MSc Computer Engineering

Distributed Systems and Middleware Technologies

Taboo-Game Web Application
(Distributed Platform)

Group Members:
Francesco Bruno
Gaetano Sferrazza

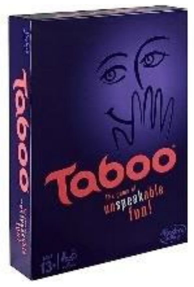
Academic Year: 2023/2024

Contents

Project Specification

Introduction

The application proposed is inspired by the boardgame "Taboo", in which two teams are challenged with the aim of getting their team members to guess a word, but without saying one of the so-called taboo words, which is a list of five *TabooWords* related to the word to be guessed. For example, "*Police*," "*Stop*," "*Liberate*," "*Imprison*," and "*Handcuffs*" may not be used to make team members guess the word "*Arrest*". Moreover, the prompter chosen at the beginning of the game randomly by the application, has a maximum time of 1 minute to get his team members to guess as many words as possible. In each game session we have one prompter, multiple guessers and the system that acts as Controller. More precisely, the role of the System (the Controller) is to verify that the prompter does not use any of the so-called *TabooWords*.



We would like to make a webapp with the following features.

Functional Requirements

In our application we have implemented 3 types of actors:

- Administrator
- Registered user
- Unregistered user

The possible action that each actor can perform are:

1. An **Unregistered user** can:
    - Create an account.
    - Log-in to an account.

2. A **Registered User** can:
    - Login/logout
    - See its friends list (online and offline)
    - Add friends to play with
    - Browse his past matches.
    - Invite friends to play.
    - Create its team before play a match.
    - If the user has been invited as Rival, he can create its own team to deal with the other team.

3. An **Admin User** can:
    - Login/logout
    - Remove user.
    - Browse all the matches played from users.

Non-Functional Requirements

For what concerns **Non-Functional requirements** we have:

- Concurrent service accesses management
- Strong consistency during matches
- High service availability
- Allow high horizontal scalability.

Synchronization and Communication Issues

Let's introduce in the following paragraph, the main communication and synchronization issues:

- Multiple players exchange words with each other in its team during matches in a concurrently way. The game interface of each player must be synchronized.
- Timers of users must be synchronized, when the prompter of each team enters in the game session, it triggers the start of all guesser's timers.
- The score of each team must be synchronized in any case, whether one guesser guesses the secret word, or its prompter make a mistake (he used one of five *tabooWords*).
- Notify the user if he has been invited by another user to play a game.
- Notify all waiting users (those that already accepted the invite) that one of the invited users has rejected the invitation.
- At every match termination the score of the one team must be sent to the other to let know the users if its team won or not.

System Architecture

Architecture Overview

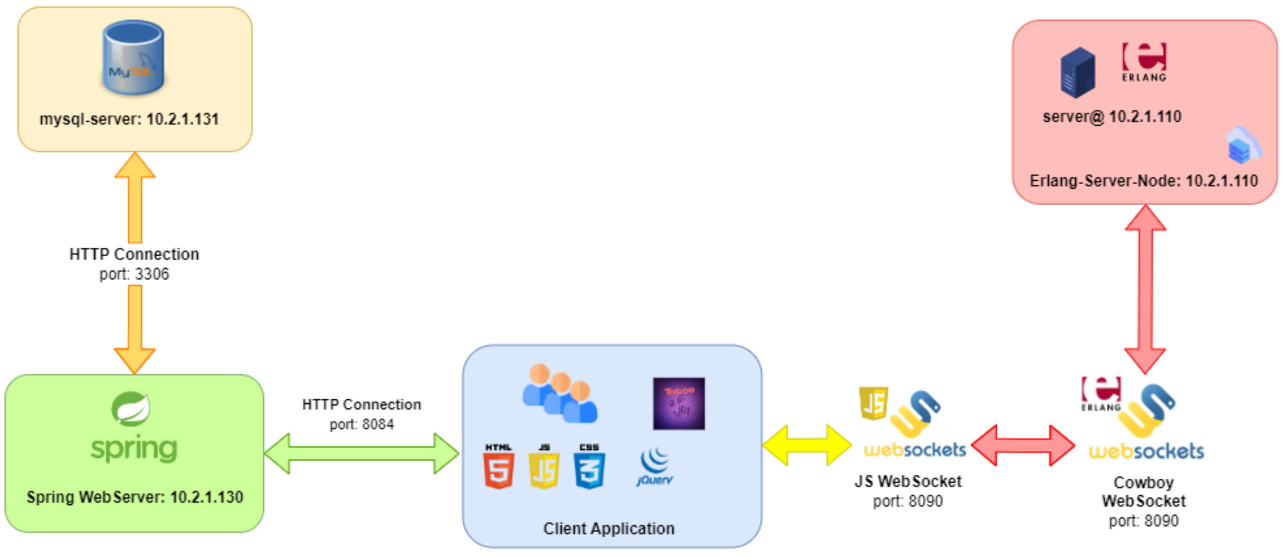


Figure 2.1: System Architecture Schema

The web application architecture, as we can see above in figure *Figure 2.1*, is divided into various distributed components to manage different functionalities and more specifically there are a **Client Application**, a **Spring Application Server**, a **MySQL RDBMS** and multiple **Erlang Server.**

Most of the functionality are implemented by ***Spring Application* Server**, that residing on node ***10.2.1.130*** and listening on ***port 8084***, which in turn communicates with the ***MySQL Server*** where the necessary data is persistent stored, the latter is hosted on node ***10.2.1.131*** listening on ***port 3306***. The ***Erlang Server*** process is in execution on the node ***10.2.1.110*** listening on ***port 8090***, handle the Gaming session of the application where the communication occurs via **HTTP Websocket** utilizing the **Cowboy** framework.

Spring Application Server

Spring Framework is a development framework for Java applications based on the concepts of *Inversion of Control* (IoC) and *Dependency Injection* (DI). As previously mentioned, within our *System Architecture*, it functions as an *application server* performing various crucial functions for the application's operation.

Firstly, by implementing authentication and authorization mechanisms, it allows for user authentication management and facilitates data retrieval from the *MySQL* database as well. Additionally, it handles client requests through its *Spring MVC* (Model-View-Controller) module, which provides an infrastructure for developing servlet-based web applications. This enables the creation of *RESTfu*l endpoints for service exposure and efficient and scalable management of client requests.

Specifically, its responsibilities include:

User Authentication: Verifying user credentials and granting access to authenticated users.

Signup: Allows users to create new accounts within the application, providing necessary credentials and information for registration.

View, remove and add friends: Enables users to manage their social connections by viewing, deleting, and adding friends to their network.

Invite friends to a game (As *Friend* or *Rival*): It offers users the ability to invite their friends to join them in a game as a Friend in the team they can create or as a rival by allowing them to create their own team to play a game against.

Browse Games: Permits users to explore their past games.

View and remove users (only for Admin User): Grants administrative users the privilege to view and remove other users from the platform.

Player Waiting Management: handle the waiting process for players awaiting others to join the game session.

Unique ID Assignment: Assigning a unique identifier to each game session, ensuring distinct identification and management of concurrent games.

Once all players are ready, the Spring Web Server orchestrates the commencement of the game session. It responds to clients by providing the usernames of all participants in the game.

MySQL Database

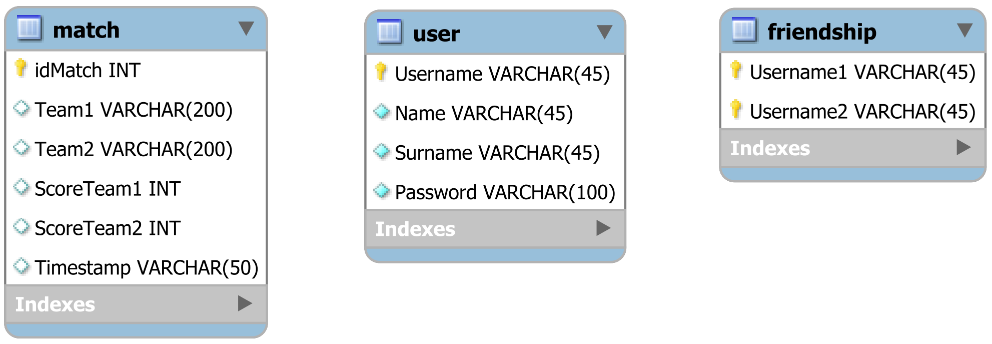


Figure 2.2: E-R Diagram Schema

The schema above, in *Figure 2.2*, represents the entities presents within MySQL Database. MySQL is a relational database management system (RDBMS) that we choose it because serves as a robust and versatile database solution, playing a critical role in supporting the functionality and data management needs of the system architecture, indeed we use it to persistent store data in the system architecture. Its role involves storing user information, details of matches played by users, and the relationships of friendship they have with other users.

As an integral part of the system, MySQL ensures data integrity, reliability, and efficiency in managing various types of information. It provides a structured framework for organizing and accessing data, supporting the storage and retrieval. With its support for transactions and data consistency mechanisms, MySQL ensures the reliability and accuracy of stored information.

Erlang Server

We've opted to build the Taboo-Game session features in Erlang, with a core focus on enabling a ***messaging system*** between *Prompter* and *Guesser* teammates. This system facilitates communication to maximize word guesses while incorporating the ***TabooCard***.

To fully leverage ***Erlang***'s inherent strengths in ***concurrent processing*** and *scalability*, we've implemented the game functionality. For this project specifically, we've configured a container to run its own independent Erlang instance (version 10.2.1.110).

The Erlang node exposes an endpoint that allows user browsers to connect via **WebSocket**. This enables the server to receive generic-taboo messages from any user and, if applicable, forward them to other active users, eliminating the need for webpage refreshes.

We've implemented the Erlang-side *WebSocket* communication using the **Cowboy Framework**, a lightweight yet powerful and modern HTTP server for Erlang/OTP. This approach lets us concentrate solely on the *business logic* and remote deployment, without deal with lower-level HTTP protocols. To streamline dependency management and automate the build process, we've chosen **Rebar3** for development and configuration of the before mention Erlang server structure. On the other hand, the client-side WebSocket communication is implemented in **Javascript**.

For message encoding between client and server, **JSON** is used as the data interchange format.

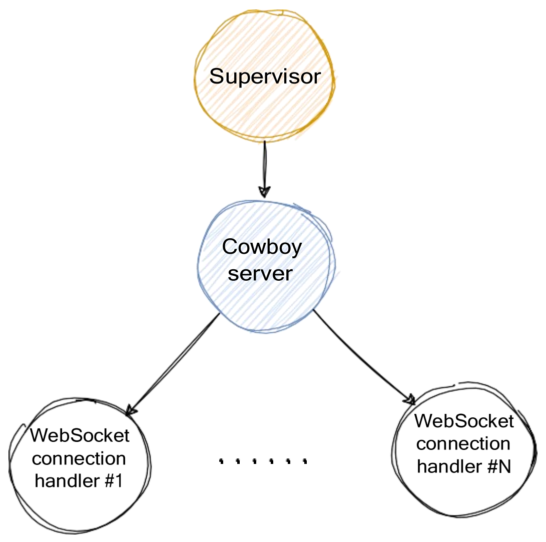

Supervisor
Cowboy server
WebSocket connection handler #1
. . . . . .
WebSocket connection handler #N

Figure 2.3: Hierarchy of Erlang processes

By dividing the functionalities of Erlang-side in two separate modules, ***server_handler.erl*** and ***player_handler.erl,*** we obtained a modular application. Furthermore, in this way we can exploit the **Dynamic Code Loading** erlang feature to increase the application maintainability. Indeed, we can maintain the Erlang Server on and update the code in the *player_handler.erl,* without restart it.

