Use Case Diagram for the Chess Game

Learn how to define use cases and create the corresponding use case diagram for the chess game.

We'll cover the following

- System
- Actors
  - Primary actors
  - Secondary actors
- Use Cases
  - Player
  - Admin
- Relationships
  - Include
  - Extend
  - Generalization
  - Associations
- Use case diagram

Let's build the use case diagram of the chess game and understand the relationship between its different components.

First, let's define the different elements of our chess game, followed by the complete use case diagram of the system.

System

Our system is a "Chess game."

Actors

Here are the main actors of our chess game.

Primary actors

- **Player:** This is the primary user and is responsible for playing the chess game. It can start a new game, make move or resign/forfeit a game.

Secondary actors

- **Admin:** This can add, remove, or update a player's account and membership, view open games, and validate player moves.

Use Cases

In this section, we'll define the use cases for the chess game. We have listed down the use cases according to their respective interactions with a particular actor.

> **Note:** You will see some use cases occurring multiple times because they are shared among different actors in the system.

Player

- **Create/Update account:** To create a new account to play a chess game or to update account information of an existing account
- **Login/Logout:** To log in to or log out from an account
- **View open games:** View available games that are waiting for players to join
- **Join a game:** Join an open game
- **Create a new game:** To create a new game to start playing
- **Make move:** To make a move in the game
- **Resign or forfeit a game:** To resign or forfeit a game so that it ends

Admin

- **Block/unblock member:** To block or unblock a member from playing the chess game
- **Cancel/Update membership:** To cancel the membership or to update the membership of an account
- **Add/modify member:** To add a new member or update member information

- **Login/Logout:** To log in to or log out from an account
- **View open games:** View available games that are waiting for players to join
- **Validate Moves:** To validate player move
- **Declare results:** To declare the result of the game when the game is over

Relationships

This section describes the relationships between and among actors and their use cases.

Include

- The "Make move" has an include relationship with the "Validate move" because the admin has to validate if the move was as per the rules set.
- The "Resign or forfeit a game" use case has an include relationship with the "Declare result" use case because the game will be over, and results will be declared when a player either resigns or forfeit from the game.

Extend

- The "Block/unblock member" has an extend relationship with the "Cancel/Update membership" since when the admin unblocks a member, there is a chance that their membership will be updated. When the admin blocks a member, its membership might be canceled.
- The "Validate move" use case has an extend relationshop with the "Declare result" use case as there is a chance that the game will be over and results will be declared when a player makes a validate move and checkmate another player.

Generalization

The "Make move" has a generalization relationship with the "Play pawn", "Play bishop", "Play king", "Play queen", "Play knight", and "Play rook", since a player can make any of these six moves.

Associations

The table below shows the association relationship between actors and their use cases.

Player	Admin
Create/Update account	Block/unblock member
Join a game	Cancel/Update membership
Create a new game	Add/modify member
Make move	Login/Logout
Resign or forfeit a game	Validate Moves
View open games	View open games
Login/Logout	Declare results

Use case diagram

Here is the use case diagram of the chess game: