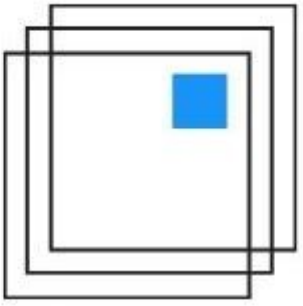
Object Oriented Programming

Chat System Project

Laura Burlon--Roux - Aurélien Michaud
4 IR A1 | 2019/20



AURA
INDUSTRIAL SYSTEMS

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Administration guide

1 - Get the sources

First, you need to fetch the source code online, by cloning the following git

https://github.com/Lau2203/ChatSystem_Project

```
git clone https://github.com/Lau2203/ChatSystem_Project
```

---

2 - Setup Aura

Then, you will need to compile and install the software from the source. Go to the cloned git directory and be sure to let `aura`, `aura_remote` and `setup.sh` files be executables

```
cd ./ChatSystem_Project && chmod +x ./aura ./aura_remote ./setup.sh
```

You can now execute the setup shell script (`setup.sh`), it will basically compile the software from source, creating a jar file (which you will be able to take anywhere), and ask for a new account's password if resources/witness file does not exist (which is the file containing the encrypted password's key). That password will then be asked to the user at the login time.

Usual execution of `setup.sh` :

```
Creating 'jar' directory...
Compiling from source...
Creating jar file...
[*] No account was found, do you want to create one ? [yes] :
New password:
Confirm new password:
Removing compiled files...
rm -rf ./build/ ./logs*
Setup done.
```

---

3 - Remote server setup

As for the remote server, please stop it and start it again, since the servlet might have previous memory of users not online anymore, so that you have a clean and untouched new servlet instance. The war file is located in `server/web-app/AURA-Server.war`.
As you will be able to understand in the *Notes & Future Improvements* section, the servlet seems to be working fine, but the remote side of aura (we are talking about `aura_remote`, not `aura` script) still have difficulties handling the server's response.

---

4 - Important files

In case of troubleshooting, please take into account the following files and their meaning for `Aura`.

- `configs/config.cfg`

This file is. You can directly modify it by hand if you will to, but be careful to respect the format `^key=value$`.

```
witness-file=../resources/witness
nsl-port=54321
username=Casgon
remote-server-url=https://srv-gei-tomcat.insa-toulouse.fr/AURA-Server/AURA-Server
remote-server=0
server-hostname=archlab
server-port=5555
log-file=../logs
message-history=../resources/history/history_backup.mh
log-filling=true
```

If `log-filling` is true, do not hesitate to have a look into the log file (its path is the value of the `log-file` key). `message-history` mentions the path of the message history xml file. As you can see, it can be useful to modify the `remote-server-url`, `nsl-port` or `server-port`, but keep in mind that all the instances of Aura will need to get their configuration file modified, since they all work with the same port numbers and remote server URL. You can also see the `remote-server` boolean variable, which disables the servlet requests when equals to 0, and enables it otherwise. As it will be later explained, the implementation of the servlet is still

fragile, so you can toggle its function by changing that `remote-server` value. For obvious security reasons dealing with the witness file, this config file should only have root permission for writing.

- `resources/history/`

This directory should contain all the XML-formatted history files. Aura will only use one of these files which is referenced in the `configs/config.cfg` file (value of the `message-history` key). By default it is history_backup.mh but you can use another file aswell, e.g. when that history file reaches a considerable size.

- `aura & aura_remote`

These are the shell scripts launching the Aura software. `aura` is supposed to be launched when in the company's local network and `aura_remote` should be used while in a remote network. It could be useful to add an alias at the end of the `$HOME/.bashrc` file to make the command 'aura' execute that `aura` or `aura_remote` script.

Implementation choices

Security : authentication key

When a user first launches the Aura software, it is asked to enter a new password (through the `setup.sh` shell script). We then take the sha256 fingerprint of that password (then base64'ed), available to everyone, so that we are able to identify everyone, with a constant and unmodifiable string. At the first launch, that same password is used as a key to encrypt phrase ("THAT IS COOL") into a file. That process is useful for the login method. Once the user enters the password in the login window, the password he entered is used as a key to decrypt that same file. If the decrypted string is the same ("THAT IS COOL"), then we know he typed the correct password and he is then logged in.

Message history : XML or DBMS ?

The application maintains a history of time-stamped messages, local to the machine which it runs on. Consequently, we assume that each user has a dedicated machine he will work on (and launch Aura on). This history is then translated into an XML format. We have chosen to use XML instead of a database because :

- It is very easy to convert an XML document into another document in PDF, HTML, Excel or CSV format : the user should easily be able to export a copy of the messages.
- An XML file is directly editable, which is easier to test our chat system application (cf. testing procedures).

