POO/COO Project report
Chat System Application

Kouki Sarra and Farhat Salma

4 IR B1

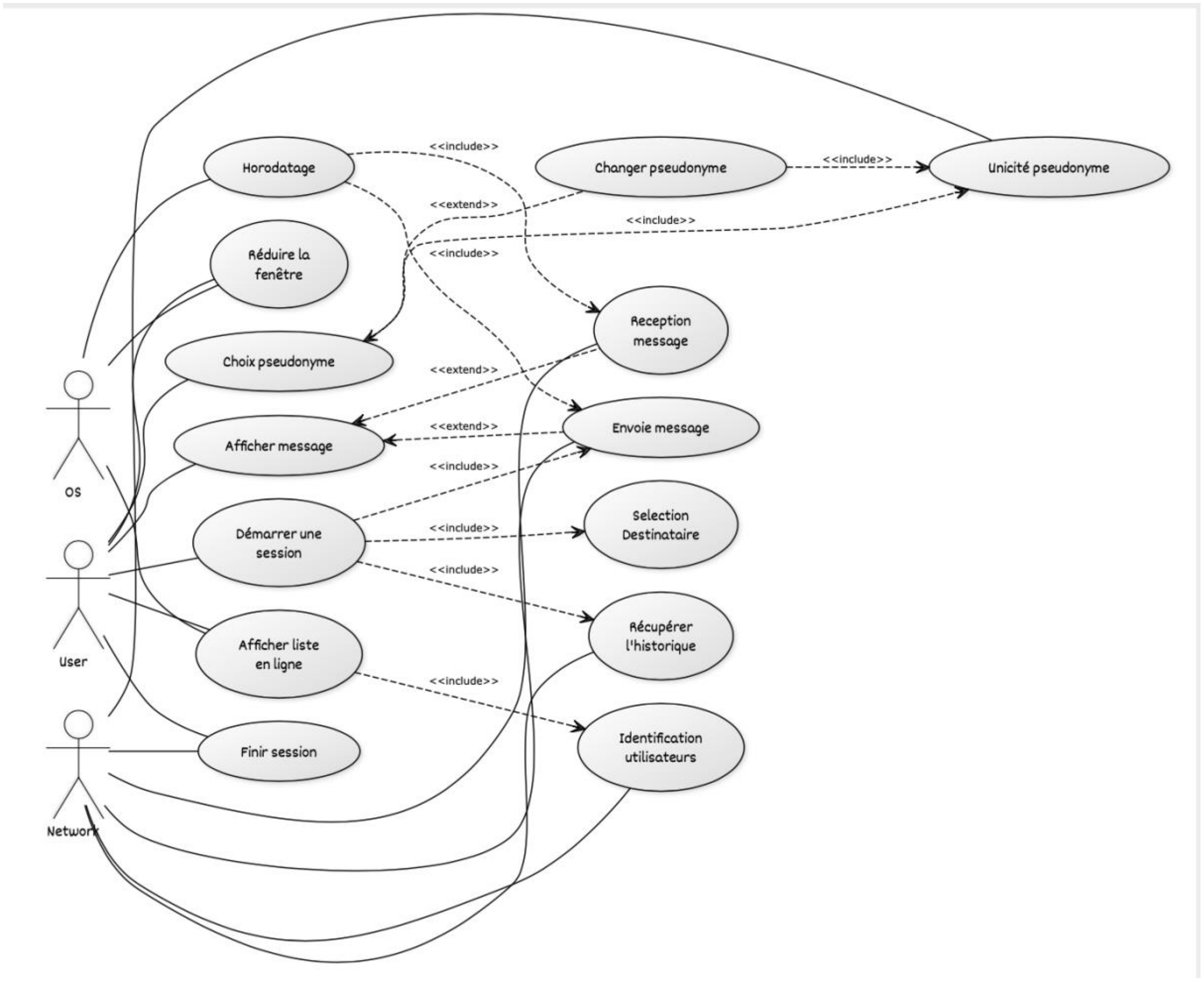
Introduction:

As part of the POO/COO course we were asked to implement a distributed chat system. This system allows users to communicate by sending and receiving messages or files using interconnected devices. In this report we will trace the progress of our project, from its design to its implementation. We will also present the tools that we used to manage the project.

