Chat-Pack
System Specification

Stephan Do
Nico Vasquez
Thomas Antensteiner

HTBLA LEONDING

Content

1. Initial Situation and Goals

1.1. Initial Situation

Nowadays there are already a lot of chat-clients which are used regularly for example TeamSpeak, Skype, etc. People use it for different situation for example for a relaxed conversation with a family member or friend abroad or for a serious discussion in a project meeting. But they got many problems.
This is why we want to create a new client. This project will be focused on the user's comfortableness. They should not be confused or lose their patience over a client.

They should enjoy using this client and be happy about it. To succeed, we want to make the client simple and resource limited in order to let the user experience a comfortable conversation with his friend/s.

Every user has and can use all features (no cost). A special feature will also be available and that will be the stats for each friend.

This option allowed the user to see the information between user and friend for example, messages sent in total, messages in last week. This should also prevent the user to forget about a friend.

Every account will get exactly one tag-number. The purpose of the tag-number will be the facilitation for searching a friend.

1.1.1. Application Domain

Everyone has friends and if the person is far away and he/she still wants to talk to them, he/she needs to find way to overcome the distance. For Example, when someone meets a friend in his/her holiday and he/she still wants to be in touch with the friend, even after the holiday, then a communicate-application is needed.
Our goal is to create such an application. Everyone of our users has their friends in his/her friend-list and is able to chat with them and see info about them, like what they are playing/doing currently.

1.1.2. Glossary

- **User**:
  The user is the person who starts the program and logs in, in order to use it.

- **Friend list**:
  The friend list is a list that contains all the friends of the user.

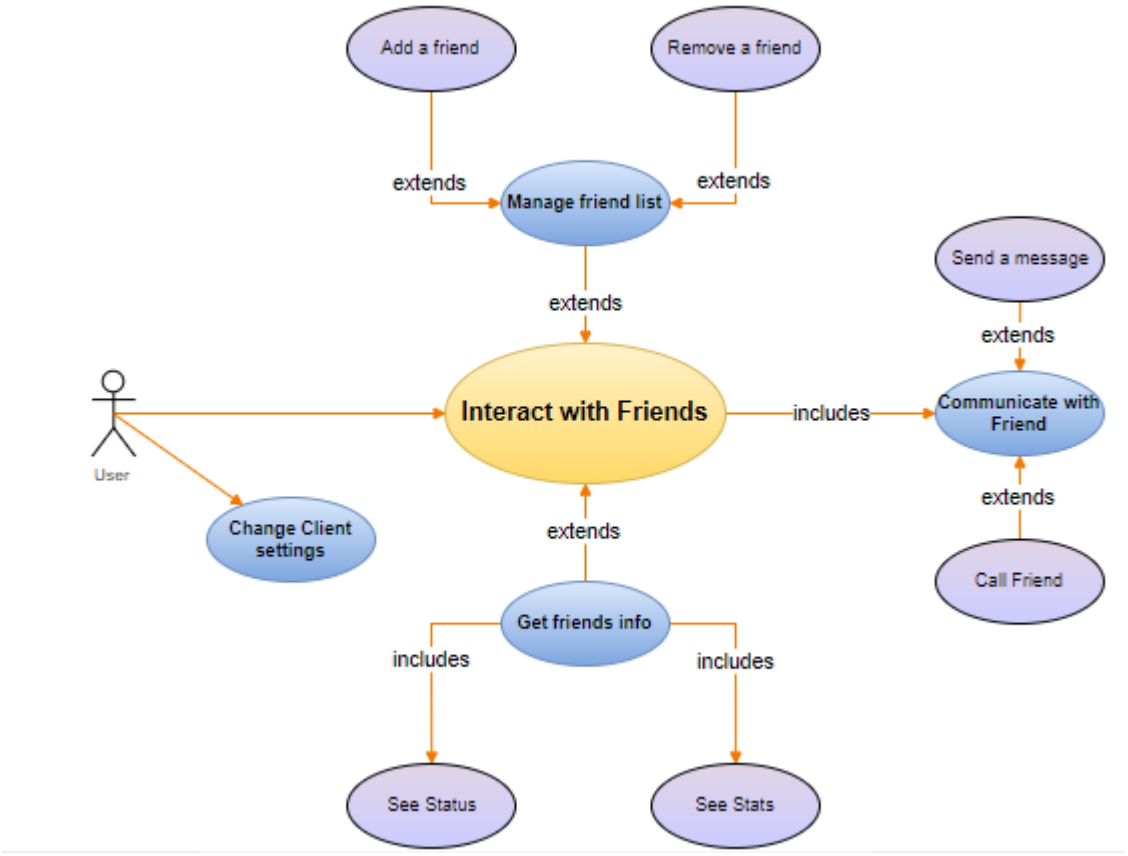
- **Friend**:
  A friend is another user. In order to become friends with someone, a friend-request has to be sent.
  The user can either accept the friend-request or decline it. If the friend-request is accepted, the person who sent the request will be added to the friends list.

