Product Specification

Group 5

1 Description

The chess application will allow the user to play chess against another person locally, or against an AI which will have
three different difficulties; easy, medium and hard. The application will also be able the track the results of the games and
save the skill rating (using the Elo system) of the users and place them on a scoreboard. The AIs will be at a static rating
for each difficulty level.

2 System Requirements

2.1 Functional:

- Users shall be able to play against another human opponent (multiplayer)
- The system shall appoint rankings to the players of the game, which indicates that the winner of a game is awarded points which will have to be calculated
- The system keeps track of the results for each game played
  - This enables the system to provide an overview of the ranking of the human participants, based on on how many games they have won.
- Users shall be able to play against an AI, which will have 3 levels of difficulty
  - Novice
  - Intermediate
  - Expert
- Winning against a more intelligent or higher ranked machine player should award more points.

2.2 Non-functional:

- The system shall be made with 2D graphics
  - It must feature a board layout
  - It must feature player statistics
- The system shall be implemented on top of a open source graphics engine
  - Implementers may choose from JavaFX, LibGDX, Unity etc.
- The system shall be programmed in such a way that the ruleset (board size etc.) can easily be changed.
- System shall be programmed in Java

3 User stories

USER STORY 1: As a Chess player i want to play against good chess AIs so i can become better at chess.

USER STORY 2: As a pair of friends, we want to play chess together so we can spend more time with each other.

USER STORY 3: As a user of the chess game I want my available moves to be visualized when i pick a given chess-piece.

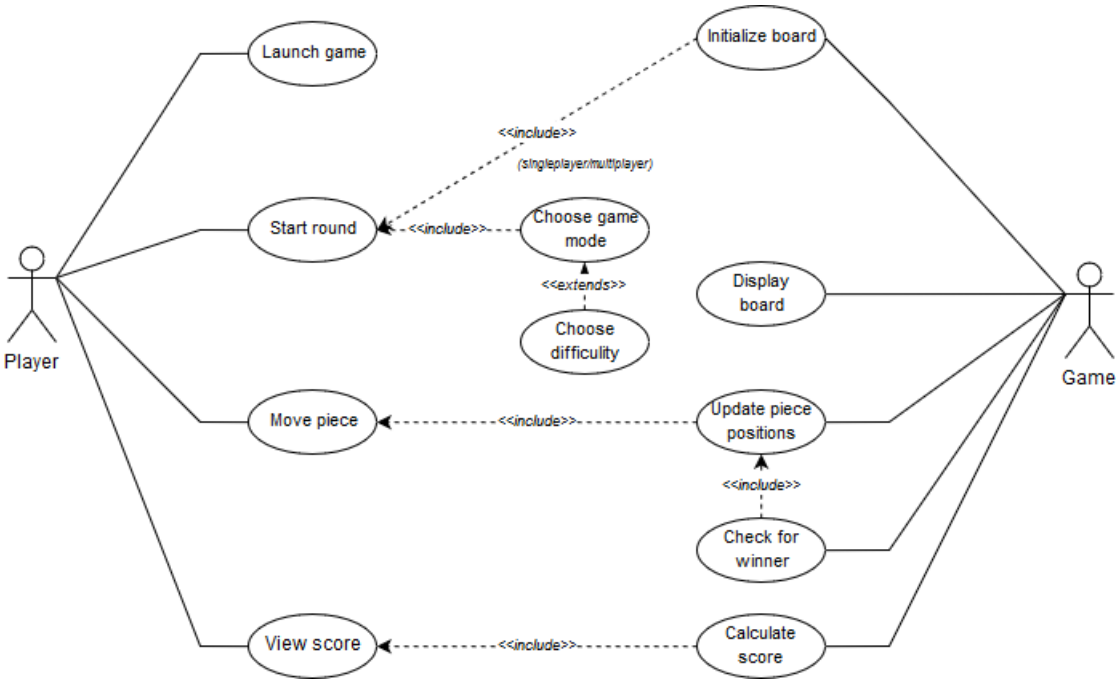
USER STORY 4: As a beginner level chess player I want to learn to play the game/basic rules of the game so I can play the game without help.