Therefore, we will divide this report into three parts:
1- **Object oriented design**: We will present the UML diagrams
2- **Object oriented programming**: We will explain how our chat system works and how we implemented it
3- **Project Management**: We will mention the tools used within the framework of the Agile method

I. Object oriented design:

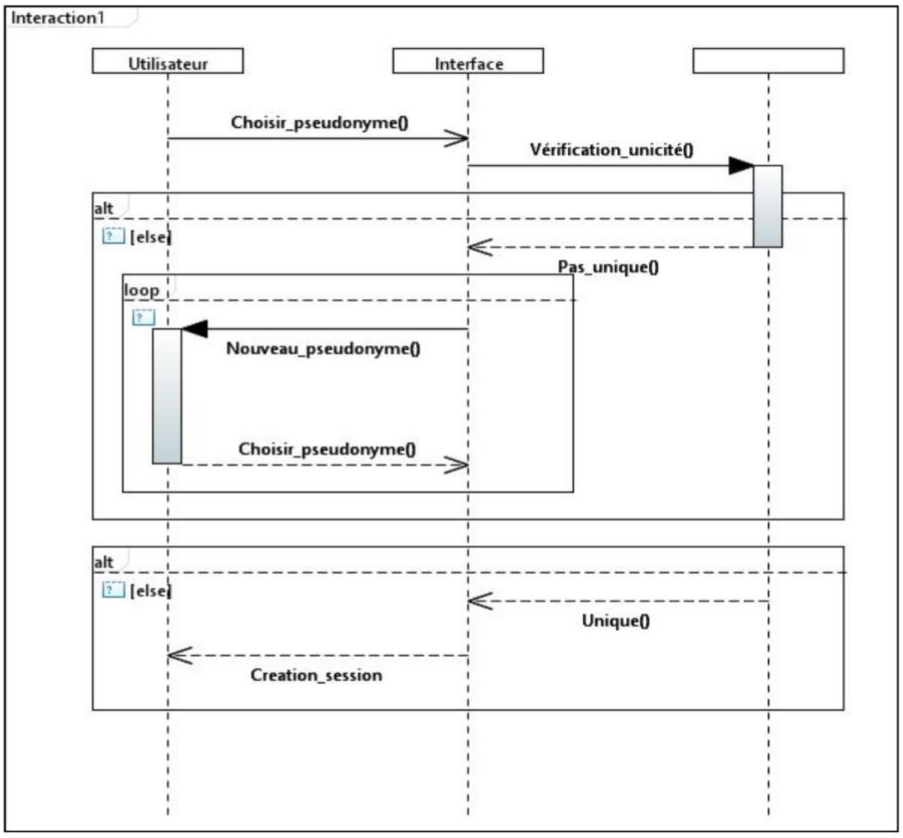
1. Use case diagram:



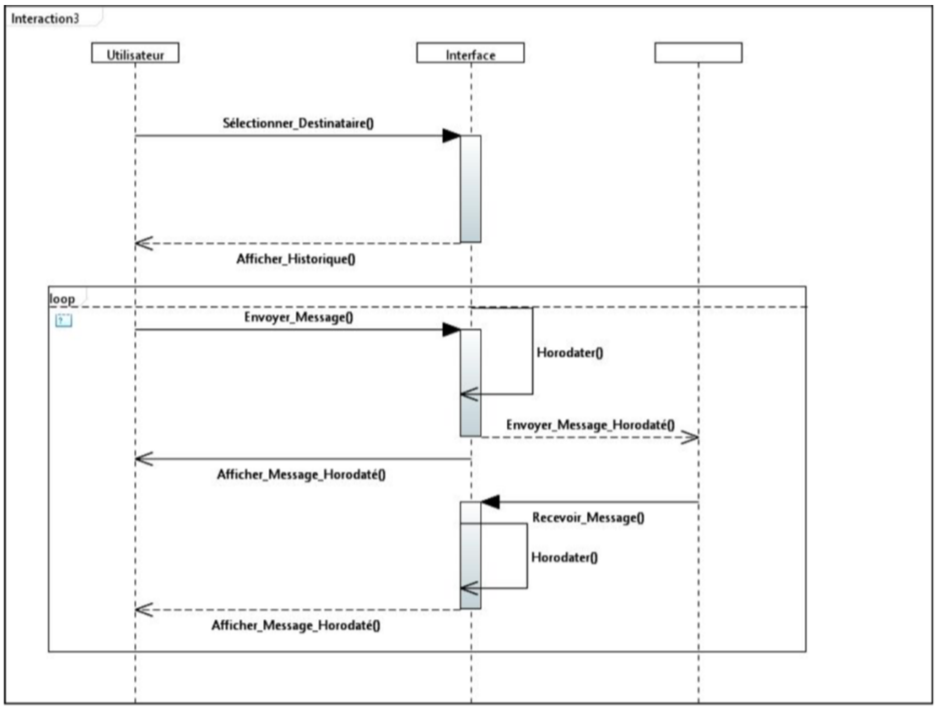
Our use case diagram has evolved to implement all the functions imposed by the specifications.