1.2. Goal Definition

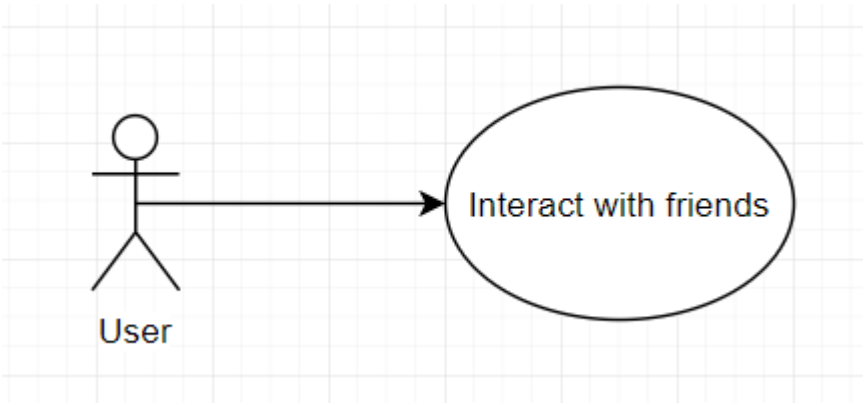
The goal of our program Chat-Pack is to create a simple, minimalistic chat application where anyone can chat with another user without paying anything. Also it is important for us to make the overlay so clear that everyone is able to deal with the program without confusion quickly. Another point we are aiming for, is that the user can individualize the interface by his own liking.
The program is going to be published as of a WPF program.

2. Functional Requirements

2.1. Use Case Diagrams



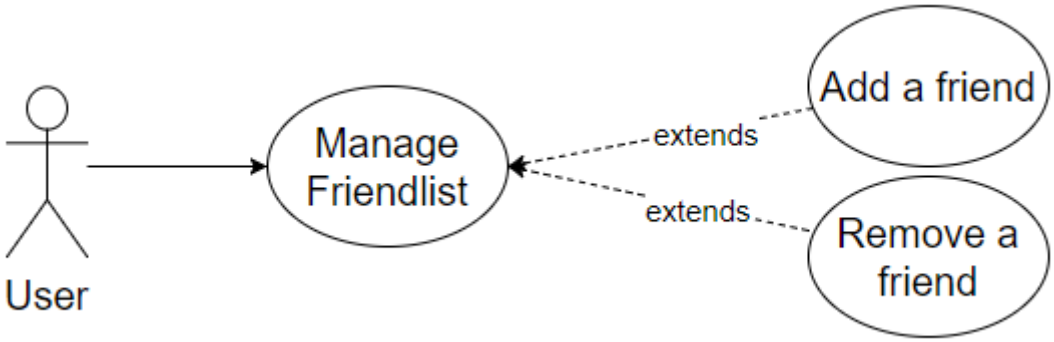
2.2. Interact with friends



2.2.1. Characteristic Information

Goal:	The user should be able to have contact with his/her friend/s
Precondition:	The user want to hang out with a friend with an application for a quick way to chat or to talk about something.
Postcondition:	The user talked or messaged with his/her friend/s.
Involved User:	The user and a friend
Triggering Event:	The user want to talk or chat with a friend via an application.