USER STORY 5: As a user I want the moves I make to be viewable through the course of a game.

USER STORY 6: As a frequent user of the software i want my MMR to be saved/updated after every match so i can show it to my friends.

4 Use case diagram

Diagram showcasing the major user operations of the chess application.



5 Use cases

5.1 Use case 1 (fully dressed)

A start menu allows a user to start the game and choose single player to play against a machine AI. The user will be prompted to write their player name so the high score can be saved in an external file. In the next step the player has to choose between three difficulties. After the AI difficulty is chosen, the game will start. The player must progress by moving their pieces according to the game rules. The player can move pieces by clicking on their piece (if it is their turn and a legal move) and pick the square that the piece should be moved to. The machine will respond according to the difficulty chosen by the player earlier.

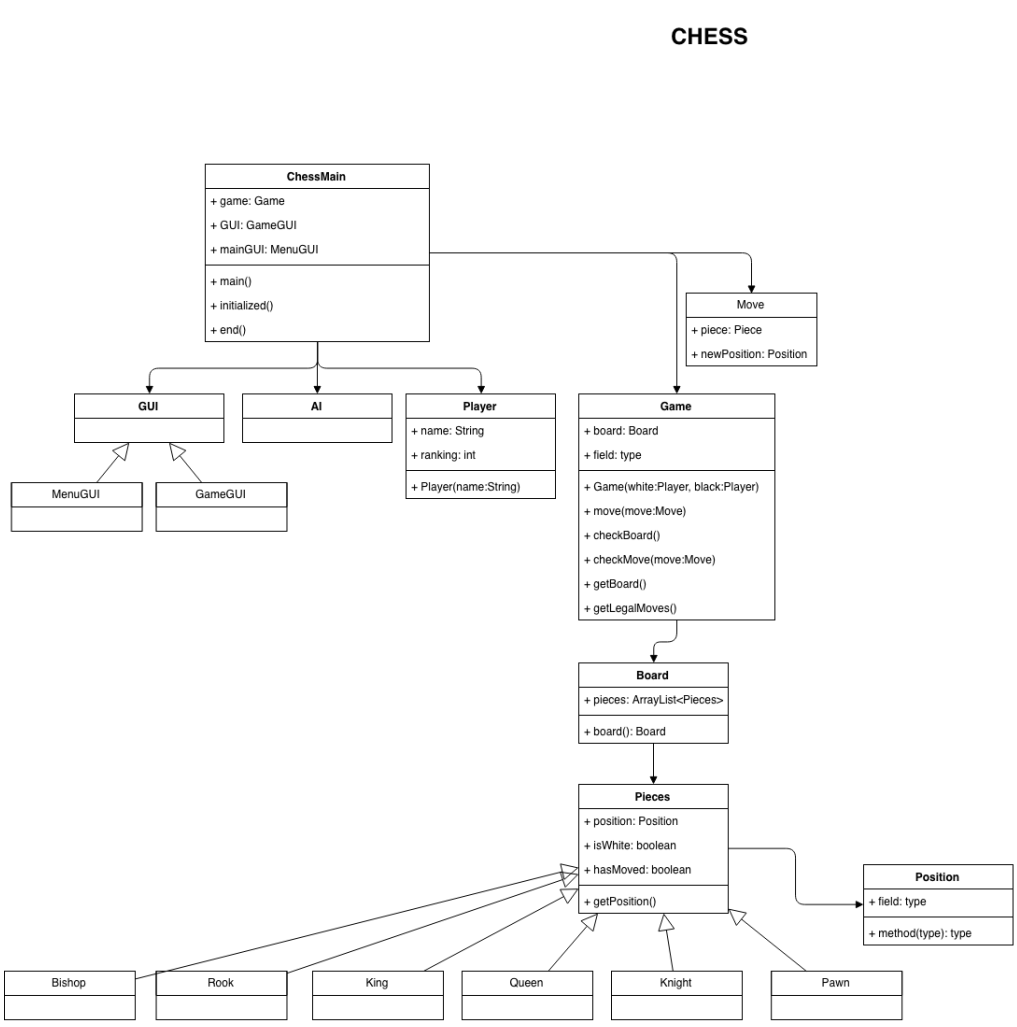
Name	Play chess with AI
Scope	Chess
Primary actor:	Player 1
Level	Summary
Stakeholders and interests	Player: wants to win and proceed in game machine: wants to win
Precondition	The player knows chess
Minimal guarantees	Player gets to play chess
Success guarantees	Either Player or machine wins
Trigger	Starts game in game menu
Main success scenario	1) Game starts 2) Player will take their turn to move pieces 3) After a couple of rounds the player will put the opponent in check 4) Checkmate, which results in a win for the player
Extensions	1) Game starts 2) Player will take their turn to move pieces 3) After a couple of rounds the machine will put the player in check 4) Checkmate, which results in a win for the machine