2. Sequence diagrams:

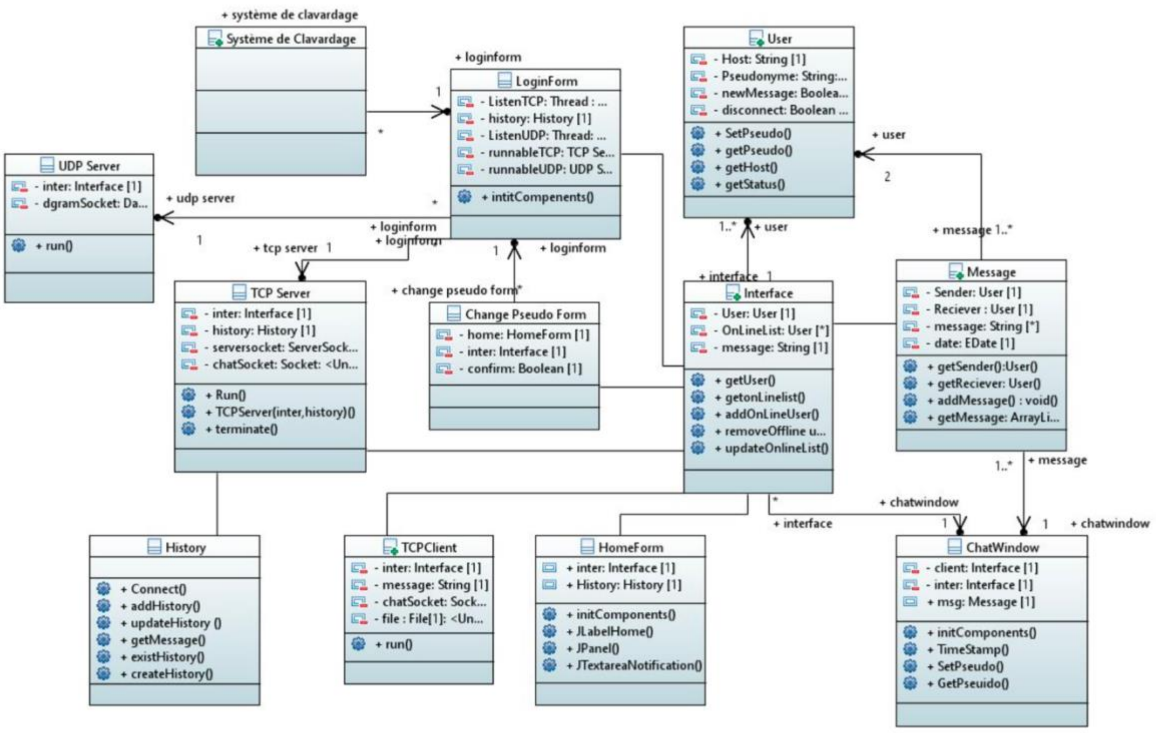
a) Session creation:



b) Send and receive messages:



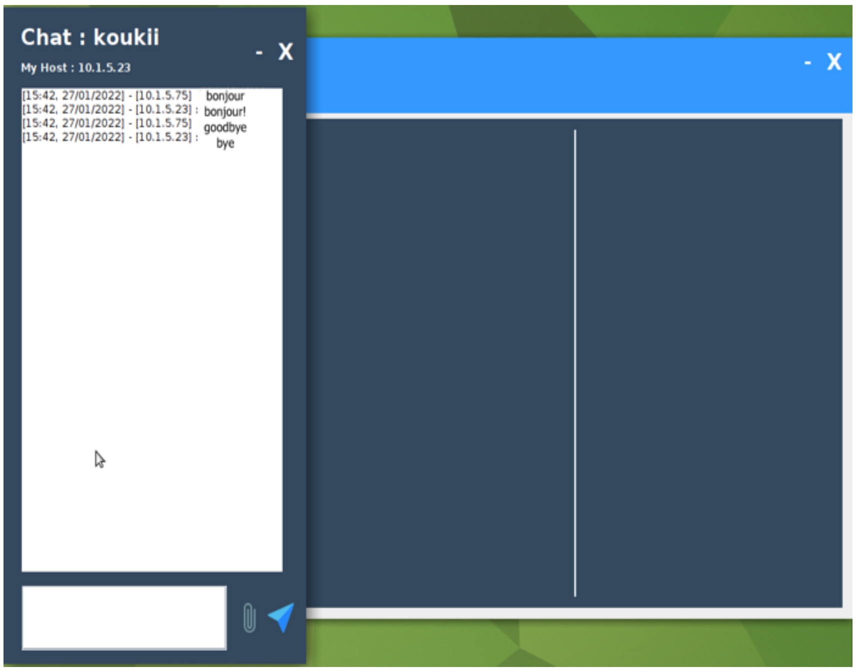
c) Class diagram:



II. Object Oriented Programming:

1. Presentation:

According to the specifications, this application will be used within a company, so we assumed that users would have fixed stations and therefore fixed IP addresses that identify them.


