



ft_transcendence

Soon, you will realize that you already know things
that you thought you didn't

Summary:

No more C! No more C++!

*This project is about doing something you've never done before.
Remind yourself the beginning of your journey in computer science.
Look at you now. Time to shine!*

Version: 11.2

Contents

I	Preamble	2
II	Mandatory part	3
II.1	Overview	3
II.2	Security concerns	4
II.3	User Account	4
II.4	Chat	5
II.5	Game	5
III	Submission and peer-evaluation	7

Chapter I

Preamble



Chapter II

Mandatory part

This project is about creating a website for the mighty **Pong** contest!

II.1 Overview

Thanks to your website, users will play Pong with others. You will provide a nice user interface, a chat, and real-time multiplayer online games!

Your work has to comply with the following rules:

- Your website backend must be written in **NestJS**.
- The frontend must be written with a **TypeScript** framework of your choice.
- You are free to use any library you want to in this context. However, you must use the **latest stable version** of every library or framework used in your project.
- You must use a **PostgreSQL** database. That's it, no other database.
- Your website must be a **single-page application**. The user should be able to use the **Back** and **Forward** buttons of the browser.
- Your website must be compatible with the **latest stable up-to-date version** of *Google Chrome* and one additional web browser of your choice.
- The user should encounter no unhandled errors and no warnings when browsing the website.
- Everything has to be launch by a single call to: `docker-compose up --build`



When your computers in clusters run under Linux, you will use Docker in rootless mode for security reasons. This comes with 2 sideways: 1) your Docker runtime files must be located in `/goinfre` or `/sgoinfre`. 2) you can't use so called "bind-mount volumes" between the host and the container if non-root UIDs are used in the container. Depending on the project, your situation and the context, several fallbacks exist: Docker in a VM, rebuild you container after your changes, craft your own docker image with root as unique UID.

II.2 Security concerns

In order to create a fully functional website, here are a few security concerns that you have to tackle:

- Any password stored in your database must be **hashed**.
- Your website must be protected against **SQL injections**.
- You must implement some kind of server-side validation for forms and any user input.



Please make sure you use a strong password hashing algorithm



For obvious security reasons, any credentials, API keys, env variables etc... must be saved locally in a `.env` file and ignored by git. Publicly stored credentials will lead you directly to a failure of the project.

II.3 User Account

- The user must login using the **OAuth** system of 42 intranet.
- The user should be able to choose a unique name that will be displayed on the website.
- The user should be able to upload an avatar. If the user doesn't upload an avatar, a default one must be set.
- The user should be able to enable **two-factor authentication**. For instance, *Google Authenticator* or sending a text message to their phone.
- The user should be able to add other users as friends and see their current status (online, offline, in a game, and so forth).
- Stats (such as: wins and losses, ladder level, achievements, and so forth) have to be displayed on the user profile.
- Each user should have a **Match History** including 1v1 games, ladder, and anything else useful. Anyone who is logged in should be able to consult it.

II.4 Chat

You also have to create a chat for your users:

- The user should be able to create **channels** (chat rooms) that can be either public, or private, or protected by a password.
- The user should be able to send **direct messages** to other users.
- The user should be able to block other users. This way, they will see no more messages from the account they blocked.
- The user who has created a new channel is automatically set as the **channel owner** until they leave it.
 - The channel owner can set a password required to access the channel, change it, and also remove it.
 - The channel owner is a channel **administrator**. They can set other users as administrators.
 - The administrators of a channel can ban or mute users for a limited time.
- The user should be able to invite other users to play a Pong game through the chat interface.
- The user should be able to access other players profiles through the chat interface.

II.5 Game

The main purpose of this website is to play Pong versus other players and show everyone how good you are!

- Therefore, users should be able to play a live Pong game versus another player directly on the website.
- There must be a **matchmaking system**: the user can join a queue until they get automatically matched with someone else.
- It can be a canvas game, or it can be a game rendered in 3D, it can also be ugly, but in any case, it must be faithful to the **original Pong** (1972).
- You must offer some **customization options** (for example, power-ups or different maps). However, the user should be able to select a default version of the game without any extra features if they want to.
- The game must be **responsive**!
- The user should be able to watch a live play between other users without interfering with it.



Think about network issues, like unexpected disconnection or lag.
You have to offer the best user experience possible.

Chapter III

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your files to ensure they are correct.