JAR Chess Game Use Case Survey

# Actor Summaries

## Player

Players will use the game to play matches and access saved historical matches.

* Speed - The application needs to be responsive or players will find other games to play.
* Ease of use - The UI needs to be intuitive. All components of the application need to be easily understood and managed without training or external documentation. Most players won’t bother putting that much effort to learn to use the app.
* Security - While there is not much for sensitive data, the app needs to protect players’ username\password combinations and email addresses.
* Privacy – Players don’t want their information in the hands of anyone they didn’t give it to, and they won’t want us monitoring their location or other private data without cause and consent.
* Availability – If any aspect of the game is not available at any given time, that will make the players look elsewhere for entertainment.

## Opponent (secondary actor)

The opponent is entity that the player is playing against.

# Use Case Summaries

## Login to Account

The goal of this use case is to allow a logged-out player to login to an account. The player tells the system they wish to log in. The system asks for the player’s login credentials. The player provides the login credentials. The system verifies that the credentials are linked to an account. The system will then log the player in.

## Manage Account

The goal of this use case is to allow a logged-in player to view and make changes to their account. The player tells the system they want to manage their account. The system shows them their current account data. The player tells the system what changes they want to make. The system validates the changes proposed by the player and stores them.

## Play Match

The goal of this use case is to allow a player to join or create a match and play chess. The player tells the system they want to play a match. The system asks for opponent selection and match setting information. The player provides the needed information. The system joins or creates a match, pairs up the player and opponent, and starts the match.

## Manage Friendships

The goal of this use case is to allow a logged-in player to view and make changes to their friendships. The player tells the system that they want to manage friendships. The system show the player their current friendships and asks the player what they want to do. The player tells the system what they want to do[[1]](#footnote-2). The system will attempt to comply to the request.

## Change Preferences

The goal of this use case is to allow a player to view and make changes to their preferences. The player tells the system they want to change preferences. The system presents the player with current preference information. The player tells the system what changes they wish to make. The system validates the changes and applies them.

(JAR Chess Use Case Diagram on next page)

# Use Case Diagrams

A close up of text on a white background

Description automatically generated

1. Choices include request friendship, respond friendship, remove friendship [↑](#footnote-ref-2)