Chat : koukii

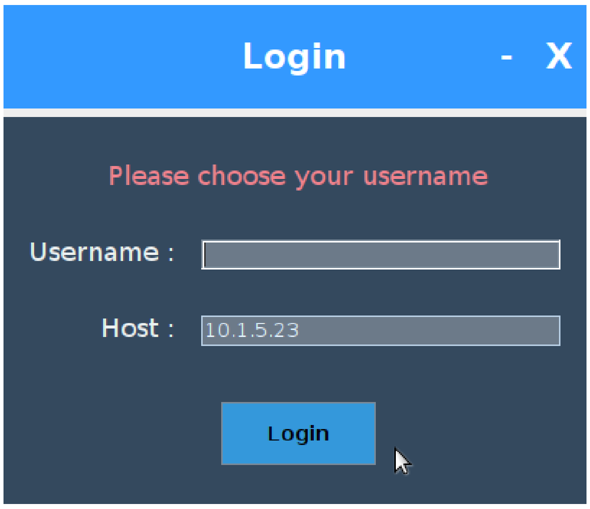
My Host : 10.1.5.23

```
[15:42, 27/01/2022] - [10.1.5.75]    bonjour
[15:42, 27/01/2022] - [10.1.5.23] :  bonjour!
[15:42, 27/01/2022] - [10.1.5.75]    goodbye
[15:42, 27/01/2022] - [10.1.5.23] :  bye
```

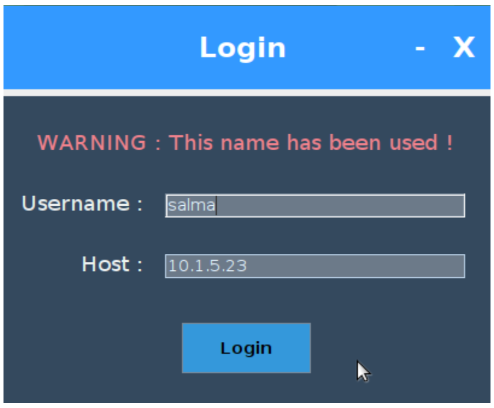
2. User manual:

In order to run the program, you have to open a terminal in the folder where you downloaded the program and type the following command line: **java -jar ChatSystem.jar**

For a first connection you must choose a username. Type your pseudonym in the right field and click on login or hit the “enter” key on your keyboard.
You cannot change the IP address in the “Host” field is auto filled and cannot be changed. It is unique to each device.



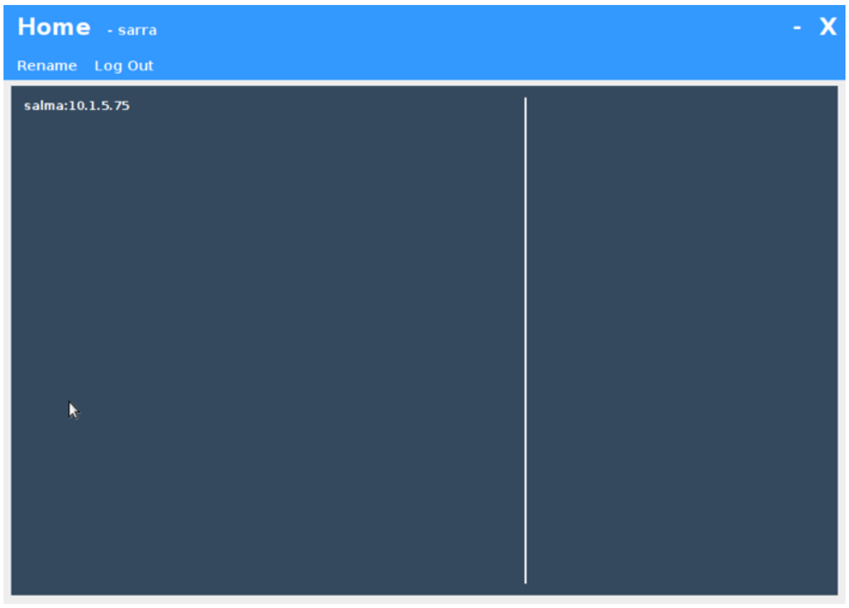
If your nickname is not unique, an error message will appear, and you will have to choose another one. If the nickname is not unique, you will be unable to log in.


WARNING : This name has been used !