5.2 Use case 2 (fully dressed)

A start menu allows a user to start a game and choose multiplayer with another local player. After the player picks the multiplayer option it will be prompted to write name of player 1, then name of player 2, so the high score can be saved in an external file. After the name dialog, the players will be taken to the chess game and the possibility to play against each other. The game will be started and either player 1 or 2 starts. The players must progress with moving their pieces according to the game rules. The player can move pieces by clicking on their piece (if it is their turn and a legal move) and pick the square that the piece should be moved to.

Name	Play multiplayer chess
Scope	Chess
Primary actor:	Player 1
Level	Summary
Stakeholders and interests	Player 1: wants to win Player 2: wants to win
Precondition	Both players know chess
Minimal guarantees	Player 1 and player 2 gets to play chess
Success guarantees	Either player 1 or player 2 wins a game of chess
Trigger	Starts game in game menu
Main success scenario	1) Game starts 2) Player will take their turn to move pieces 3) After a couple of rounds player 1 will put the opponent in check 4) Checkmate, which results in a win for player 1
Extensions	1) Game starts 2) Player will take their turn to move pieces 3) After a couple of rounds player 2 will put the opponent in check 4) Checkmate, which results in a win for player 2

6 Domain model (class diagram)



6.1 Short explanation of the class diagram

ChessMain:
The ChessMain class is the class where we begin the game (main method).

- main() method will call the initialize(). Once game is initialized we will go into an endless loop where we repeatedly asks the GUI/AI for the next move. If the GUI send a resign signal instead of a move we exit the loop and call the end() method. Otherwise we send the move to the game with the game.move() method. This method will return a Boolean value, true means everything is ok, and we can go on to the next round. False means

that an illegal move was made, and we make the GUI tell the player this and we let him/her try again until the move is accepted. After a move is made we call the game.checkBoard() method. This will check the board after the move is made to see if someone has won, there is a check, there is a checkmate or a there is a promotion. Here we are only interested in whether someone has won, the game will take care of the rest. If someone has won we call the end() method.

- Initialize() method will make a MenuGUI object where the player can provide information about the game he/she is about to play. This information is used to initialize a Game and two player objects or one player and one AI object. Then we can make a GameGUI object with the game as a parameter.

- end() method will make the GUI display an end screen and calculate the new rankings.

Game:
The Game class is where we have our current game. It has a Board variable that is simply a list of all the pieces, the position of the pieces is stored in the pieces themselves. The ChessMain class will interact with the game via a few methods.

- Move(Piece piece, Position newPos) will simply make a new Board with updated pieces. This is mostly just a copy of the previous board except the piece that were moved and pieces that are knocked out. However before we make this new board we have to make sure its a legal move by calling the checkMove(Piece piece, Position newPos) method. If its not a legal move it returns false and does nothing, otherwise true.

- checkBoard() will check the current board for certain events such as: someone has won, there is a check, there is a checkmate or a there is a promotion. This method will then make a new board if needed. It will return true if someone has won, false otherwise.

- getBoard() the GUI will need the board to display it.

- getLegalMoves(Piece piece) mainly used by the checkMove() method but can also be used by the GUI to display legal moves.