Being able to talk to ourselves

We decided to let Aura see ourselves as a full-blown user. We thought that that can be useful, especially when we send notes to ourselves not to forget them, as almost all the current world wide instant messaging apps allow.

User guide

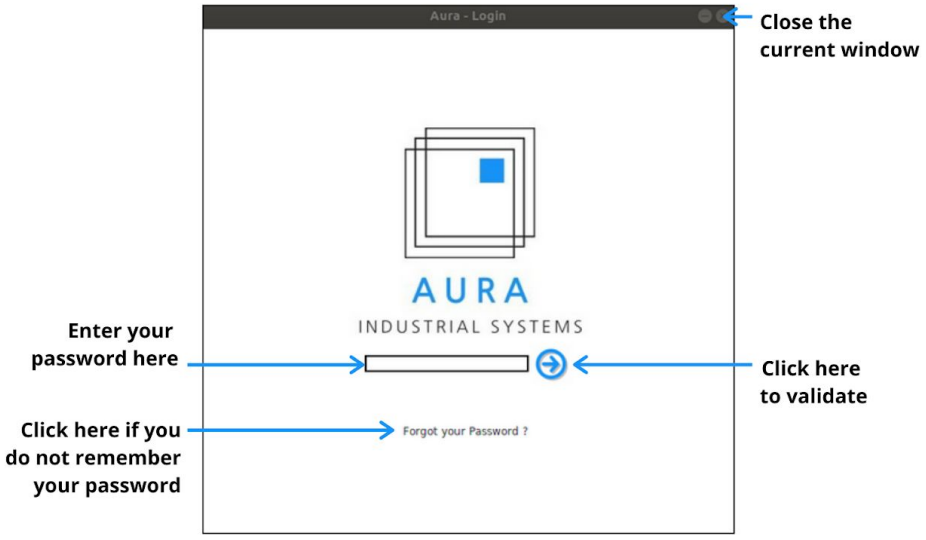
Quick and easy to use, Aura enables you to communicate with your work colleagues.

Main advices for a pleasant utilisation of our services :

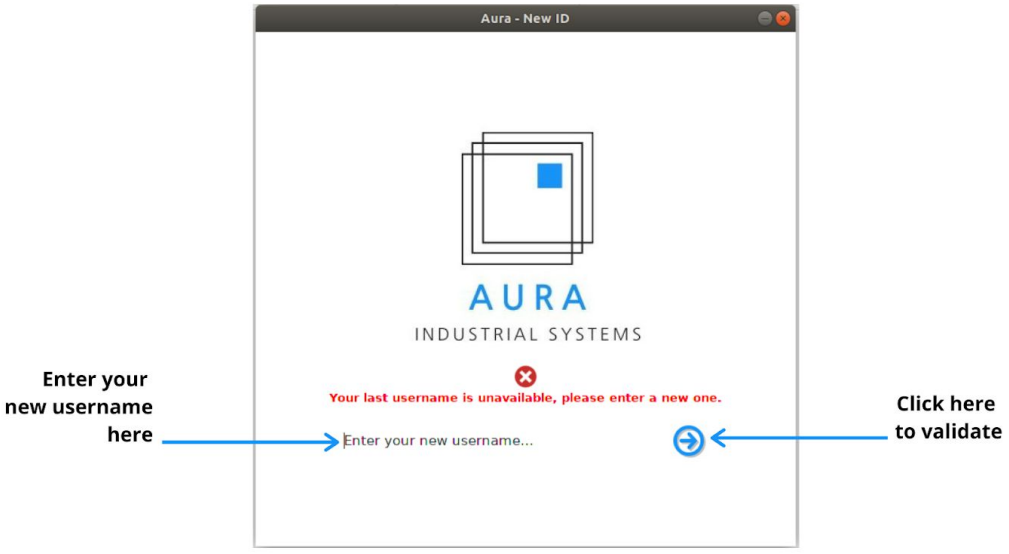
- Use Aura Chat System only on your own computer.
- Do not share your password with anyone else, it is strictly confidential.
- All your history is stored in an encrypted directory on your computer.

To launch the application, please enter `Aura` on your terminal.

While the application is launched, the connection window appears. You have to enter the password which was given by your network administrator. If your password is incorrect, a message “Wrong login information” will be displayed.