Username : salma

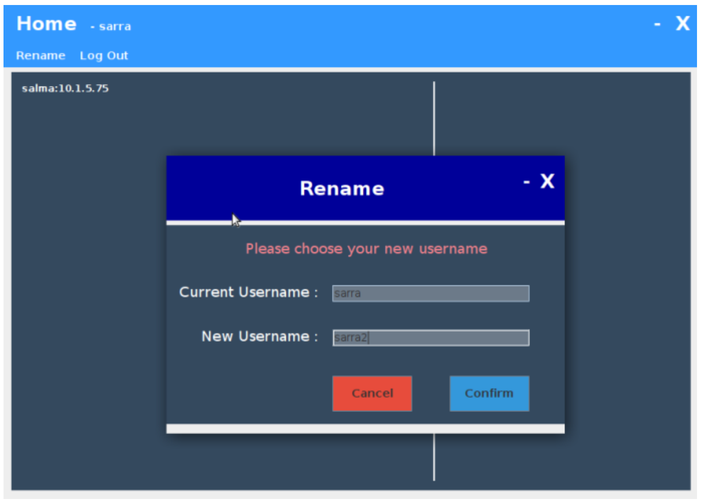
Host : 10.1.5.23

Once connected, the list of other connected users will appear on the left section in the “Home” window. Click on one of the users to be able to start a conversation with him.


salma:10.1.5.75

When you receive a message, a [!] notification signal is displayed in front of the sender's username. Click on the sender's username to view the received message. You will find displayed the message received, its date and time and the history of your messages with this user if it exists. The history doesn't get deleted even if you log out, it is attached to your IP address.

If you want to change your click on the "Rename" button in the "Home" menu and you will be prompted to choose a new unique nickname.
Type the new username in the "New Username" field then confirm your choice by clicking on "Confirm" or hit the "enter" key on your keyboard.


Home - sarra

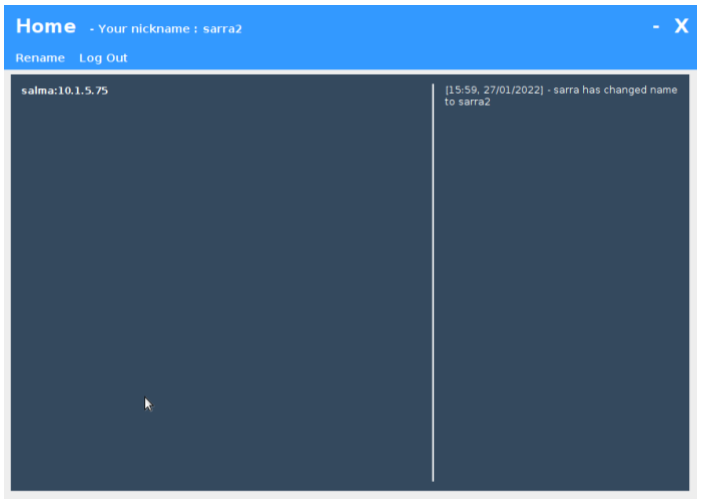
salma:10.1.5.75

Please choose your new username

Current Username : sarra

New Username : sarra2

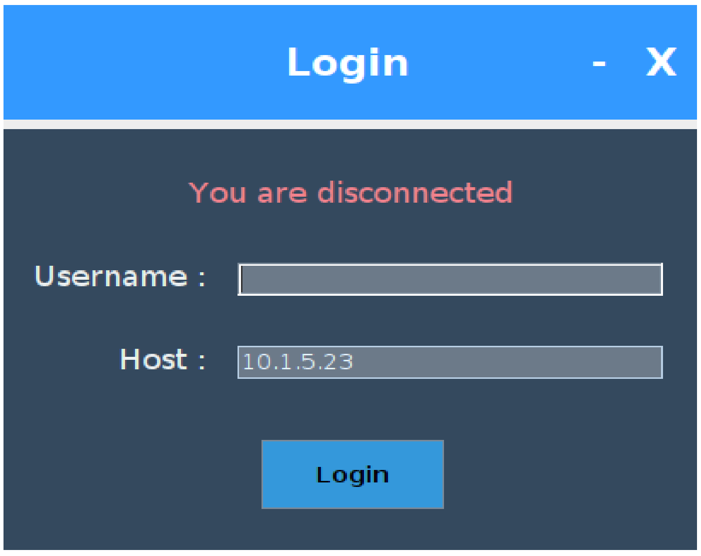
You will be notified each time a user changes their nickname in the notification field on the right side of the Home window


Home - Your nickname : sarra2

salma:10.1.5.75

[15:59, 27/01/2022] - sarra has changed name to sarra2

If you want to log out, click on the "Log out" button. And you will find the "Login" window.
For a future connection you can choose another pseudonym different the previous connection


You are disconnected

Username :

Host : 10.1.5.23

3. Interface:

To ensure the simplest interface for users of all computer levels, the **Swing** package (in Java: **javax swing**) is used, thanks to its many components which have extended functions and mechanisms to handle events while offering a graphical appearance that employs the style of familiar operating systems, such as Windows.
Our chat System has a total of 4 graphical user interfaces, which are presented here and whose functionalities are already explained in the manual: Login window, home window, Chat window and window for changing nickname (Rename).