2.3. Manage friend list

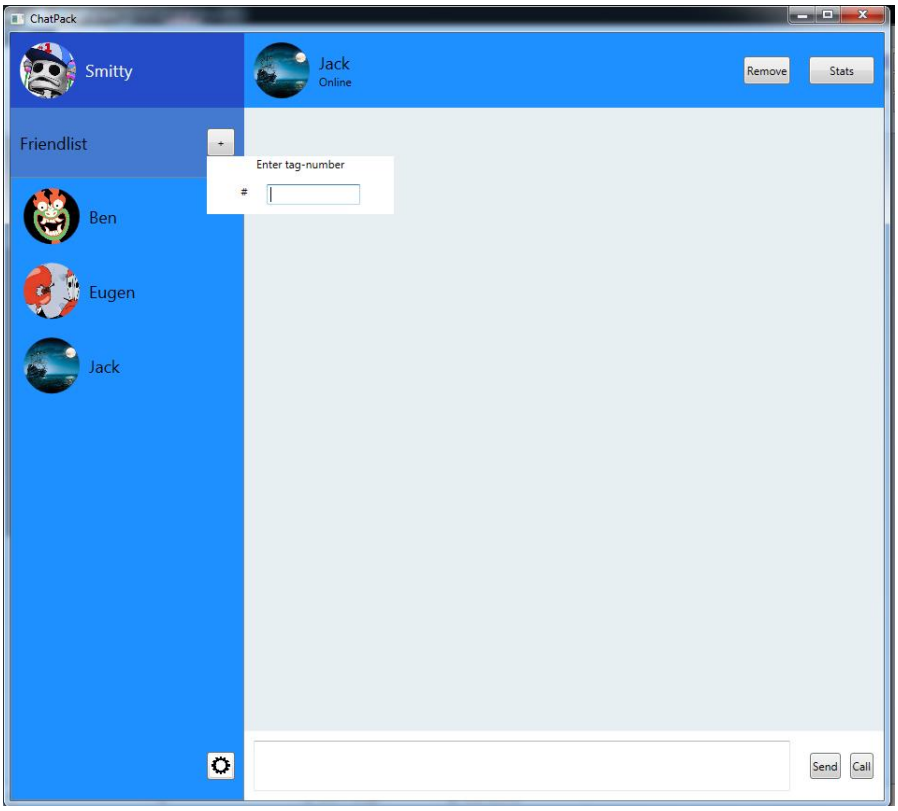


2.3.1. Characteristic Information

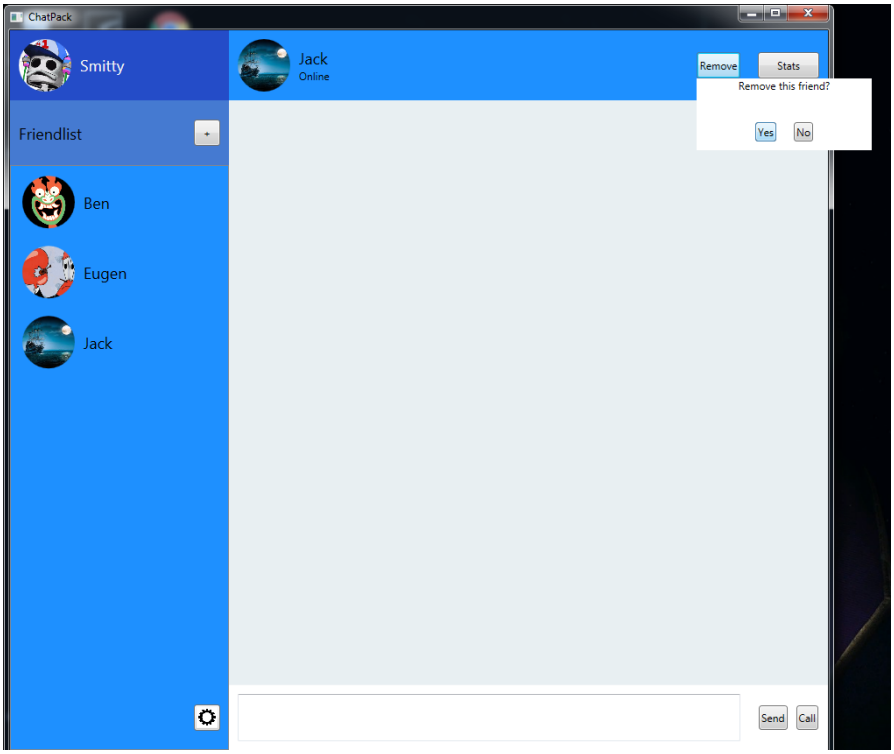
Goal:	The user is able to add new friends or to remove current friends.
Precondition:	In order to add a friend the user needs to know the friend tag-number to add him/her. To remove a friend he/she has to be in the friend list.
Postcondition:	The user added a friend or he/she removed an existent friend.
Involved User:	The user and the friends
Triggering Event:	The user wants to have a new friend or he/she wants to remove an existent friend.

This use case is divided by two sub use cases called “Add a friend” and “Remove a friend”