To ensure the effective functioning of Aura Chat System, every username must be unique. If your last username, used during your latest connection, is now used by another user, you have to change it and find another one. This window is now displayed :

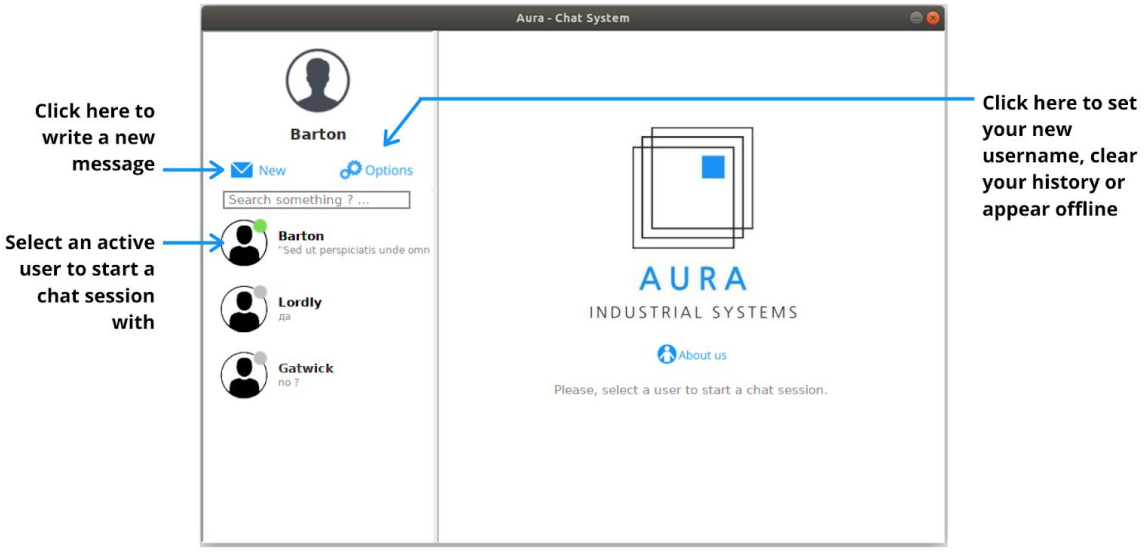


When you are connected to the Chat System, a bigger window is displayed : it is the main window of the application.

How to start a chat session from the main window ?

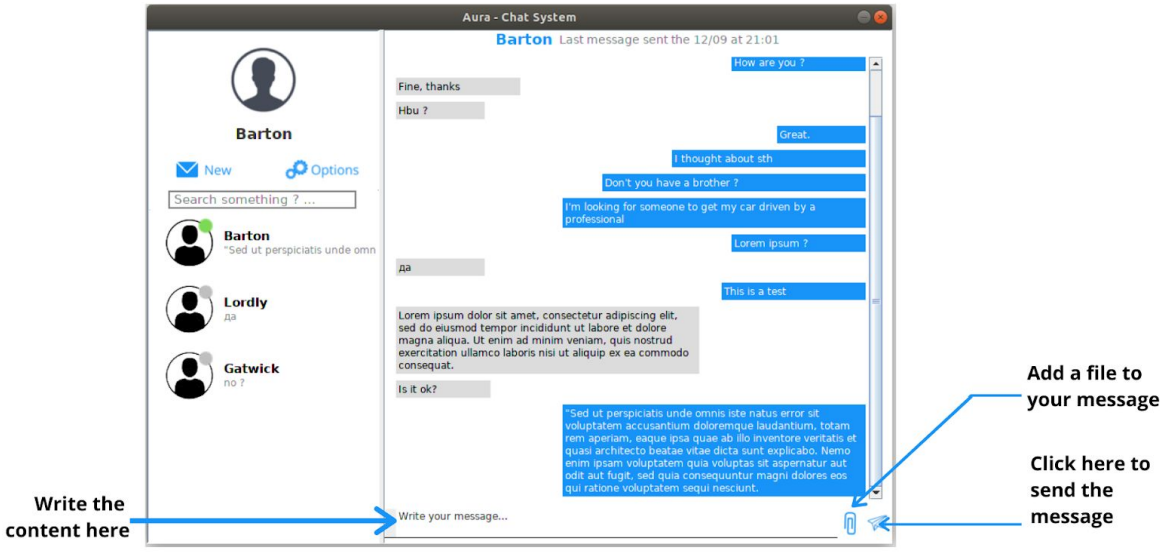
On the left of your screen, you can select an active user to start a chat session with him. An active user has a green mark on his profile picture.

You can also click on the New button to start a chat session.