3.1. Username management:

Each time a user logs in, he has the right to choose a username at his will, as long as the username is unique. If the uniqueness of the username is not satisfied, the user cannot enter the system, namely the home window, and remains in the Login window with a warning message. Once accessed the system, the user also has the right to change his username, provided that this new username is unique and different from the old username. If the constraints are not met, warning messages will be displayed in the Rename window.

3.2. Notification management:

The notification field located in the Home Window is used to inform the user of the change of username of other users in the network. Each time a nickname change signal is received, the notification field will display a corresponding notification.
Also, when the user receives messages that are not yet read, there will be a mark [!] next to the nickname of the sender of these unread messages.

3.3. Processing of text and visual messages:

Our Chat System supports both text and visual messages. Upon receipt of a message, the system will distinguish between a text message or an image.
During a conversation, all text messages will be displayed on the message field of the Chat window, while, when receiving an image, it will be displayed directly on the user's screen, plus a notification on the Chat messages field.

4. Connect and send messages:

4.1. User login/logout management:

Our Chat System enables a communication within a local network according to the peer-to-peer model, thus the connection and disconnection of users are managed via UDP broadcast messages: each time a user logs in, he, automatically sends a UDP message to all the other users online to update their table of online users. Same scenario during the disconnection phase.

4.2. Sending/receiving messages:

Our Chat System uses the TCP service in the transport of messages, when sending and when receiving. Unlike UDP, TCP provides reliable message delivery. TCP ensures that data is not damaged, lost, duplicated, or delivered out of order to a receiving process.

4.3. Online user list management:

The list of online users, found in the Home window, is managed automatically: depending on the "signal" received - in the form of UDP broadcast messages, whether it is a connection, disconnection or change of username, the list of online users will automatically add, remove or update the user who sent this signal.

In summary, the online users list ensures that users who are online, not offline, are correctly displayed along with their currently used usernames.

5. Database:

For the creation of the database, we used the **JDBC SQLite API** which is easy to deploy and support by all operating systems. The database is used to record messages, their date of sending, the IP addresses of their senders and receivers to generate the history of messages between 2 users. Messages are inserted into the database as they are sent and received.

6. Testing and validations:

6.1. Validation of requirements:

6.1.1. Agent Functional Requirements:

●Agent Administration Features:

✓ [CdC-Bs-1] The system must be able to be deployed on a workstation running on the Windows operating system
✓ [CdC-Bs-2] The system must be able to be deployed on a workstation running on the Linux operating system
✓ [CdC-Bs-3] The system must be able to be deployed on a workstation running on the OS X operating system
✗ [CdC-Bs-4] The system must be able to be deployed on a workstation running on the Android operating system (it is not possible to use Java Swing on Android)
✓ [CdC-Bs-5] System deployment should be limited to copying a series of files to the workstation and creating a shortcut for the user (downloading and copying a JAR in the desired directory)
✓ [CdC-Bs-6] The overall size of all the resources making up the system must not exceed 50 Mega-Bytes in uncompressed form regardless of the operating system on which it is deployed (File size .JAR: 18.9 MB)

● Agent Usage Features:

✓ [CdC-Bs-7] The system must allow the user to choose a pseudonym with which he will be recognized in his interactions with the system (In the Login window)
✓ [CdC-Bs-8] The system must allow the user to simply identify all the users for whom the agent is active on the network (In the list of online users displayed in the Home window)
✓ [CdC-Bs-9] The system must allow the user to start a chat session with a system user that he will choose from the list of users for whom the agent is active on the network (By clicking on the user's nickname in the list of online users)
✓ [CdC-Bs-10] The system must guarantee the uniqueness of the pseudonym of the users for whom the agent is active on the network (Verification of the uniqueness of the pseudonym is carried out during Login and Change of Pseudonym)
✓ [CdC-Bs-11] All messages exchanged within a chat session will be timestamped
✓ [CdC-Bs-12] The timestamp of each of the messages received by a user will be accessible to him in a simple way (Direct display next to the message content)
✓ [CdC-Bs-13] A user can unilaterally end a chat session
✓ [CdC-Bs-14] When a user starts a new chat session with a user with whom he has previously exchanged data through the system, the message history is displayed (Display in the Chat)
✓ [CdC-Bs-15] The user can minimize the agent, in this case, it is discreetly placed in the taskbar as an icon when the operating system allows this functionality
✓ [CdC-Bs-16] The system must allow the user to change the pseudonym he uses within the chat system at any time (By clicking on the Rename button in the Home window)
✓ [CdC- Bs-17] When a user changes his nickname, all the other users of the system are informed (Notification displayed in the Notifications part of the Home window of all users)
✓ [CdC-Bs-18] The change of pseudonym by a user must not lead to the end of the chat sessions.