2.3.2. GUI to call the use case “Add Friend”



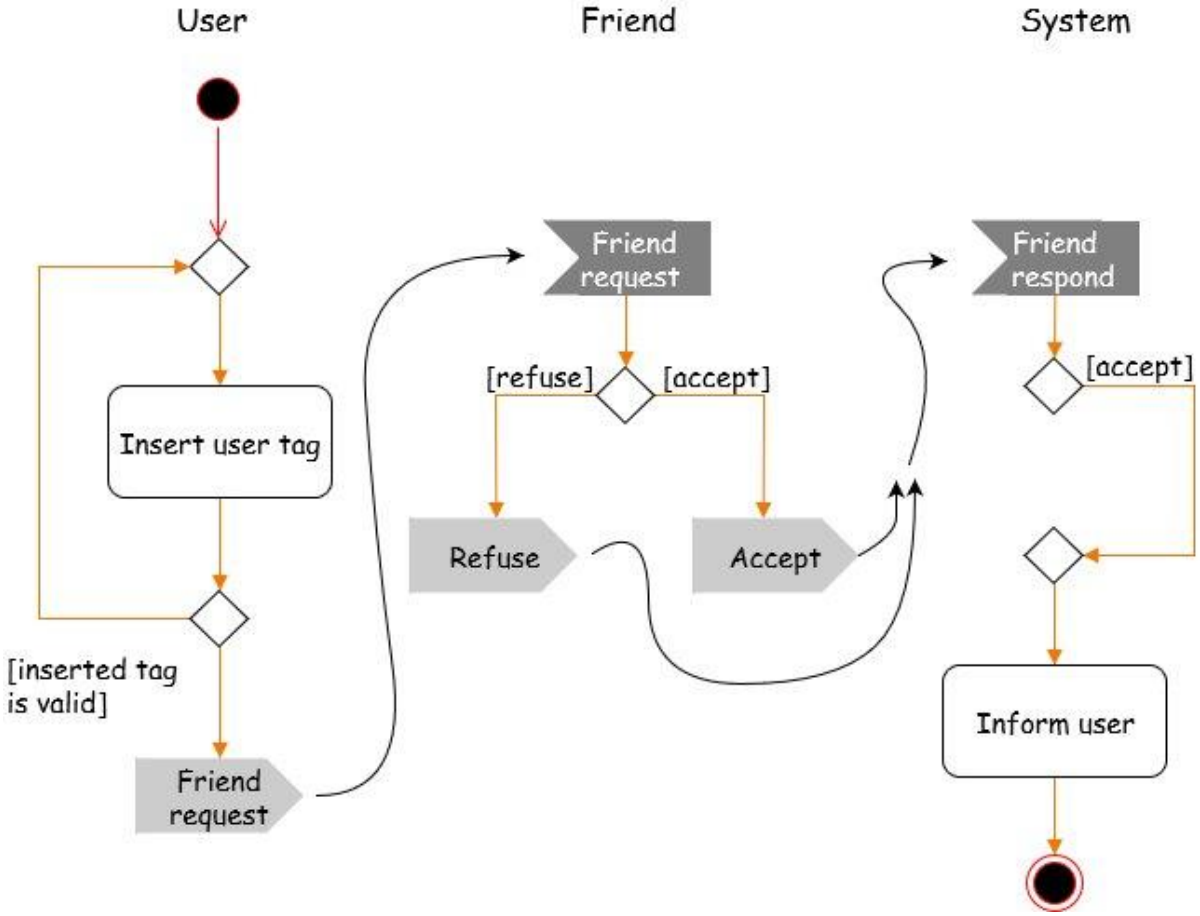
2.3.3. GUI to call the use case “Remove Friend”



2.3.4. Scenarios for the standard use

Step	User	Activity
1	User	Log in
2	User	Click the “+”-button to add a friend
3	User	Type the tag-number
4	User	Select a friend
5	User	Click the “Remove”-button to remove the selected friend
6	User	Confirm the removal

2.3.5. Workflow

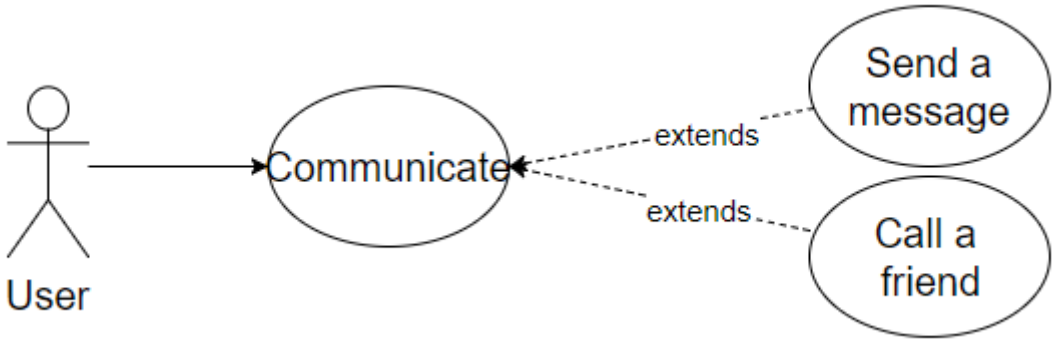


Open Points

Adding:

The person who gets a friend-request can decline it. Although this person decline the request, he/she can get another request by the same user.