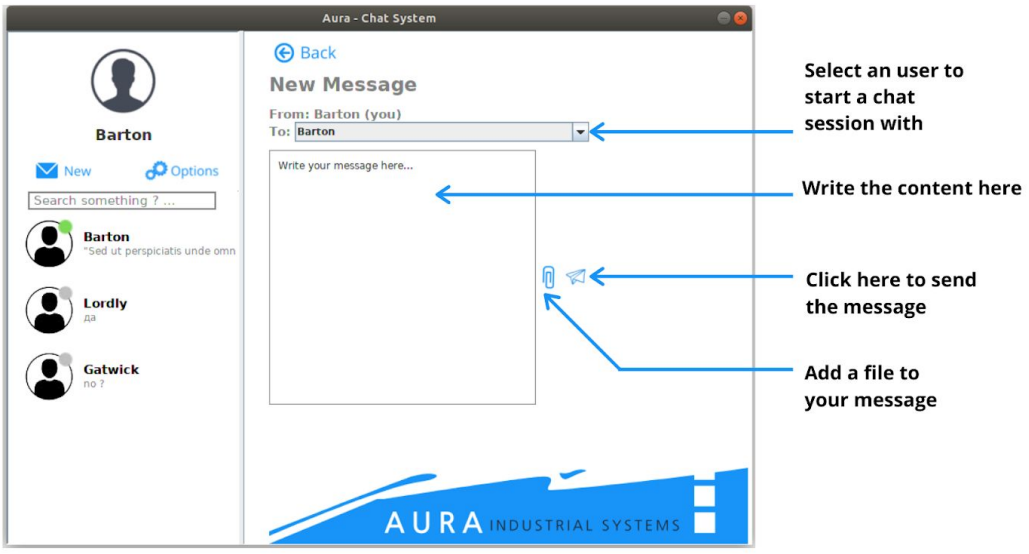


How to send a text message or a file ?

If you have selected an active user,



If the user is not active but you still had previously a conversation with him, his profile picture has a gray mark and all buttons on the conversation are unavailable. If you received a file from another user, you will be able to find it in the `resources/uploads/` directory. If you have clicked on New button, you can see the following frame :



All messages are timestamped, if you would like to see when the message was sent, put your cursor on it and wait : the date and the hour are displayed.

How to change my username ?

You have to clicked on Options button, and next enter your new username. In the next version of Aura Chat System, you will also be able to clear your history and set your connection status offline in the near future.

Please keep in mind that '*Clear your history*' and '*Appear offline*' options are not operational yet.

For any further information, please feel free to contact your network administrator.

Testing procedures

New active user and user disconnection notifications

In order to make sure a new connected user or the disconnection of a user would be notified to all the other online users, we basically started by launching one instance of Aura on one computer. Then we ran another instance on another computer. On the left side of the application, an online user would now appear with the green dot on the top right corner of the profile picture of that user. If a user goes offline, that green dot becomes gray. This objective was quite

Servlet part

The servlet accepts GET and POST methods. Once a user log in the Aura software and has a valid username, the software calls a POST method on the servlet in order for him to add it to the active users list. Then, every 3 seconds, the software orders a GET request to the servlet in order to get the active user list.

Unfortunately, the aura_remote side seemed to have difficulties taking into account the server's response, although we clearly saw that the server's response was correct (correct connected users with their username, fingerprint and address). Of course it seems to work fine with the local aura software, since it can still rely on the broadcast signals, but we can see the servlet still needs improvements.

Message and File Transfer

Message sharing was one of the first features we developed, so we had been using it throughout the whole chatsystem development in order to test other features (such as online users apearance in the GUI). As for the file transfer, it was one of the last functions we added before the servlet. Although that works perfectly fine, the maximum size of the files (and messages) Aura is able to handle is hard coded into the source code.

Message History

Regarding the message storage, the XML format allowed us to test the process of reading, displaying and storing the message history way more easily than it would have been with a DBMS. We could add, modify, suppress directly the message history and that helped us know what was wrong when we tested the message history storage system.

Cross Platform

Throughout the whole development, we have to say that we mainly coded and tested in the Linux environment, though we tried Aura on Windows at some point (in particular the message sharing part). However, we cannot ensure that it is truly proficient under Microsoft's operating system.

Notes & Future Improvements

Graphical User Interface (GUI)

Concerning the GUI, there is still visible work to do. The two options '*Clear your history*' and '*Appear offline*' need to get implemented and there sometimes appears to be not-thread-safe behaviours with the GUI's thread task dispatcher, raising an exception.

Remote server

The implementation of a remote service by the means of a servlet remains unstable. Indeed, the actual servlet looks like it's working fine, but the client side handling its responses happens to allow some undesirable side effects, such as clearing off the whole user list on the GUI when fetching that list from the remote server. So please be careful when using the remote server (use of `aure_remote` and/or setting to 1 the `remote-server` key in the

`configs/config.cfg` file). Please do not forget to restart the servlet since it may still have memory of previous tests, thus you might be seeing unknown people online otherwise.

Configuration range

Despite the fact that we tended to use the config file to keep Aura and the tests as flexible as we wanted, we still have some hard coded parameters such as the file's maximum size for transferring. We thought we could use it further by adding features to Aura so that it could deploy an administrative interface which would then configure the config file in a much easier way.