The previous two modules mentioned *in detail*:

1) **server_handler** *module*: responsible for managing WebSocket connections and interactions between players during a game. Initially, the *init* function is called to initialize the WebSocket connection and set the initial state of the server. Subsequently, the *websocket_handle* function handles messages sent by clients, decoding the requested *action* and coordinating responses based on the current state of the game.

- ***init/1***: called whenever a request is received, to establish a websocket connection and to initialize the state of that erlang process.
- ***websocket handle/2***: called whenever a message frame arrives from the client. It will handle the reception and deserialization of the JSON objects coming from the client.
- ***websocket info/2***: called whenever an Erlang message arrives (i.e. a message from another Erlang process).

2) **player_handler** *module*: handles the behaviors and engagements of individual ***players*** throughout a game. It contains functions enabling players to ***input words*** or and ***sentences*** as generic messages to chat with friend during a match, make ***guess a word*** **attempt**, and ***await actions*** from other players

Each player corresponds to a relative *process* on the server, the process is registered with the username of the player to which it refers, so that the other players in the game can communicate with this process only by knowing the username of the player to whom it refers. So, the player username must be unique for each user.

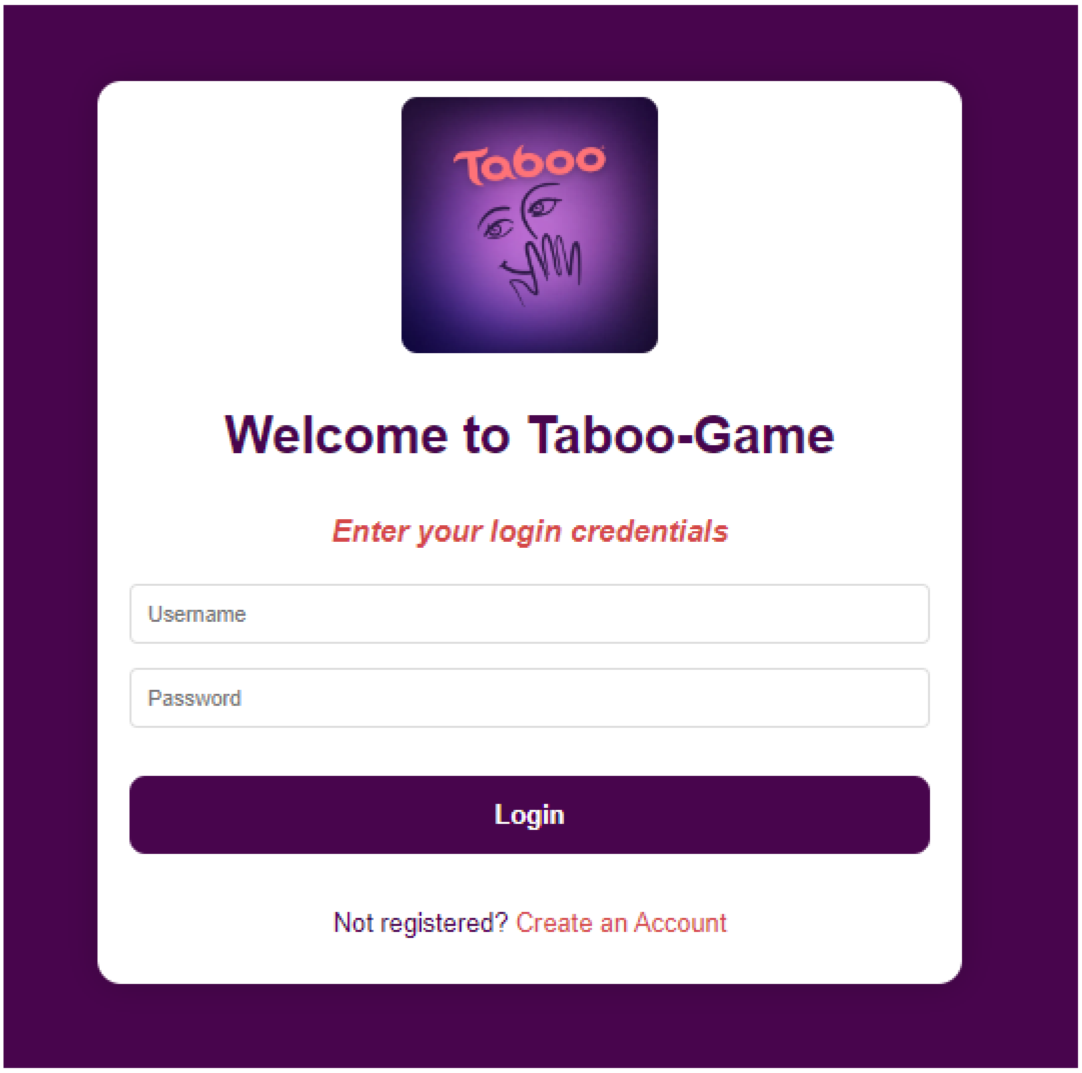
The usernames of the players in the game are information that each player has previously received from the Spring Web Server, in the login phase.

User Manual

The aim of this *User Manual* is to guide the user to proper use of the **Taboo-Game Web Application** by analyzing in detail what happens when interacting with the components of its *Graphical User Interface*.

Login-Welcome Page

This page is common to *Unregistered*, *Registered* and *Admin User*. This is the web application root page that allows a registered user to login in the application by inserting its credentials and by clicking on ***Login*** button. Moreover, by clicking on ***Create New Account*** text-button the user will be redirected to *Sign-Up Page.*

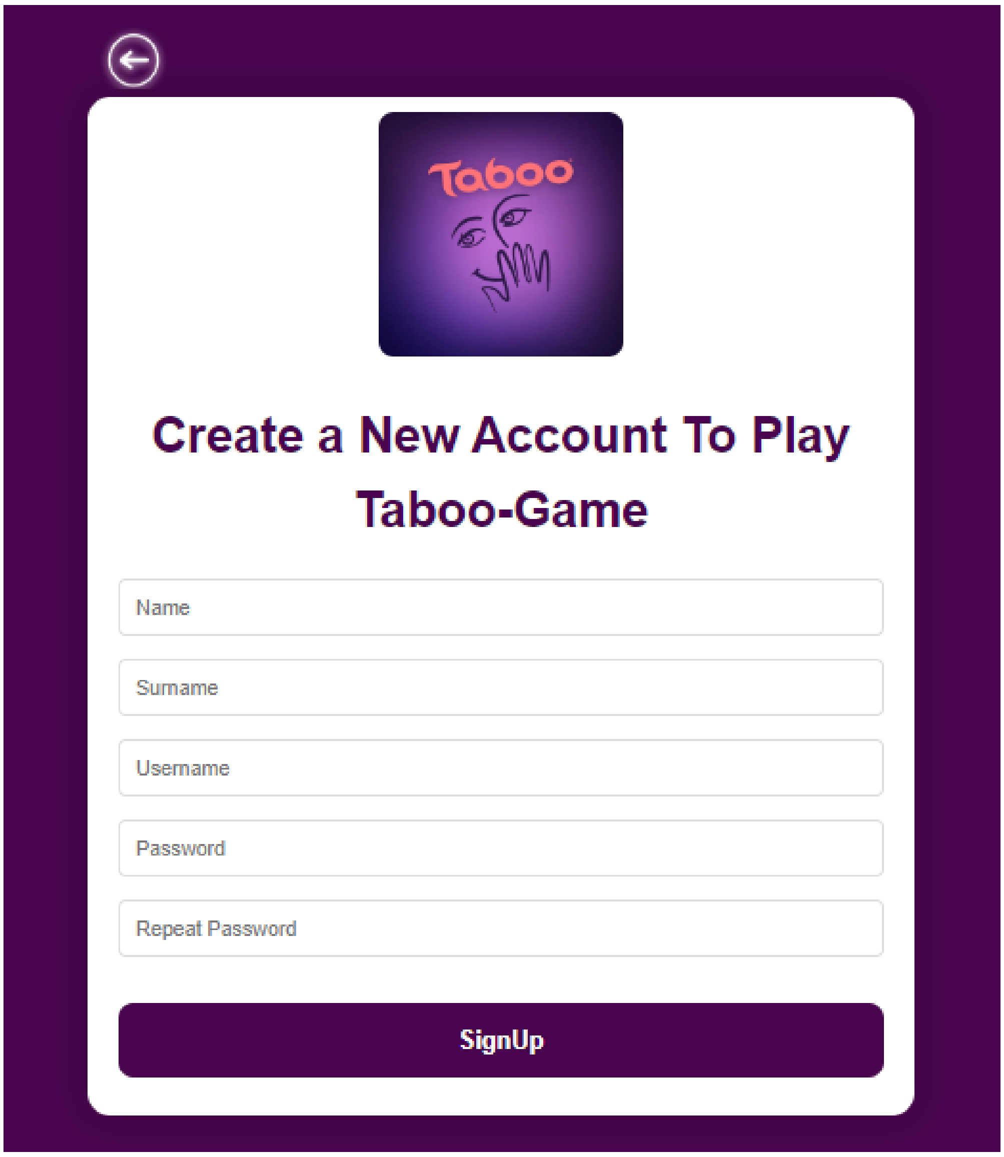

Taboo
Welcome to Taboo-Game
Enter your login credentials
Username
Password
Login
Not registered? Create an Account

Unregistered User

Once the user clicks on *Create New Account* button from *Login-Welcome Page*, previously explained, the *Unregistered User* can insert its personal basic information to subscribe a new account on Taboo-Game WebApp as shown in the image below.

Create New Account Page

This is the web page that allow a Not-Registered user to create a new account for the ***Taboo-Game WebApp*** and to play the game. In the page is shown a list of fields that the user must fill with its personal information to sign up in the system. After insert all the needed information, there will be checked if the inserted *username* is already used. In that case the user will advise to change it to complete the sign-up step.