6.1.2 Operational requirements:

● Performance requirements:

✓ [CdC-Bs-19] The deployment of the system must be feasible in 2 hours from the deployment decision. (Deployment takes a maximum of 10 minutes: Download the folder, then launch the .JAR file)
✓ [CdC-Bs-20] The change of a user's pseudonym must be visible to all other users in less than 20 seconds (The change of pseudonym is visible is almost instantaneous)
✓ [CdC-Bs-21] The time elapsed between sending a message by one user and receiving it by another user must not exceed 1 second
✓ [CdC-Bs-22] The system must allow the setting up of 1000 simultaneous chat sessions within it (This will not pose a problem because the exchanges of messages are of Peer-to-Peer method)
✓ [CdC-Bs-23] The agent must allow the setting up of 50 simultaneous chat sessions (Each Chat session consists of 2 threads while the maximum number of threads on a machine is around 105)

✓ [CdC-Bs-24]When the verification of the uniqueness of the user's nickname fails, the user must be informed within a period not exceeding 3 seconds (The warning displays instantly) ✓ [CdC-Bs- 25]The time of appearance of users within the list of users for which the agent is active must not exceed 5 seconds (Instant display)
✓ [CdC-Bs-26] The system must allow simultaneous use by at least 100,000 users

● Resource requirements:

✓ [CdC-Bs-27]The system must have a memory footprint of less than 100MB
✗ [CdC-Bs-28] During its execution, the system must not solicit the processor more than 1% of the time when the measurement is carried out over an interval of 5 seconds
✓ [CdC-Bs-29]The system must exhibit normal responsiveness for a chat application

● Dependability requirements:

✓ [CdC-Bs-30]The system must guarantee message integrity greater than 99.999% (Test campaign)
✓ [CdC-Bs-31] Normal use of the system should not have any impact on the rest of the system

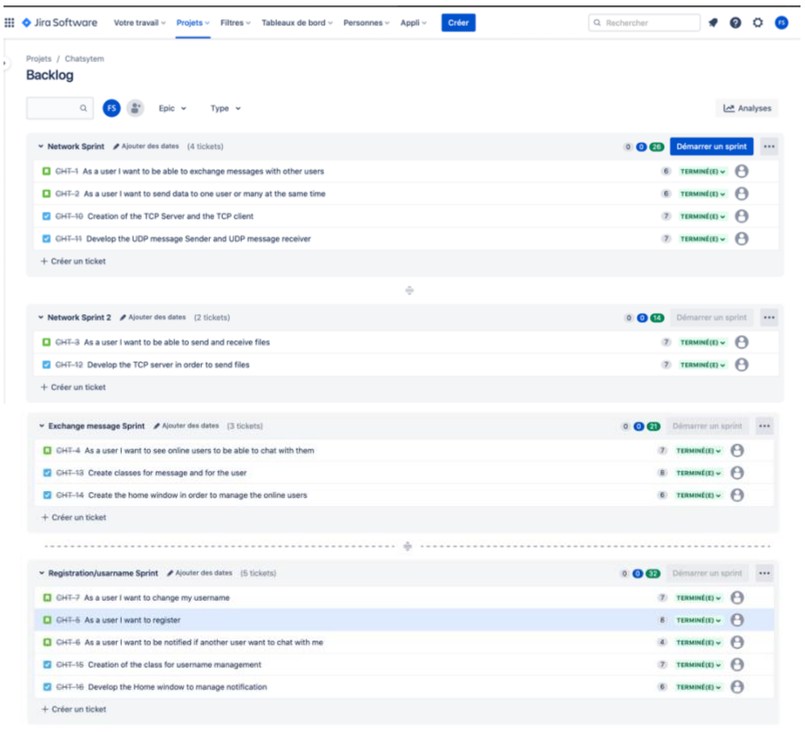
● Special requirements:

✓ [CdC-Bs-32] The system must allow a simple extension of the functionalities using a system of modules which will be the subject of standardization

III. Project Management:

1.Agile Method: Jira:

During this work we discovered a very useful method to manage our project. It's the Agile Scrum method. We used the Jira tool in order to create sprints. A sprint designates the development cycle during which a certain number of tasks will be linked together to ultimately end with the design of a final product. We find this method very interesting because the creation of the sprints depended on our development rhythm. Therefore, we adapted the deadlines and goals according to our progress. The implementation of the Network package was the longest to work