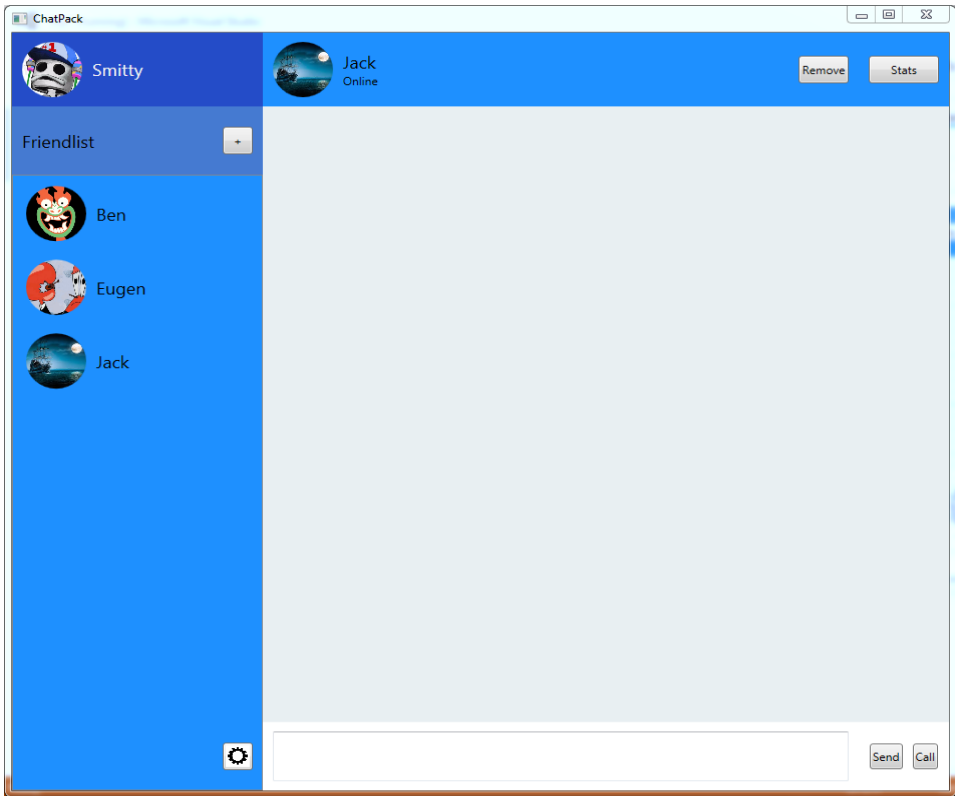
2.4. Communicate with Friend



2.4.1. Characteristic Information

Goal:	The user communicate with friends in term of sending messages or talking.
Precondition:	The user has to have at least one friend to be able to communicate with. And for the calling he/she needs to accept the call.
Postcondition:	The user communicated with a friend.
Involved User:	The user and one or more of his/her friends
Triggering Event:	The user wants to call or to chat someone.

2.4.2. GUI to call the use case



2.4.3. Scenarios for the standard use

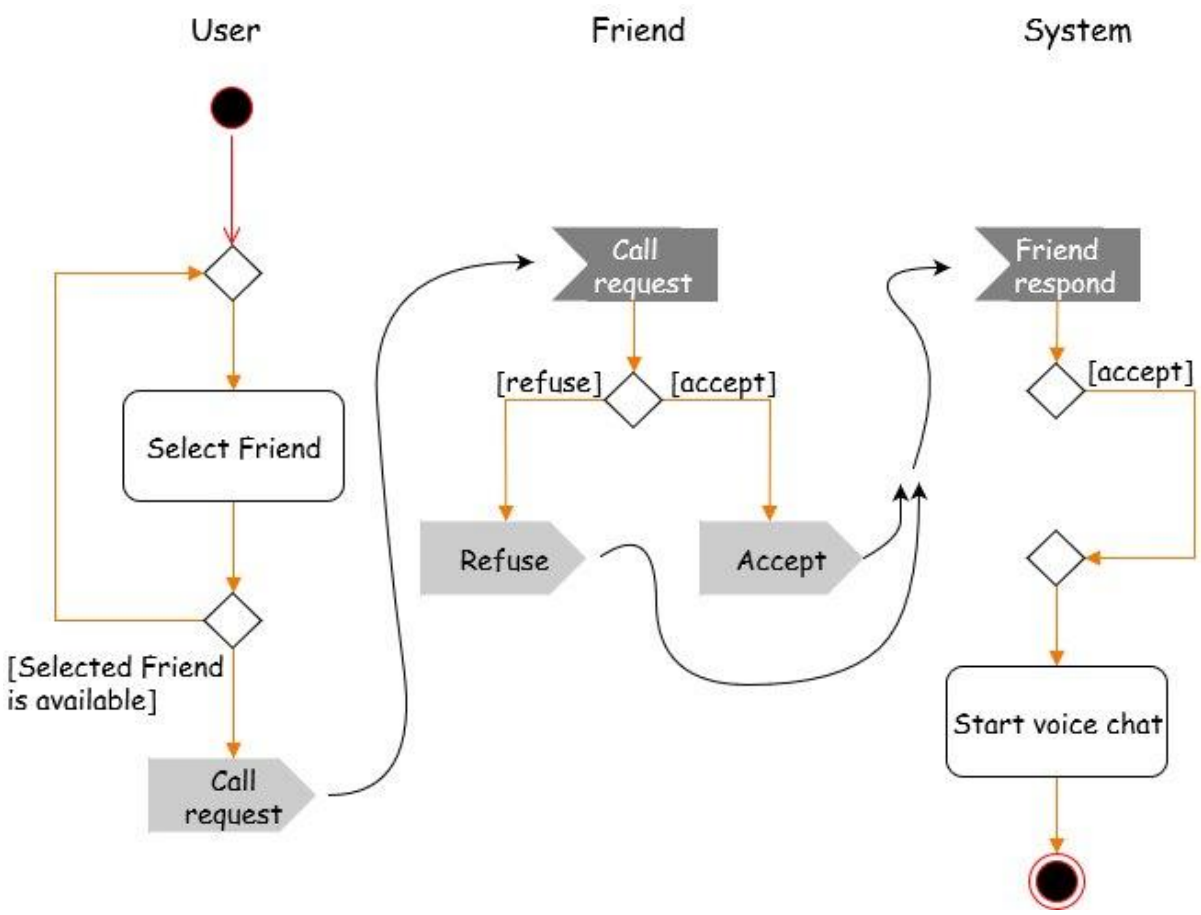
Step	User	Activity
1	User	Log in
2	User	Select friend
3	User	Call the selected friend