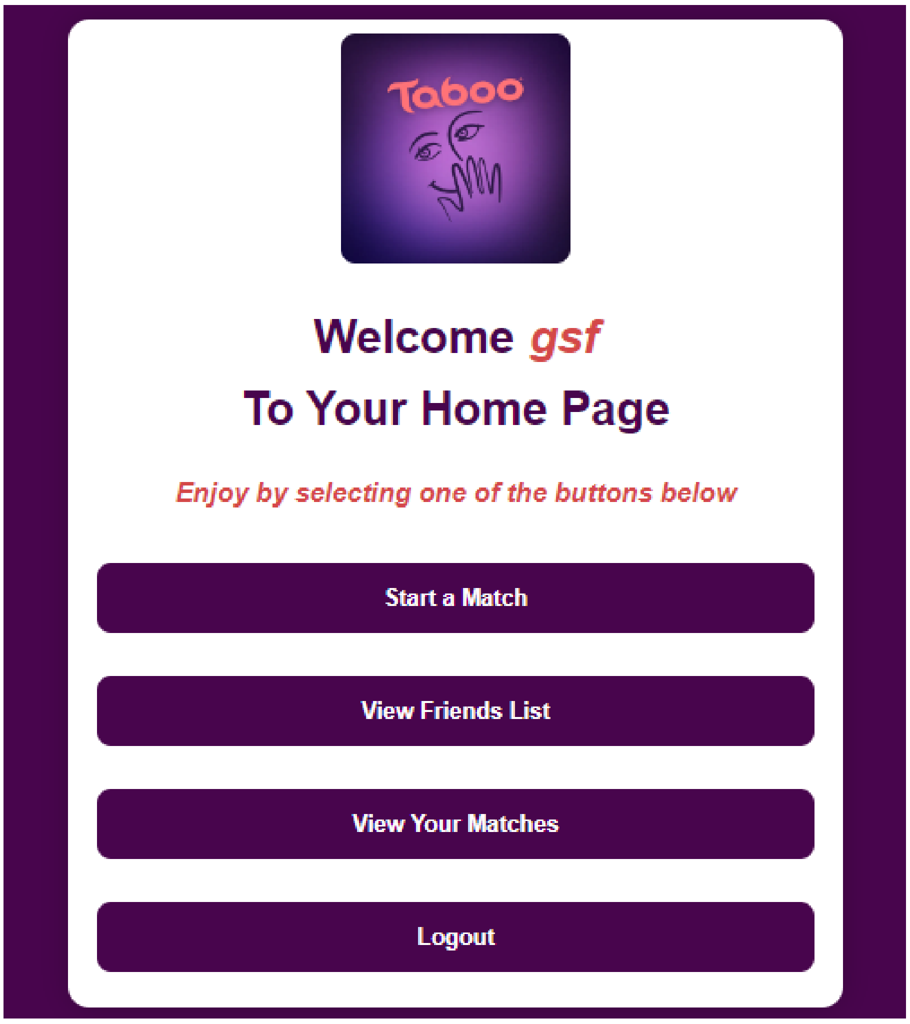
Registered User

Once a *Registered User* inserted the credentials *username* and *password*, by clicking on *Login* button from *Login-Welcome Page* previously explained, the User can visualize its Home Page and navigate the application by clicking on different button that will be address more in detail in the next section of this chapter.

Below an overview of the Home Page.

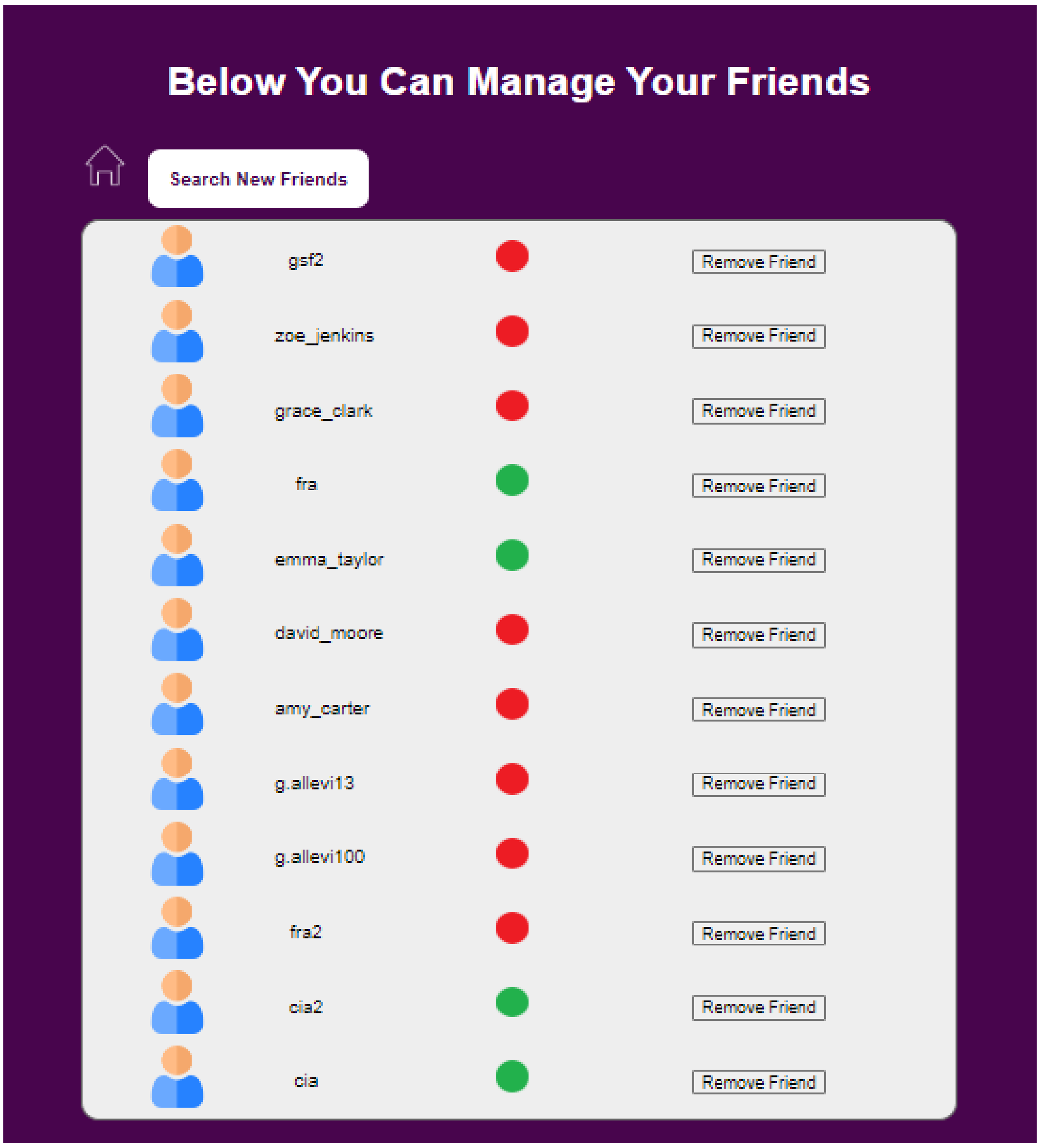
Home Page

When the user completes the login phase he will be redirected to their ***Home Page***. This page represents the starting point for the *User* to navigate the application to perform its activities. Specifically by clicking on the ***Start a Match*** button he will be able to create a new match, by clicking on ***View Friend List*** button he will view its actual friends list, by clicking on the ***View Your Matches*** button he can visualize its history matches and by clicking on the ***Logout*** button the user will leave the Game.



Friend List Page

When the logged user enters in its Friend List Page, he can visualize the list of users he has as friends ***Taboo-Game Web Application***. se its friend list. In the friend list, for each of them, it will be displayed its *username*, the *online/offline* information and a ***Remove User*** button, located next to the username, that allows the logged user to remove that specific friend from its friend list. By clicking on the button represented by a ***Home icon*** located in the top-left will be redirected to his *Home Page*, by clicking instead on the ***Search New Friends*** button, also located in the top-left, he will be redirected to the *Search New Friends* page where he will be able to perform a search among the global users registered in the Web Application.

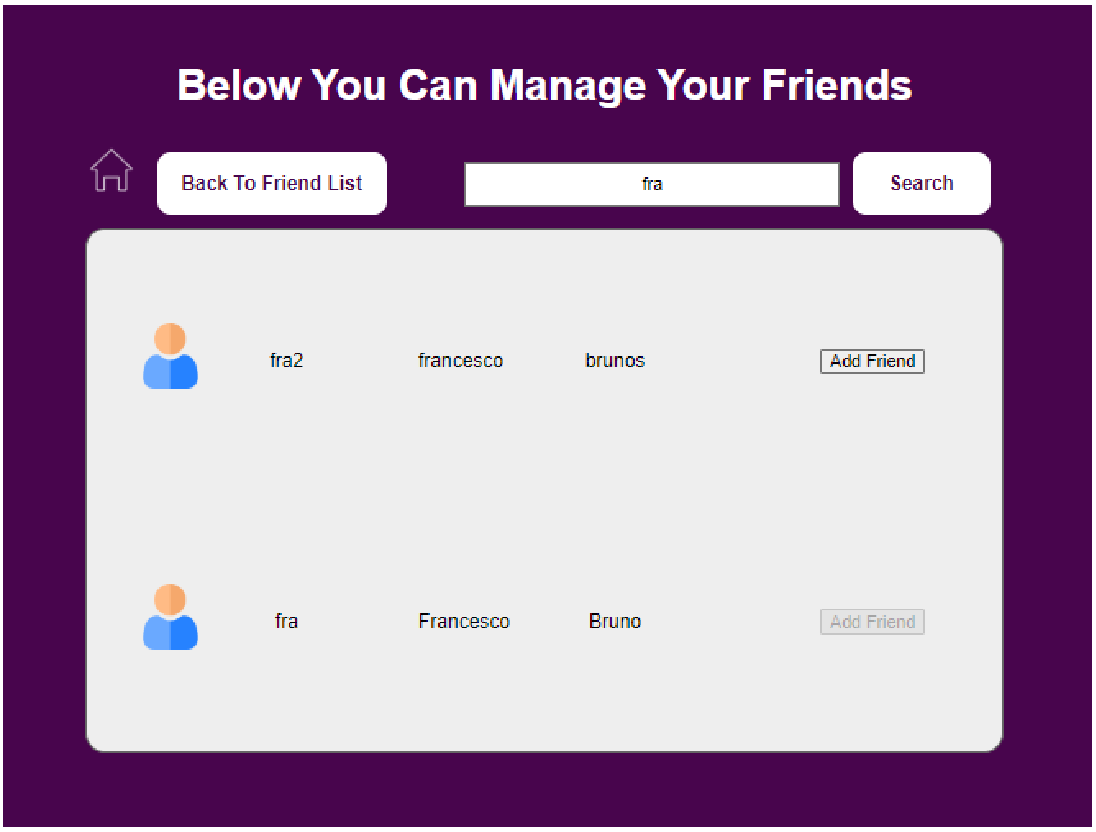


Below You Can Manage Your Friends

gsf2	🔴	Remove Friend
zoe_jenkins	🔴	Remove Friend
grace_clark	🔴	Remove Friend
fra	🟢	Remove Friend
emma_taylor	🟢	Remove Friend
david_moore	🔴	Remove Friend
amy_carter	🔴	Remove Friend
g.allevi13	🔴	Remove Friend
g.allevi100	🔴	Remove Friend
fra2	🔴	Remove Friend
cia2	🟢	Remove Friend
cia	🟢	Remove Friend