Backlog

Network Sprint   Ajouter des dates   (4 tickets)   0 0 26   Démarrer un sprint

~~CHT-1~~	As a user I want to be able to exchange messages with other users	6	TERMINÉ(E)
~~CHT-2~~	As a user I want to send data to one user or many at the same time	6	TERMINÉ(E)
~~CHT-10~~	Creation of the TCP Server and the TCP client	7	TERMINÉ(E)
~~CHT-11~~	Develop the UDP message Sender and UDP message receiver	7	TERMINÉ(E)

+ Créer un ticket

Network Sprint 2   Ajouter des dates   (2 tickets)   0 0 14   Démarrer un sprint

~~CHT-3~~	As a user I want to be able to send and receive files	7	TERMINÉ(E)
~~CHT-12~~	Develop the TCP server in order to send files	7	TERMINÉ(E)

+ Créer un ticket

Exchange message Sprint   Ajouter des dates   (3 tickets)   0 0 21   Démarrer un sprint

~~CHT-4~~	As a user I want to see online users to be able to chat with them	7	TERMINÉ(E)
~~CHT-13~~	Create classes for message and for the user	8	TERMINÉ(E)
~~CHT-14~~	Create the home window in order to manage the online users	6	TERMINÉ(E)

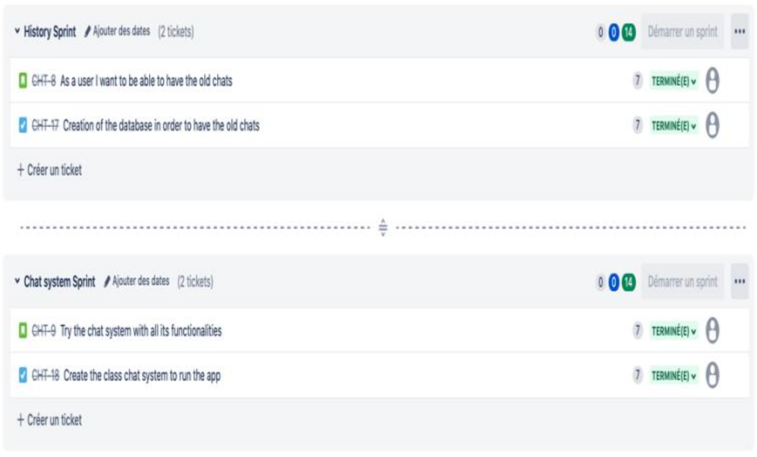
+ Créer un ticket

Registration/usarname Sprint   Ajouter des dates   (5 tickets)   0 0 32   Démarrer un sprint

~~CHT-7~~	As a user I want to change my username	7	TERMINÉ(E)
~~CHT-5~~	As a user I want to register	8	TERMINÉ(E)
~~CHT-6~~	As a user I want to be notified if another user want to chat with me	4	TERMINÉ(E)
~~CHT-15~~	Creation of the class for username management	7	TERMINÉ(E)
~~CHT-16~~	Develop the Home window to manage notification	6	TERMINÉ(E)

+ Créer un ticket

on. So, we had to extend the first sprint. Then we picked up the right rhythm and the process got smoother as we started working in parallel.
You can find the captions of the different sprints that we implemented it during our project. However, we encountered problems with the Jira tool online every time the sprint that we define get randomly delete it so that's why we had to constantly rewrite them. It might be the case when you visit our account.


History Sprint — Ajouter des dates (2 tickets) — 0 0 14 — Démarrer un sprint
~~CHT-8~~ As a user I want to be able to have the old chats — 7 — TERMINÉ(E)
~~CHT-17~~ Creation of the database in order to have the old chats — 7 — TERMINÉ(E)
+ Créer un ticket

Chat system Sprint — Ajouter des dates (2 tickets) — 0 0 14 — Démarrer un sprint
~~CHT-9~~ Try the chat system with all its functionalities — 7 — TERMINÉ(E)
~~CHT-18~~ Create the class chat system to run the app — 7 — TERMINÉ(E)
+ Créer un ticket

2. Maven:

During the development of our chat system, we used the build tool Maven. Its main job was to configure the project, compile using required libraries and do the final packaging. Therefore, Maven automated the process of packaging the app and managing the dependencies.

Conclusion:

In conclusion, this object-oriented design and programming project allowed us to train ourselves in project management from design to implementation. The fact of starting from a specification only to program the application was a new experience and a challenge on another level and it allowed us to have an idea of our job as an engineer in the future. In addition, this project represents the perfect combination between different areas of computer science: design, telecommunications, and database management, which allows us to deepen our knowledge.