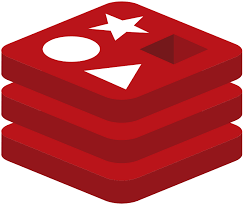
**Chess Project – High Level Design**



Applicative Communication Protocol



Frontend Client & Graphical User Interface

Backend Game Logic and Move Validator

**System Overview:**

A 2-player single-computer chess game!

The system consists of a graphical user interface in which a user can view the game board and try to move chess pieces by dragging the pieces visually on the presented board according to the rules of chess. The user client sends the move details in a certain textual protocol in a data transfer consisting of a representation of the source and the destination of the move. That data, through an applicative protocol, reaches the backend process, the goal of which is to analyze the move, update the game state and the player to play on its side if the move was valid and to return one of the following responses based on the given move and the state of the game ( in a textual / numerical shared format):

VALID\_MOVE – the move was valid, yet did not match any of the special valid moves mentioned below.

CAUSES\_CHECK – The move was valid and caused check on the rival opponent.

ORIGIN\_NOT\_ONWED – The move was not valid, the origin of the move is not owned by the player which played the move (includes empty source).

DEST\_OWNED – The move was not valid, the destination is owned by the player to move.

IMPLIES\_SELF\_CHECK – The move makes the king of the player which played the move vulnerable, unaccepted!

OUT\_OF\_BOUNDS – one or more of the given positions that represent the move is out of the borders of the board

DEFIES\_SOLIDER\_MOVE\_PATTERN – the move is not possible to execute based on the moved solider’s allowed move pattern and the game state

CHECKMATE – the move checkmated the opponent (bonus)

According to the backend validator response, a textual message that described the result of the move based on the response is presented to the user via the Graphical User Interface and the client’s game state is updated accordingly.

Extra Info on the communication:

At the start of the communication, the backend sends the client process the initial board state. When the user exists the game, a message is sent to the backend accordingly. Again, it is important to mention that the backend is NOT a server, the two sides are different processes that communicate via and applicative protocol, but we shall not get into the specific technological implementation (in this document), there are many different ones that are not important at this point.