Search New Friend Page

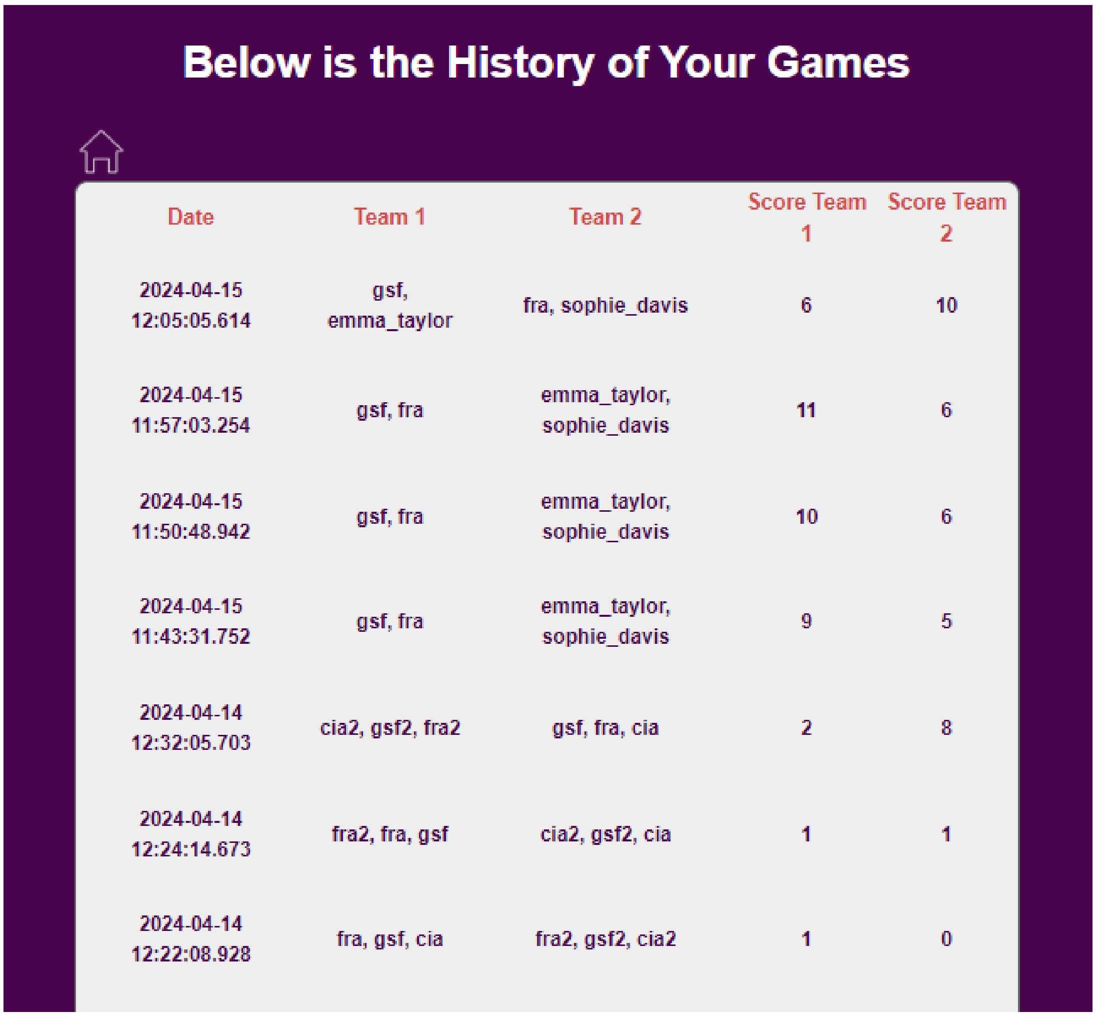
In this page, the *User* can perform a search for users by filtering them in the Taboo-Game web application by the users' *username* to obtain a filtered list of people that he may want to add to its friend list. Specifically he can insert the *username* in the text-box located in the top-right and then by clicking ***Search*** button or simply push the *Enter Kye* by keyboard of PC. By clicking the ***Add Friend*** button located next to the *username* of a specific user into the filtered list the *user* will be able to add them in its friend list creating a new friendship. By clicking on the button represented by a ***Home icon*** located in the top-left will be redirected to his *Home Page*, by clicking instead on the ***Back To Friend List*** button, also located in the top-left, the admin will be redirected to the its *Frind List* page previous explained.



fra2	francesco	brunos	Add Friend
fra	Francesco	Bruno	Add Friend

Matches Played Page

In this page, the logged user can see the list of its past matches played. For each these past matches, the player can see the *Date,* the *team1* and *team2* members, and the *Score* of each of team. By clicking on the button represented by a ***Home icon*** located in the top-left will be redirected to his *Home Page.*



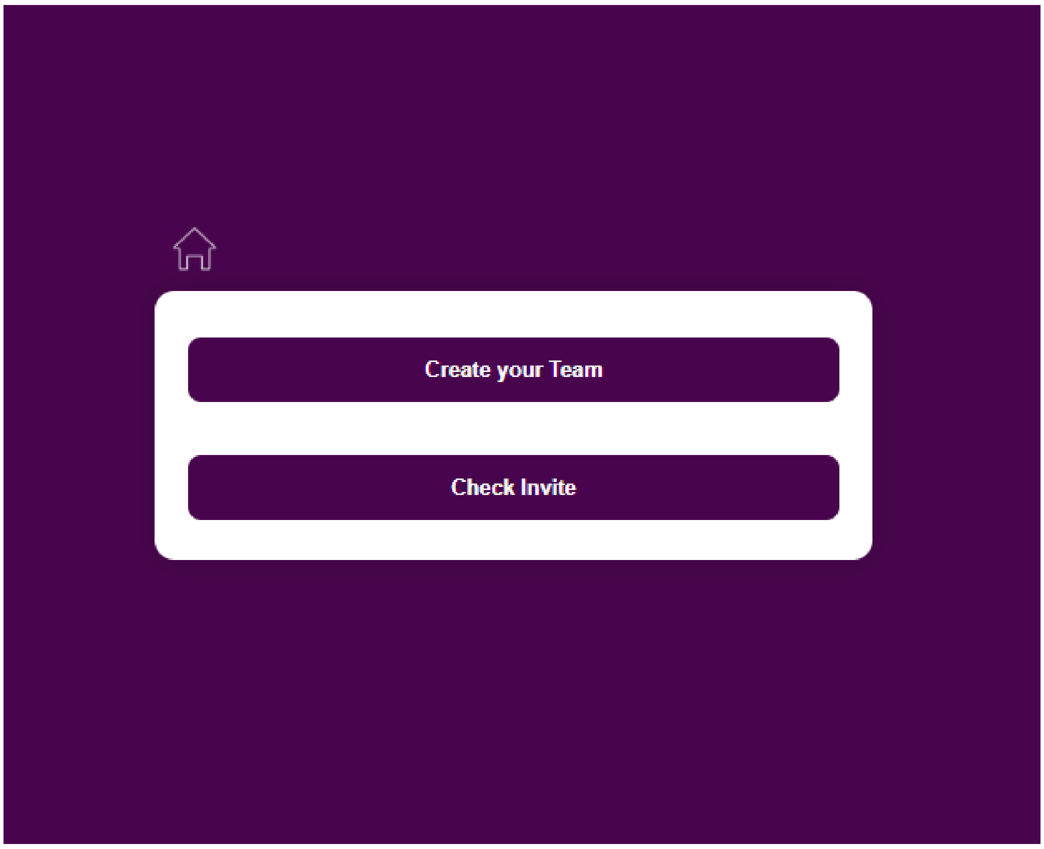
Below is the History of Your Games

Date	Team 1	Team 2	Score Team 1	Score Team 2
2024-04-15 12:05:05.614	gsf, emma_taylor	fra, sophie_davis	6	10
2024-04-15 11:57:03.254	gsf, fra	emma_taylor, sophie_davis	11	6
2024-04-15 11:50:48.942	gsf, fra	emma_taylor, sophie_davis	10	6
2024-04-15 11:43:31.752	gsf, fra	emma_taylor, sophie_davis	9	5
2024-04-14 12:32:05.703	cia2, gsf2, fra2	gsf, fra, cia	2	8
2024-04-14 12:24:14.673	fra2, fra, gsf	cia2, gsf2, cia	1	1
2024-04-14 12:22:08.928	fra, gsf, cia	fra2, gsf2, cia2	1	0

Start a Match Page

When the user enters in this page when click on the ***Create your Team*** button he will be redirected in the relative *Create Your Team Page* where he can create its team by selecting its friends as team member to play a game, or can visualize via an alert incoming invites if present. By clicking on ***Check Invite*** button he will be able to *check for an incoming Invites* to participate as a *Friend* or as a *Rival* in a new Match.

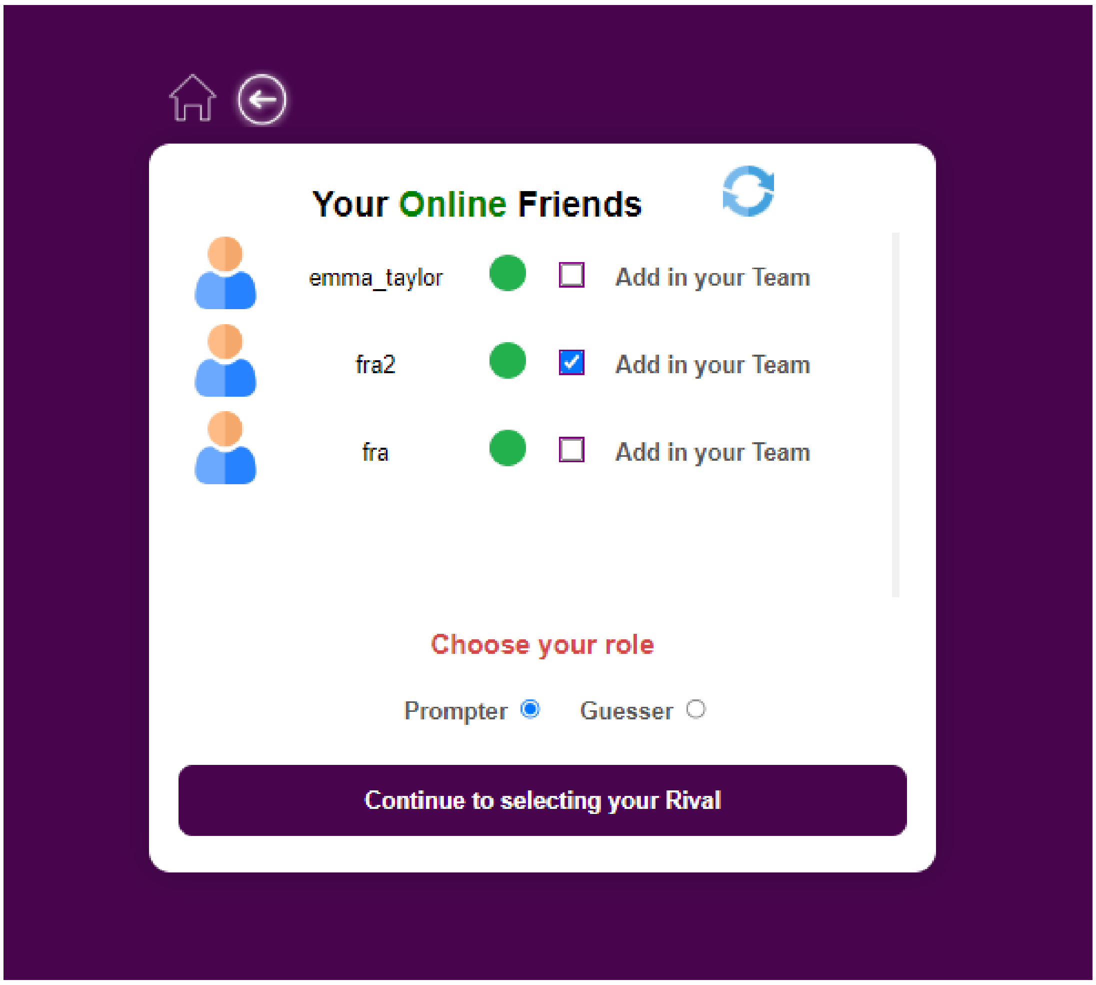
So, if the user has been invited, it will be alerted with a message, and he can choose to confirm or reject the incoming invite. If the user accepted the invite, it will be redirected to the *Waiting Page*, because it’s necessary wait all the other match members. Otherwise, if the user rejects the invite, all the other waiting users must be advised of the rejection.



Create Your Team Page

When the player access to this page, it will be displayed the list of its online friend. To create its team, the *Inviter-User* can choose its friend by checking the checkbox next to the wanted friend. Before to select its Rival friend, we have decided to give the user the possibility to choose its match role (*Prompter* or *Guesser*) by selecting one of two choices by clicking on ***radio button***. After that, the user can continue the match creation and he will be redirected to *Select Rival Page* by clicking on ***Continue to selecting your Rival*** button.

By clicking on the button represented by a ***Home icon*** located in the top-left will be redirected to his *Home Page* and by clicking on the button represented by ***Back-Arrow icon*** located in the top-left as well will be redirected to the *Start a Match Page* previously explained. By clicking on the button represented by a ***Circle-Arrows icon*** located in the top-right he will be able to refresh the list of online friends.


Your Online Friends

emma_taylor	●	☐	Add in your Team
fra2	●	☑	Add in your Team
fra	●	☐	Add in your Team

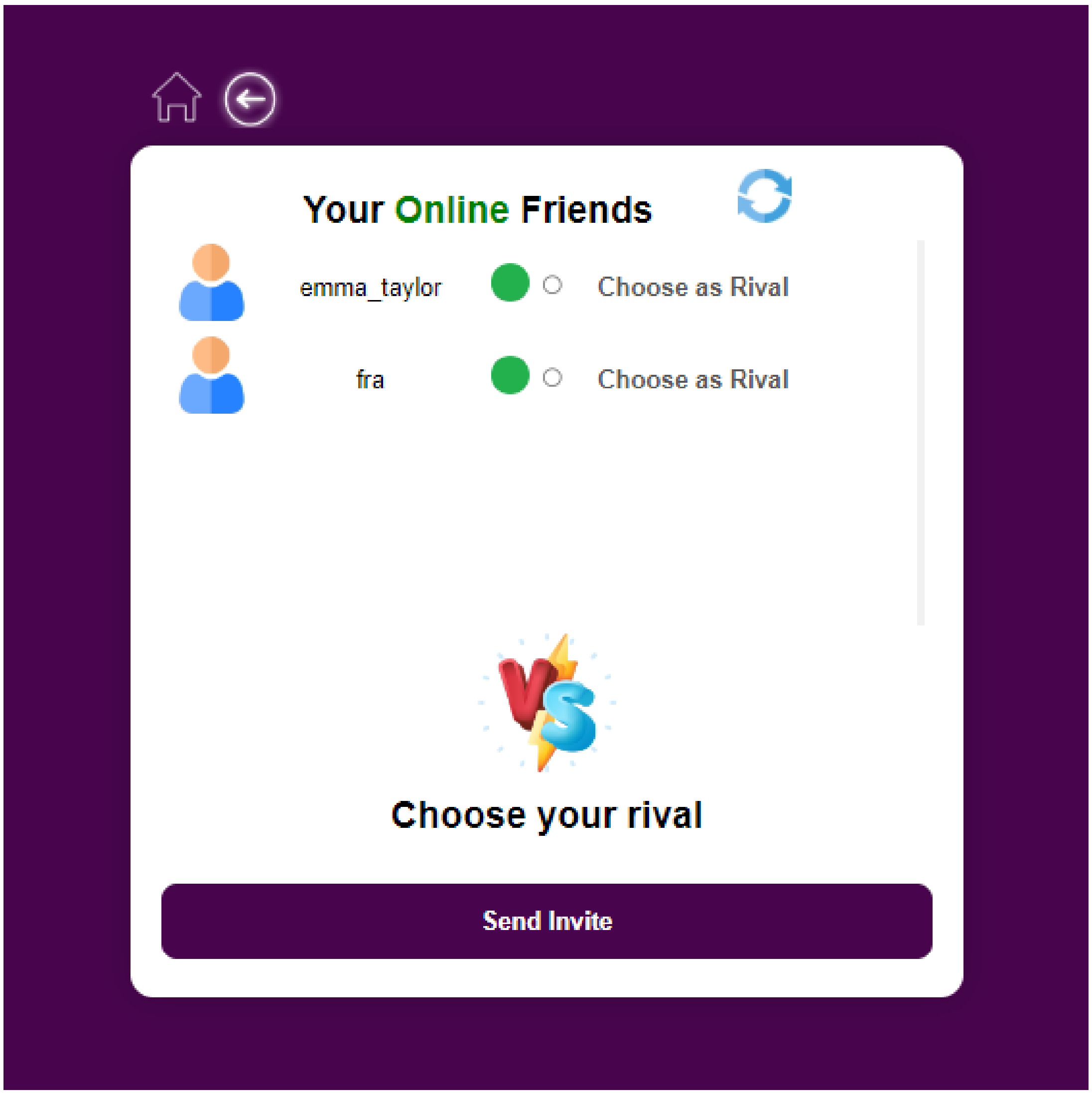
Choose your role

[✓ Prompter] Guesser

Select Rival Page

After the creation of its team, the *Inviter-User* will be able in this page to choose only one member of the rival team. The reason why is associated to give the opportunity to selected *Rival-User* to create its team by selecting the member in his own way.

Once the *Inviter-User* click on the ***Send Invite*** button he will be redirected in the *Waiting Page* in which await that all users accepted the invitation to start with the game session.


Your Online Friends

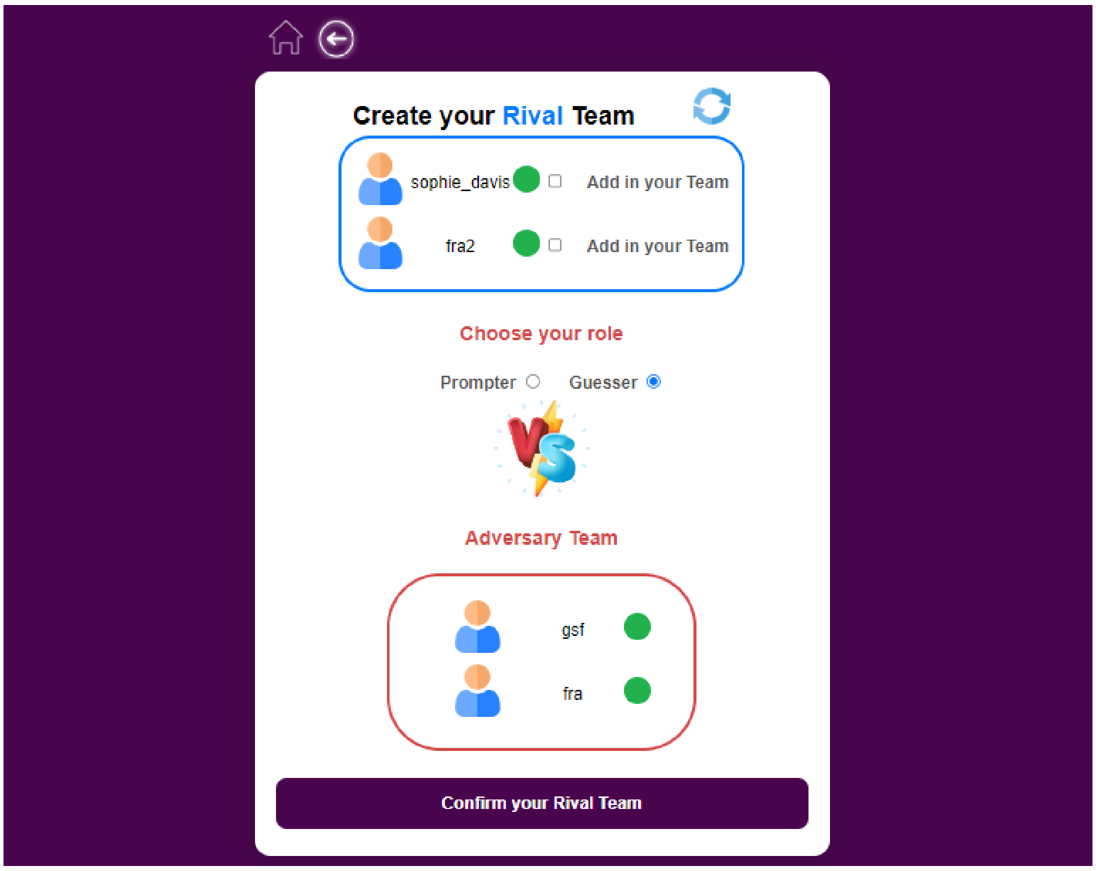
emma_taylor	● ○	Choose as Rival
fra	● ○	Choose as Rival

Choose your rival

Create Rival Team Page

This page is accessible only in case this specific user has been invited as *Rival-User* from another friend. In that case, as mentioned before, this user can create its team by selecting his favorite friend in the same way made by the *Inviter-User.*

Once the *Rival-User* click on the ***Confirm your Rival Team*** button he will be redirected in the *Waiting Page* in which await that all users accepted the invitation to start with the game session.


