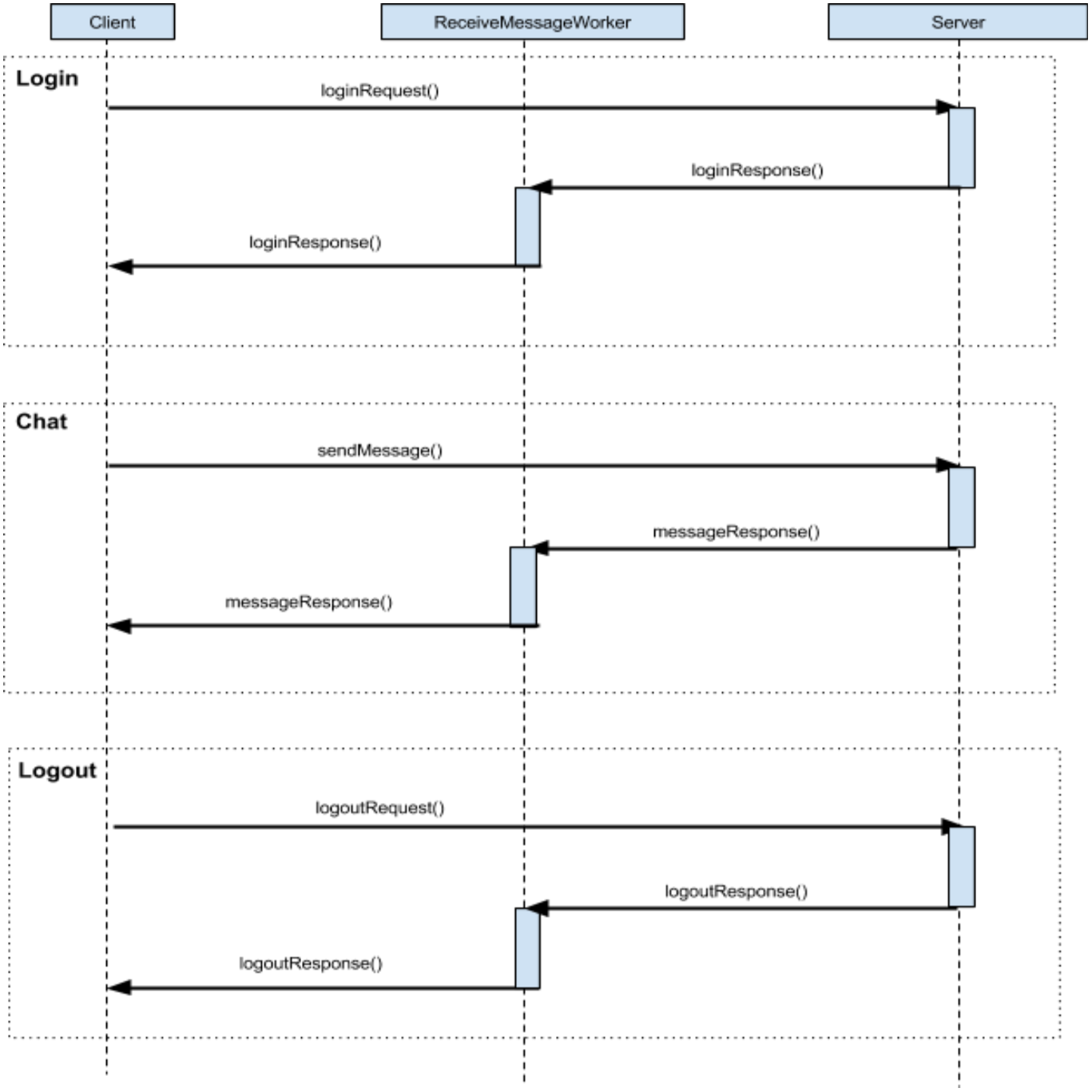



Precondition: Client connected to the same network as server.
Trigger: Client send login request to server with a username
Success scenario:
1. Server returns username and all messages from the session so far
2. Server adds client's username to online list
3. Server broadcasts username to clients on online list
Extensions:
1.a Server sends "invalid username" response to client
1.b Server sends "username taken" response to client

Preconditions: Client connected to the same network as server.
Trigger: Client requests to send a message to server
Success scenario:
1. Server sends the message to all clients on online list
Extensions:
1.a Server responds to client that he is not logged in

Preconditions: Client connected to the same network as server.
Trigger: Client sends logout request to server
Success scenario:
1. Server responds with confirmation of logout and username.
2. Server removes client from online list
Extensions:
1.a Server responds that the client is not logged on.

## Sequence diagram



## Class diagram