Or

3	User	Write a message
4	User	Send the message

2.4.4. Workflow

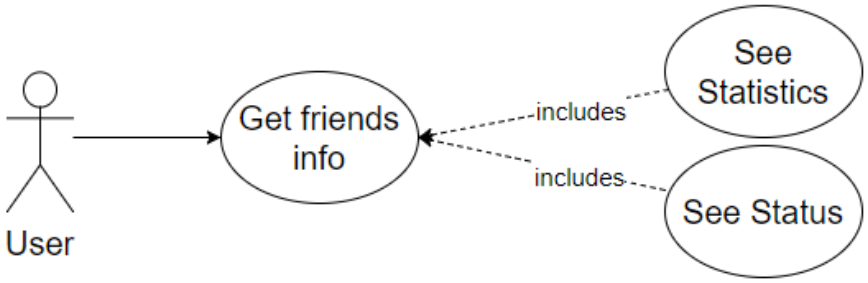
Calling a friend



2.4.5. Open Points

- Characters:
  The application will not support every characters and because of that some messages will be shown differently (e.g. Japanese symbols, Russian characters, etc.).

2.5. Get friends info

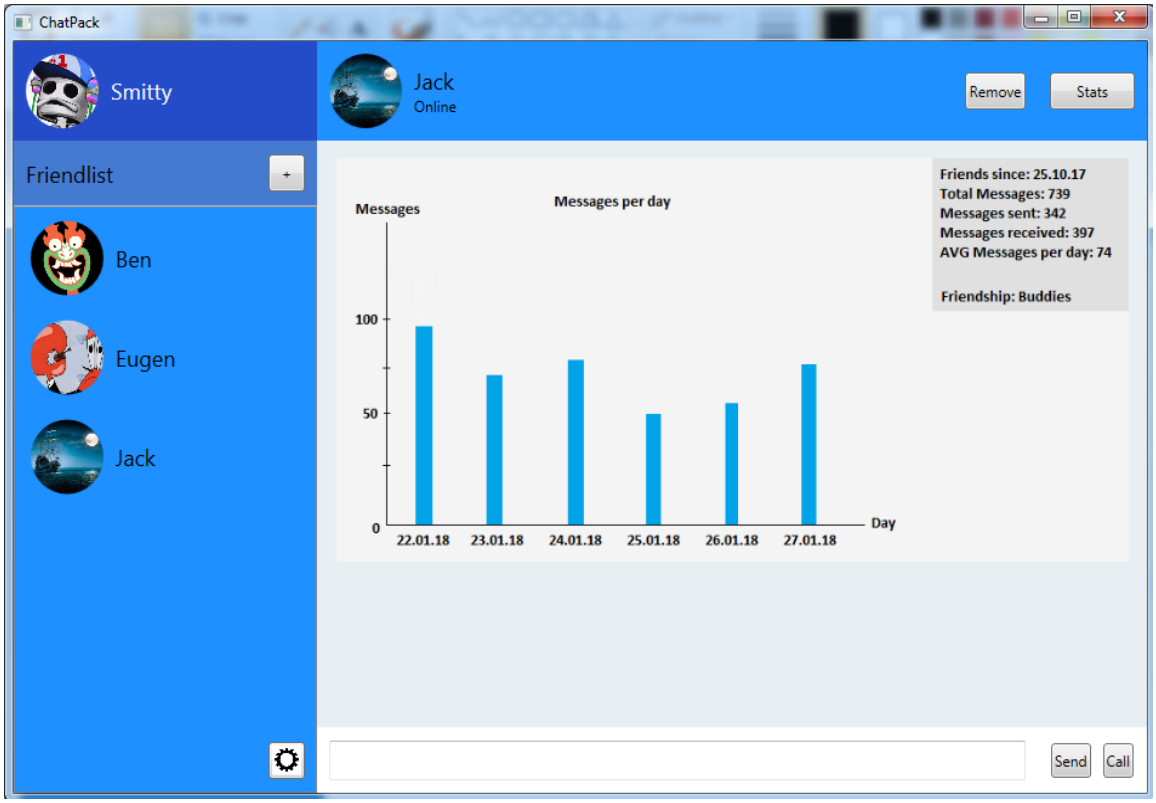


2.5.1. Characteristic Information

Goal:	The user can see different information about a friend. It can be divided in stats (friend current activity) and statistics (total messages sent, friends since, etc.).
Precondition:	The user have to have at least one friend in the friend list.
Postcondition:	The user saw and know about a friend current information and his/her recent activity.
Involved User:	The user and one friend
Triggering Event:	When the user is curious about what his/her friend is doing right now. When the user wonders and wants to see how well their friendship is going.

This use case has two sub use cases, called “See Stats” and “See Statistics”

2.5.2. GUI to call the use case



2.5.3. Scenarios for the standard use

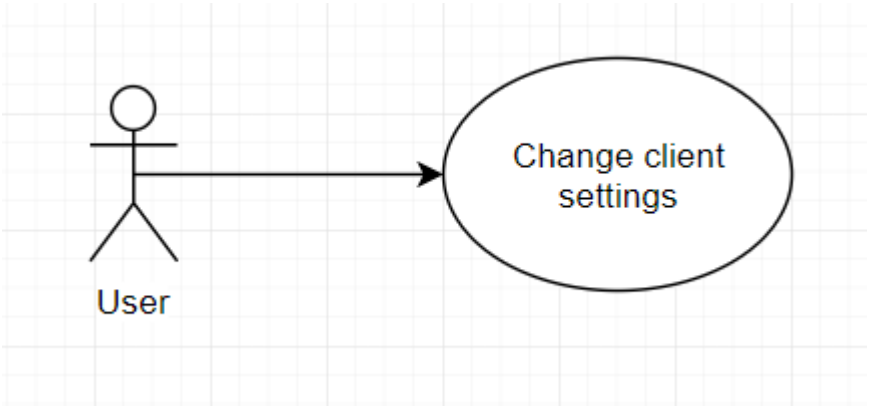
Step	User	Activity
1	User	Log in
2	User	Select a friend in the friend list
3	User	Look at the selected friend to see the friend's current activity (the status is online/ offline if the friend does nothing)
4	User	Click the "Stats" button at the right top to show the stats

2.5.4. Open Points

- Unfriending
  If the user accidentally remove a friend and send a friend-request to him/her again, the previous information will be lost.

- Spamming
  The user and the friend can fake the information with unnecessary messages in form of spamming.