Create your Rival Team

sophie_davis	● ☐	Add in your Team
fra2	● ☐	Add in your Team

Choose your role

Prompter ○ [✓ Guesser]

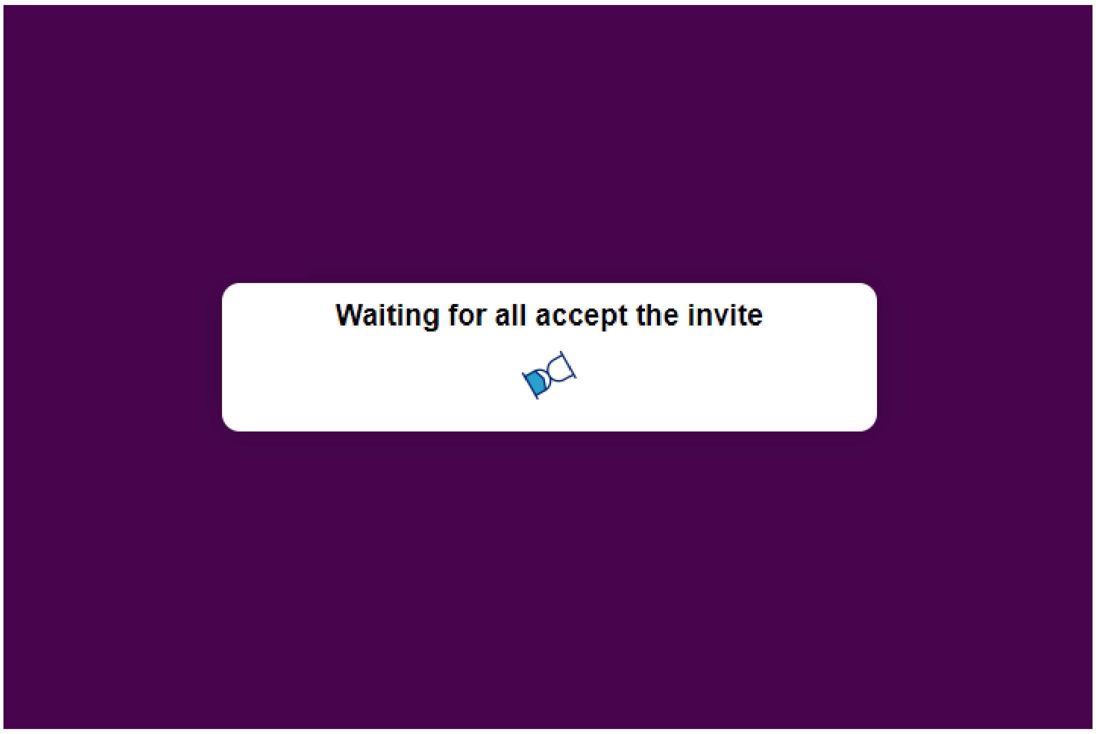
Adversary Team

gsf	●
fra	●

Confirm your Rival Team

Waiting Page

Every time a user receives an invite, and he accepts that invite, he will be redirected to this *Waiting Page*, to waits all the other invited users. If all invited users accept the invitation, then all the users can access to *Taboo-Game page.* Otherwise, if at least one user rejects the invitation, then all the users are redirected to its main page.


Waiting for all accept the invite

Taboo-Game Page (Prompter)

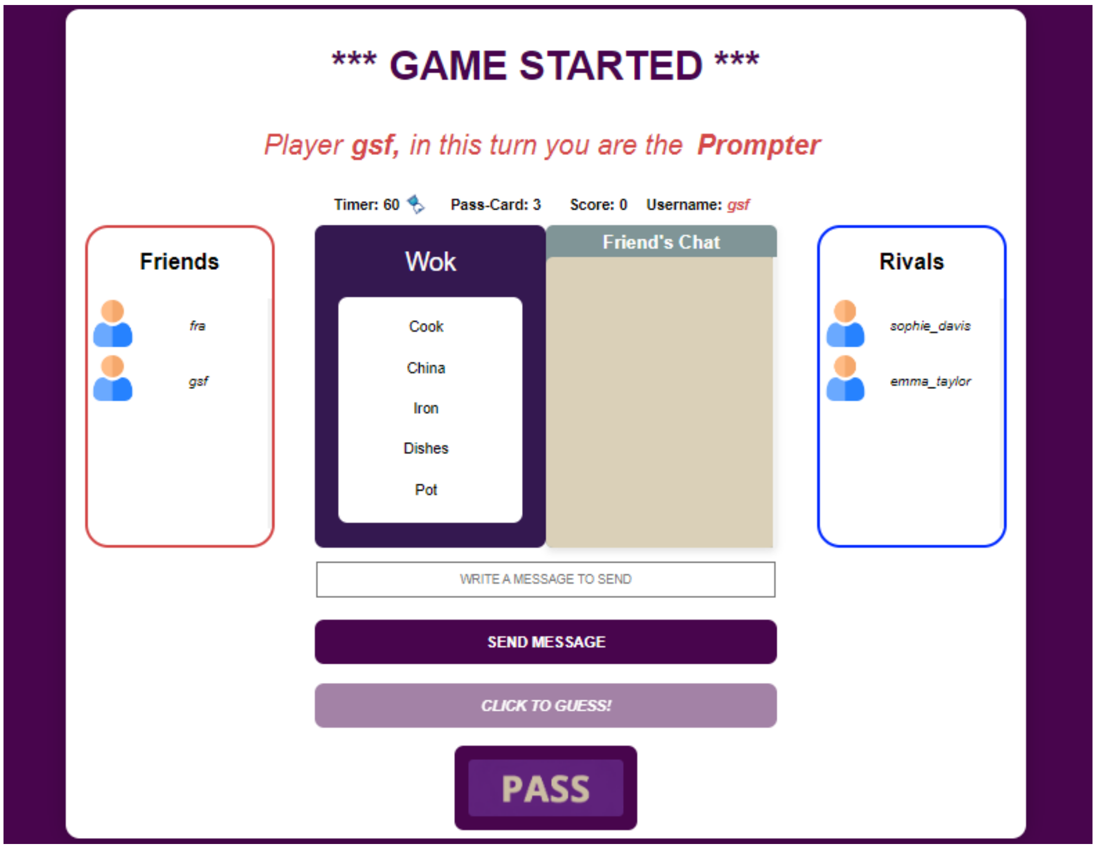
As said before, if all users accept the invite, then the match can start and all the users move to this *Taboo-Game Page.* At top of the page status information are available like *username* and actual *role* of the player, which in this case we supposed to be the ***Prompter***.

At the center pager, as Prompter, you can see the actual ***Taboo-Card***, composed by the word that you have to make guess by your friends and the ***Five-Taboo-Words*** that represent the prohibited words, the latter you must not use when trying an attempt to get guessers to guess the word. In order to make guess the friends, you can write into text-box what you think could be useful to help your friend guesser by sentences and press the Enter Key in your keyboard of your PC or click on the ***Send Message*** button to send that message to your team member.

On the right and left sides of the page challenger teams are shown, more precisely on the left is your team, on the right the Rival team.

Your objective is to make guess as much words as possible in 60 seconds in order to increment your team's score, this timer represent also the time after which the change of the role from ***prompter*** to ***guesser*** and vice versa will occur. If you are good enough to make guess the secret word, your ***team's score*** is automatically incremented, and you will immediately receive a new *Taboo-Card.* But, if the word to make guess is difficult, by clicking on the ***Pass*** button you'll receive a new *Taboo-Card*, but it's important take into account that you have only **<u>3 available pass</u>** for each turn

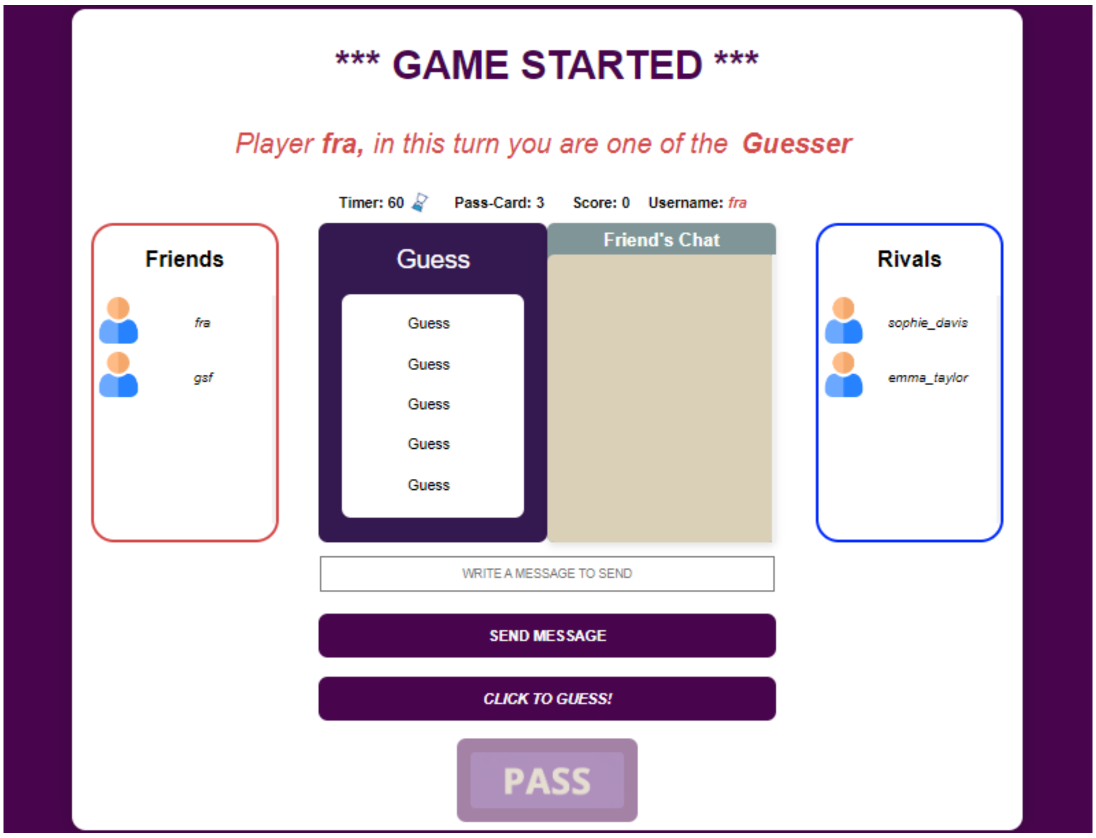
Be careful to not use none of *Taboo-Words*, otherwise your team's score it will be decremented, and the *Taboo-Card* will be changed.



