# 1 System Interactions

## 1.1 Register

When the user opens the application they will have the option to create a new account. In this case they will supply a username and password to the client GUI, which will send a createAccount request to the client backed. The client backend will relay this information to the server which will check if the credentials supplied already exist. If the username and password are acceptable the server will create the new account in the user data database and the user will be able to login.

## 1.2 Login

When the user opens the application they will be able to provide their username and password to login to their account. The client GUI will send this information to the client backend which will send a validate request to the server. The server will validate the user's password against the account details in the user database. If they passwords match the server will send a success signal to the client, and they main screen will be displayed to the user. If authentication fails, the user will receive a failure message and be able to attempt login again.

## 1.3 Game Creation

The user has the option to either join an existing game or create a new one. If the user chooses to create a new game then the client backend will instruct the server to create a new game and log the game information in the game database. A new game will be displayed to the user.

## 1.4 Game Play

When the user clicks on a piece, the client GUI will communicate to the client backend with a getMoves request. The client backend will access its game data, calculate the list of possible moves, and send a response to the client GUI. The GUI will highlight possible destination squares. When the user clicks a destination square, the move will be validated on the client backend. If the move is legal it will be sent to the server and added to the game database. When the server receives the move it will update the other user's client and send confirmation to both parties.

# 2 Benefits, Assumptions, and Risks

## 2.1 Benefits

* Loose coupling. The design calls for a dumb GUI that has no knowledge of the game being played. The benefit of this design choice is that the client and server sides will have little reliance on each other.
* High cohesion. The Game class will encompass two Players, a Board (8x8 matrix of ChessPieces), and individual Pieces. This improves the robustness, reusability, and reliability of the application.

## 2.2 Assumptions

* Our users are generally aware of the rules of chess and how to play chess. These will not be made explicit in the application.

## 2.3 Risks

* The feature to highlight possible destination squares might slow down the game and take away from the user experience.