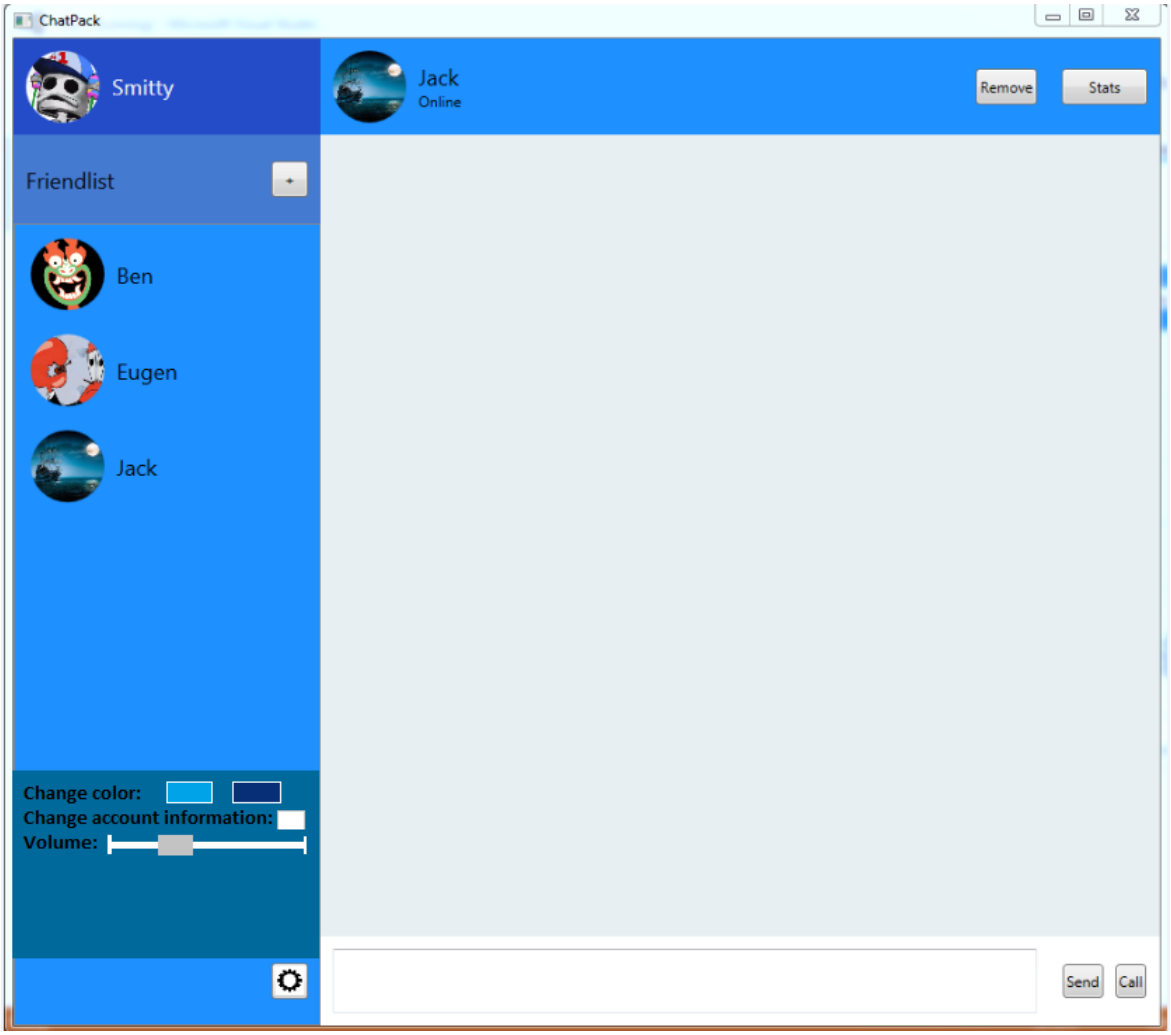
2.6. Change client settings



2.6.1. Characteristic Information

Goal:	The user is able to change the color theme of the interface. He/ She is also able to change his user data like his/ her username.
Precondition:	The user has to log in (have an account).
Postcondition:	The client changed his information and the interface color after his liking.
Involved User:	The user
Triggering Event:	The user is not satisfied with the color or he/she made a mistake about his/her information.

2.6.2. GUI to call the use case



2.6.3. Scenarios for the standard use

Step	User	Activity
1	User	Log in
2	User	Click the settings-button (gear logo)
3	User	Change the color, the volume or click the button to change the user information

3. Non-functional Requirements

3.1. Usability

The application should have a low memory usage so that it can run in the background while working/playing videogames without delays/lags easily.
Also, the interface should be so clear that the user can find everything he needs at his/her first glance.
Another important point is that the user should be able to individualize the overlay as he likes it.
The program should deliver messages at least under 5 seconds without any delay so that, if the user wants to, he can respond immediately.

3.2. Efficiency

As already mentioned in the Usability paragraph, the program should deliver messages in under 5 seconds and also, as mentioned in the previous paragraph, the program should use less memory than other application like Skype or Discord (We are trying to reach the goal of under 100 MB).
Finally, it should open in under 10 seconds so that the user can start communicating as soon as possible.

3.3. Maintainability

If we notice some bug, we will fix them as soon as possible.
Also we will add little features that the users ask for, if they are useful or appropriate.

3.4. Security