Taboo-Game Page (Guesser)

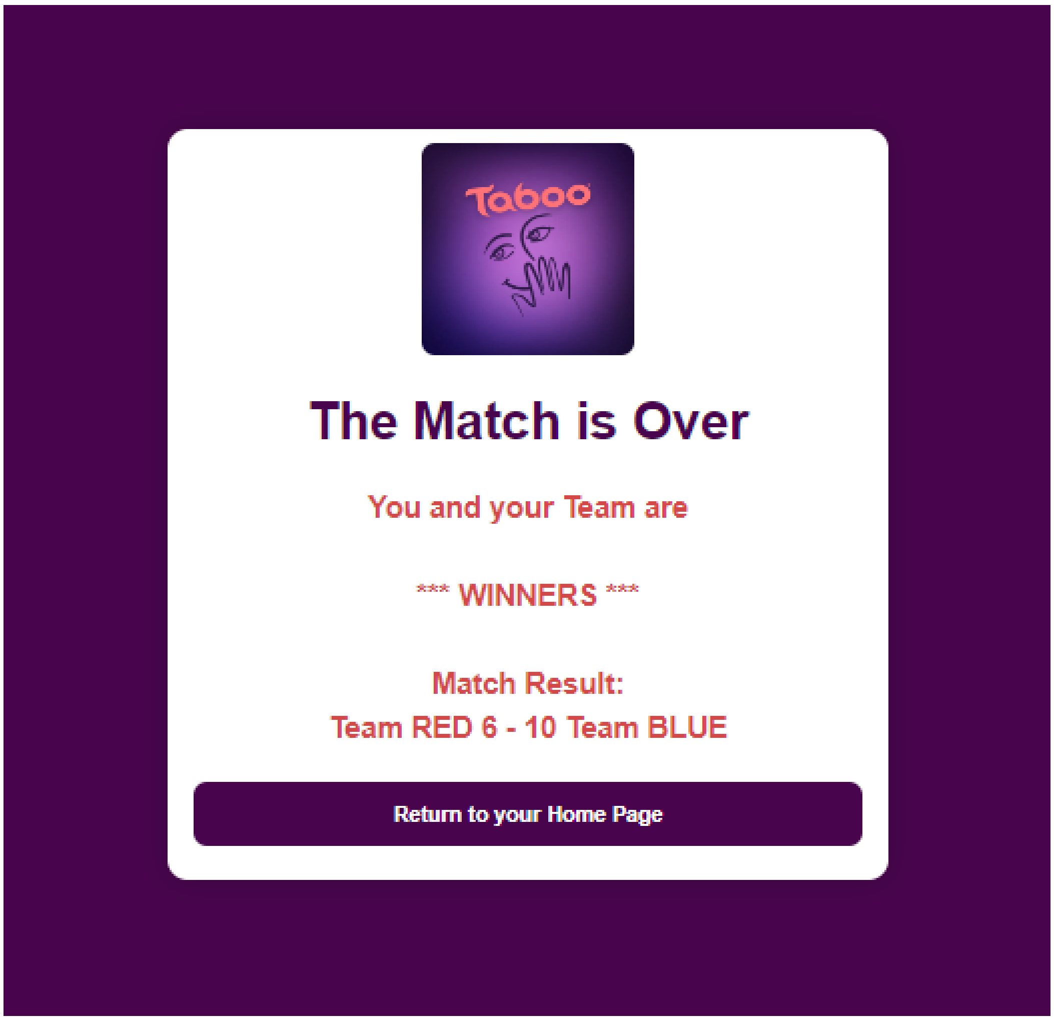
The view of this page, when at the end of the turn your role switched from Prompter to Guesser or if you're in general accessed to this page as guessers at first time, is similar to the previous page for the Prompter with some difference that lies in the *Taboo-Card*, which you will not see as a guesser, and moreover you cannot click on ***Pass*** button.

Your objective now is to guess as much words as possible. You can send a message to all your team members to ask him any hint by writing a message into text-box and send it by pressing the Enter Key in your keyboard of your PC or by clicking on the ***Send Message*** button. If you think to have guessed the secret word, you just write it in the text-box and click on ***Click To Guess!.*** If you are correctly guessed the word, the team's score is automatically incremented, otherwise you must think another word because the clock is ticking!



End-Game Page

In this page of the application the User can visualize the final result of the Match. Specifically, if at the end with its team is *Winner* or *Loser*. By clicking on the ***Return to your Home Page*** button the user will be able to return in its *Home Page* and start over the navigation of the application as it prefers.



The Match is Over

You and your Team are

*** WINNERS ***

Match Result:
Team RED 6 - 10 Team BLUE

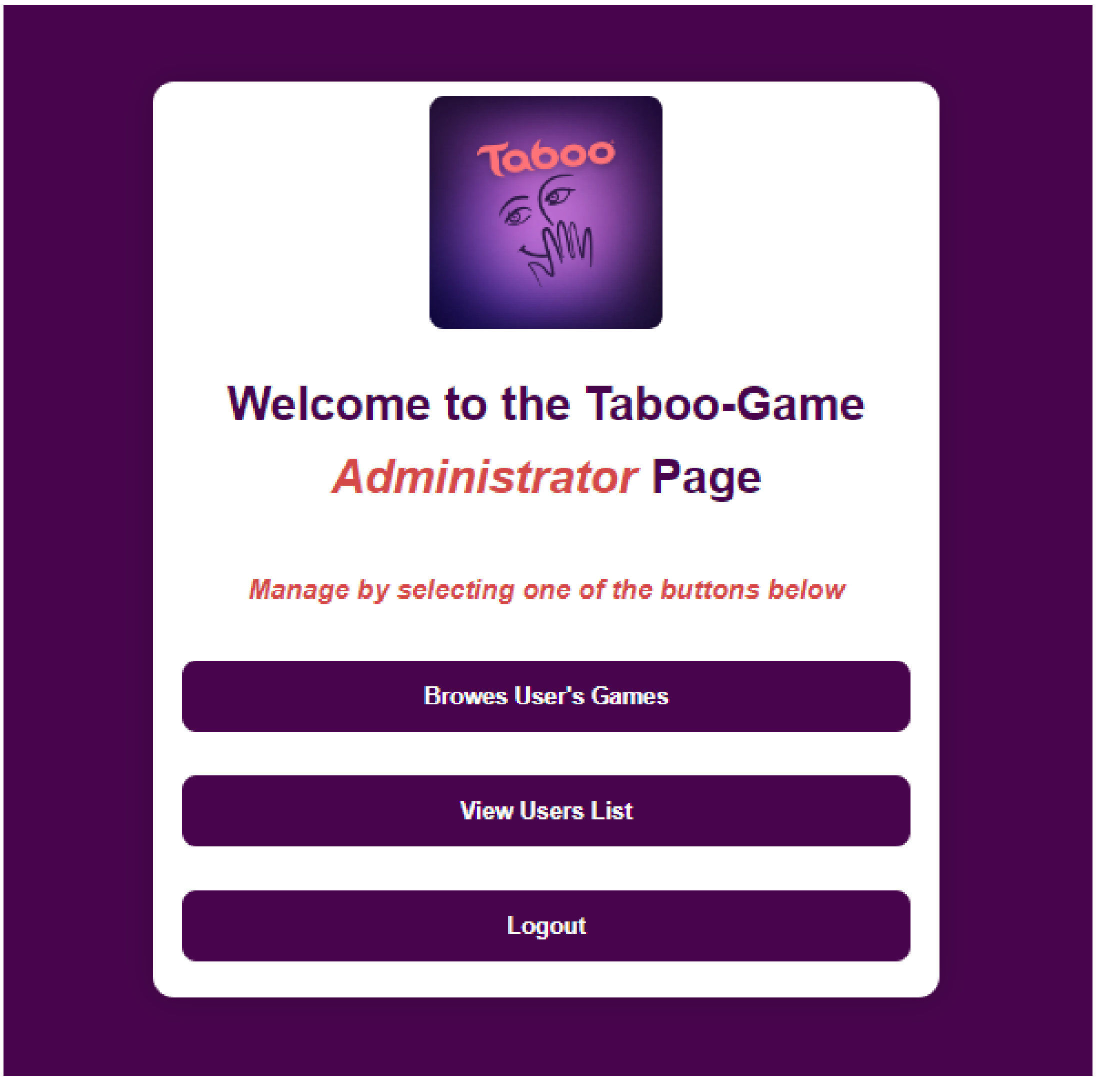
Admin User

The *Admin* is a special user who, once logged-in from the Login page previously explained, within the application can manage users registered in the Taboo-Game platform thanks to elevated privileges

Below an overview of the Home Page.

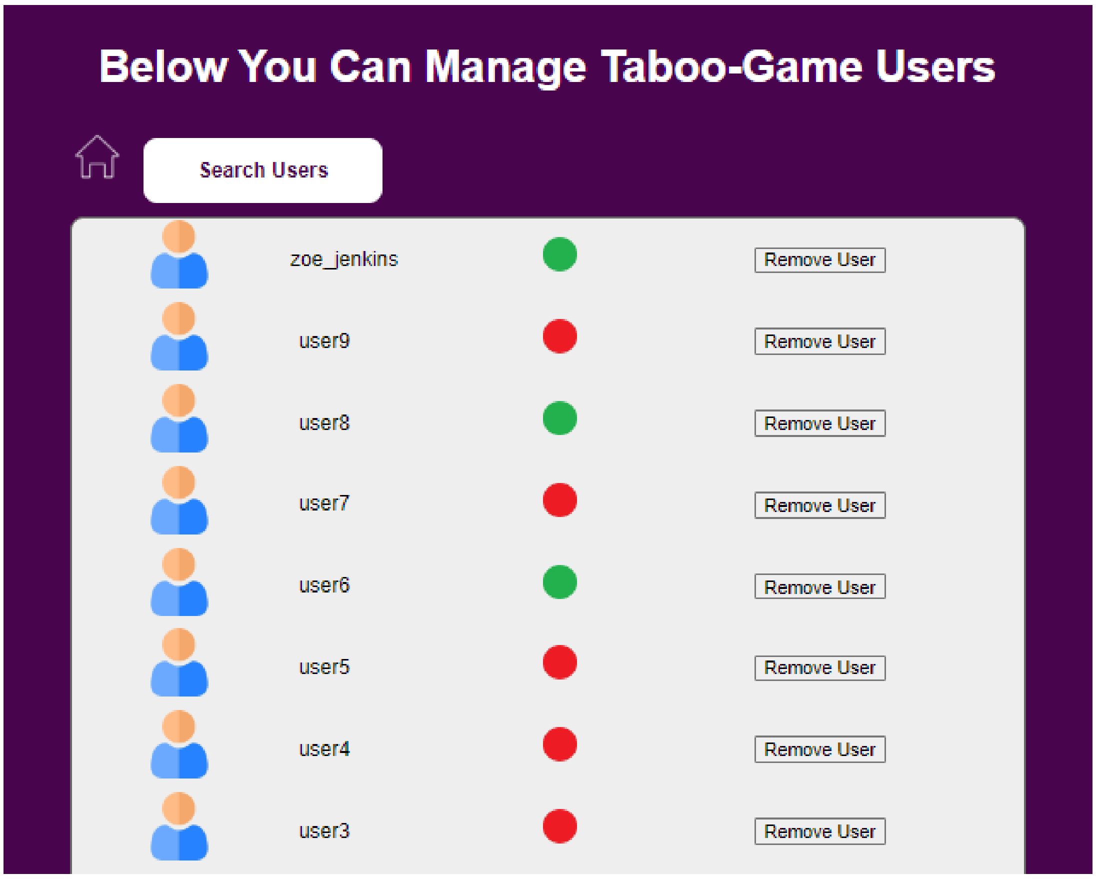
Home Page

This page represents the starting point for the *Admin User* to navigate the application to perform its activities. Specifically, by clicking on the ***Browes User's Game*** button and ***View Users List*** button the Admin will be able to manage the Users and matches played by the latter. By clicking on the ***Logout*** button the Admin will leave the Game.


Taboo
Welcome to the Taboo-Game
Administrator Page
Manage by selecting one of the buttons below
Browes User's Games
View Users List
Logout

Registered Users List Page

In this page the *Admin* displays the list of *Registered-Users* in Taboo-Game Web Application. By clicking on the ***Remove User*** button located next to the username of a specific user into the list the *Admin* will be able to delete them by canceling their registration as a user of the platform. By clicking on the button represented by a ***Home icon*** located in the top-left will be redirected to his *Home Page*, by clicking instead on the ***Search User*** button, also located in the top-left, the admin will be redirected to the *Search User* page where he will be able to perform a search among the users.

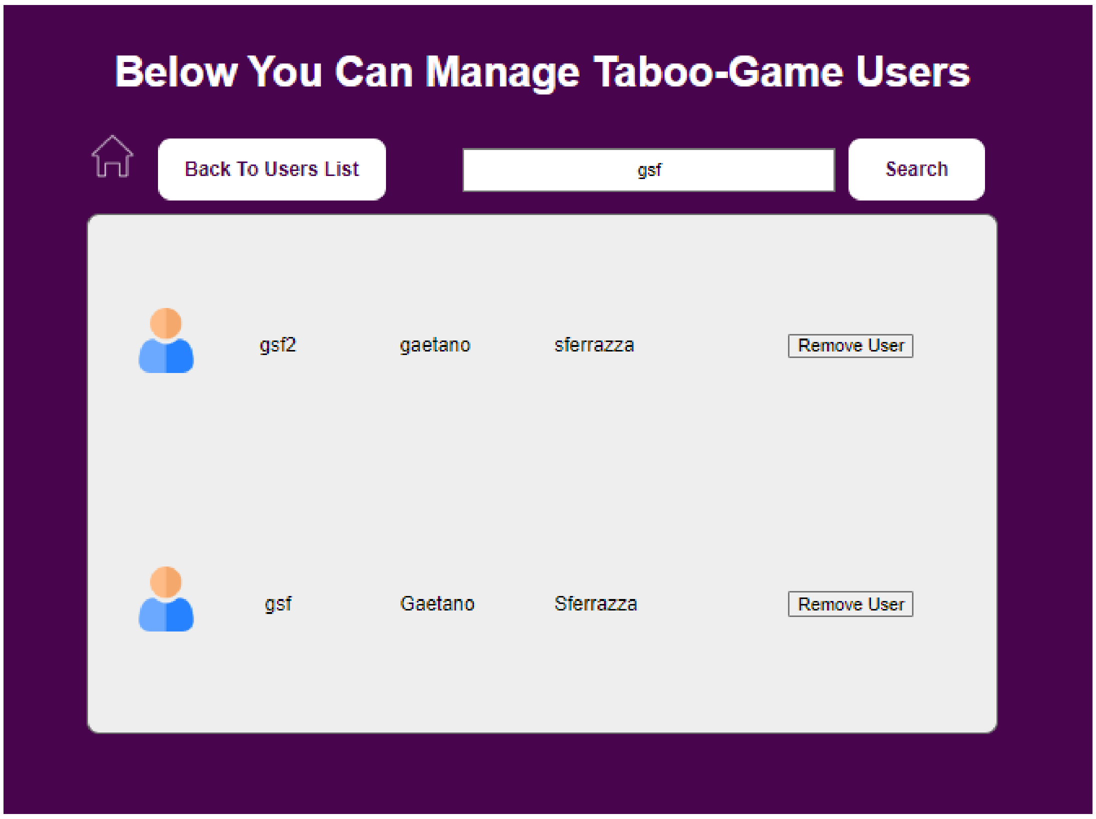

Below You Can Manage Taboo-Game Users

zoe_jenkins	🟢	Remove User
user9	🔴	Remove User
user8	🟢	Remove User
user7	🔴	Remove User
user6	🟢	Remove User
user5	🔴	Remove User
user4	🔴	Remove User
user3	🔴	Remove User

Search Users Page

In this page, the *Admin* can perform a search for users by filtering them in the Taboo-Game web application by the users' username to obtain a filtered list. Specifically he can insert the *username* in the text-box located in the top-right and then by clicking ***Search User*** button or simply push the *Enter Kye* by keyboard of PC.

By clicking the ***Remove User*** button located next to the *username* of a specific user into the filtered list the *Admin* will be able to delete them by canceling their registration as a user of the platform. By clicking on the button represented by a ***Home icon*** located in the top-left will be redirected to his *Home Page*, by clicking instead on the ***Back To User List*** button, also located in the top-left, the admin will be redirected to the general *User List* page previous explained.

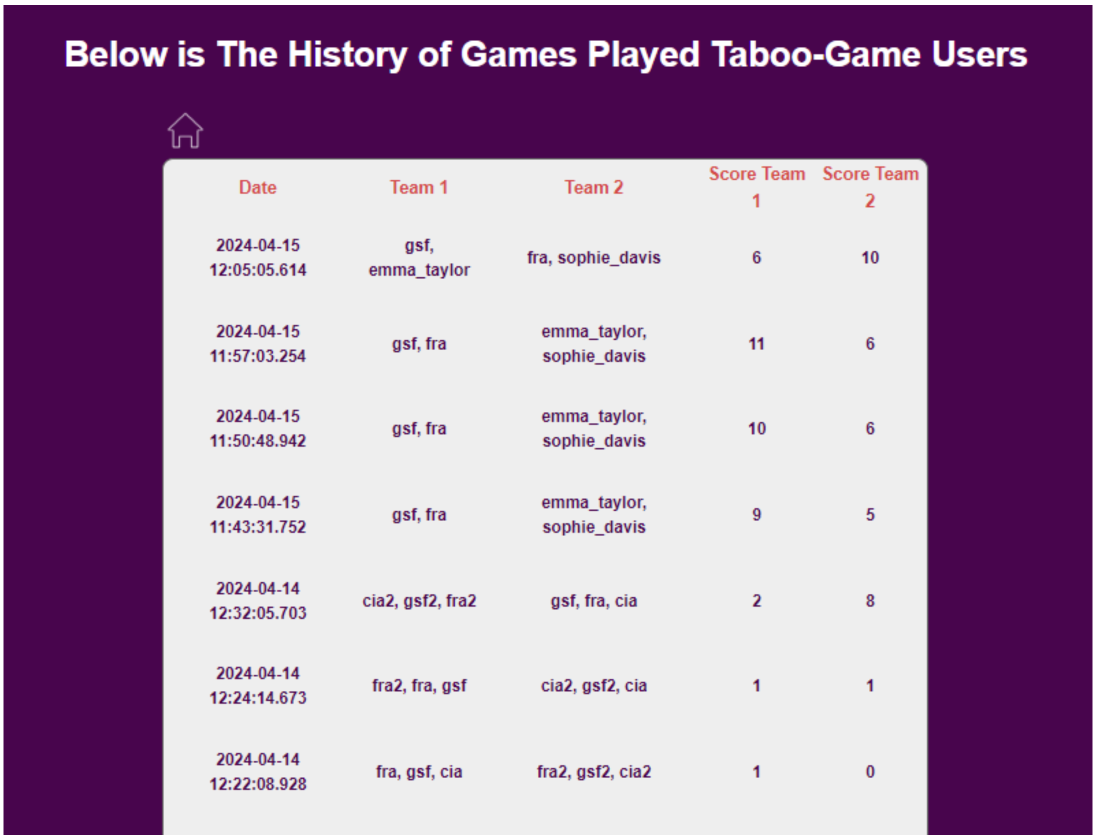


Below You Can Manage Taboo-Game Users

	gsf2	gaetano	sferrazza	Remove User
	gsf	Gaetano	Sferrazza	Remove User

Matches Played by Users Page

This page represents the history of the all played matches, visible only by the *Administrator.* This page is very similar to the page accessible by every user, in fact are showed the same information, like the *Date*, the *Team1*, the *Team2* members and the two relative result of that match. By clicking on the button represented by a ***Home icon*** located in the top-left will be redirected to his *Home Page.*



Below is The History of Games Played Taboo-Game Users

Date	Team 1	Team 2	Score Team 1	Score Team 2
2024-04-15 12:05:05.614	gsf, emma_taylor	fra, sophie_davis	6	10
2024-04-15 11:57:03.254	gsf, fra	emma_taylor, sophie_davis	11	6
2024-04-15 11:50:48.942	gsf, fra	emma_taylor, sophie_davis	10	6
2024-04-15 11:43:31.752	gsf, fra	emma_taylor, sophie_davis	9	5
2024-04-14 12:32:05.703	cia2, gsf2, fra2	gsf, fra, cia	2	8
2024-04-14 12:24:14.673	fra2, fra, gsf	cia2, gsf2, cia	1	1
2024-04-14 12:22:08.928	fra, gsf, cia	fra2, gsf2, cia2	1	0

Future Works

We have considered future enhancements to apply to our work done during the development of our version of ***Taboo-Game Web Application*** (*Distributed Platform*). Specifically, the use of a potential ***Load Balancer*** in case we want to evaluate the possibility of using multiple containers for the Erlang server, thus ensuring a fair distribution of the load; the implementation of a solution that allows ***saving game session information on the Serve-Side*** (*Spring*).

This would result in an improvement in terms of *scalability*, *performance*, and *security* of the Web Application.

Session Game Tracking on Application Web Server

The ***Server-Side*** implementation of a tracking/saving system for information regarding the gaming session would ensure greater security and prevent potential manipulation of information exclusively stored ***Client-Side*** by malicious users. Such a solution could involve logging the messages exchanged between the *Prompter* and *Guesser* of the same team during the word-guessing phase.

This would result in a historical record of these messages that could be used by the server to verify that the final outcome of a game, for example, is indeed the one observed.

Load Balancer

If We choose to increase the number of Erlang servers, we need to consider the use of a Load Balancer. The latter can be a *device* or *software* and has the characteristic of distributing network traffic or requests among multiple servers or computing resources, with the main goal of improving system reliability, efficiency, and scalability, ensuring that all resources are used fairly and optimally. One of the most common Load Balancers is, for example, ***NGINX*** which can be used to distribute *HTTP* traffic among different web servers, thus improving the overall system performance and allowing for efficient handling of a greater number of client requests. Another aspect to consider when choosing to use a Load Balancer concerns the *consistency of information* for requests in relation to the various Erlang servers residing on different distributed nodes. These must be supported by a distributed data storage system that allows them to store and retrieve data. In this way each Erlang server can access the data consistently and in a shared manner, regardless of its physical location in the cluster, allowing the Load Balancer to correctly distribute requests and responses among the servers. Examples of systems that provide this service are ***Redis Cluster*** and ***Mnesia***.

References

Our work can be found and accessed at the following GitHub link:

- https://github.com/francescoB1997/DSMT_Taboo