The private information of the user like the password or the chat history of each friend will be trusted to us and they will not be will be passed on to anyone.

3.5. Legal constraints

As far as the project goes, we have not find any standards we need to respect.

4. Quantity Structure

Profile

We are going to save our profiles which use about 1KB on the server. Every profile is going to have a tag, the current IP-Address a username and password.
Also each profile is going to have a profile picture which also is saved on the server and takes about 1 MB.

Chat history

The chat history is going to be saved as a txt file in the local Folder witch takes about 2KB.

Conclusion

So if we are talking about 10 000 users we need a little bit more then 10GB save space.

5. System Architecture and Interface

The application is going to be the user-interface. And in the background the application will send/receive messages via UDP to the server where all users with password and ID are saved. The chat history is going to be saved with the application (local).

6. Acceptance Criteria

6.1. Manage friend-list

functionality	Expected result
Send a friend-request	The presumed future friend gets a friend-request
Accept friend-request	The friend's profile is added to the user's friend list and the user to the friend's friend list
Decline friend-request	The user who sent the friend-request gets a message, that the request was declined
Remove friend	The profile is removed from the friend list

6.2. Communicate with friends

functionality	Expected result
Send a message	The user gets the message on his screen
Start a call	The called user is getting a call-request
Accept call-request	The two users are in a conference where they can talk
Ignore call-request	The user who started the call gets a decline-message

6.3. Get friends info

functionality	Expected result
Press the "stats" button	The user is able to see the stats between him/her and one of his/her friend
Look under the profile	See what your friend is currently doing