Product Specification
Chess Game Development
INF112-V18 - Team NASA

Jonas Triki (Project Manager)
Jonas Mossin Wagle (Tech Lead)
Marianne Luengo Fuglestad (QA)
Eirin Sognnes
Elise Fiskeseth
Paraneetharan Sabaratnam
Stian Fagerli
Sofia Hestenes Eika

Revision History

Date	Description	Revision
19.02.2018	Initial version	1.0

Contents

1. Introduction

1.1 Purpose

This document specifies the software requirements and relevant informations related to the chess game application developed by TeamNASA. The document is intended to be read and reviewed by TeamNASA and interested parties.

1.2 Scope

The objective of the project is to develop a user friendly, robust and reliable multiplayer chess application, where a user can play against a machine player or another human player. The system offers three levels of intelligence such as novice, intermediate and expert level. The users can choose their preferred level of intelligence according to their chess skill.

Key features of the game:

- The system offers a user to choose either to play against another human player or a machine player.

- The system offers three levels of intelligence to the user in the event the user choose to play against the machine player.

- The system displays a 2D chessboard with visually appealing pieces and board layouts that follow standard rules of the chess game.

- The system enables the valid moves for each chess pieces in the chess board.

- The system displays necessary notifications during the chess game.

- The system record the results of each game played and provides ranking of each player based on number of games each players has won.

1.3 Overview

The rest of the document elaborates the system requirements for the chess game.

2. System Requirements

2.1 Functional Requirements

- The system offers the option to choose between a human player and a machine player

- The system provides a name registration function, which enables unique identification of each user.

- The system upholds the uniqueness of user names.

- The system offers the option to choose the level of intelligence.

- The system offers the option to choose the color of the pieces (White /Black).

- Displaying the chess board, nearest right hand side square is white.

- Displaying all 32 pieces in the correct positions.

- The system only allows a white piece to make the very first move in a game.

- Enable Pawns valid moves in the system.

- Enable Knights valid moves in the system.

- Enable Bishops valid moves in the system.

- Enable Rooks valid moves in the system.

- Enable Queens valid moves in the system.

- Enable Kings valid moves in the system.

- Enable the functionality of castling in the game.

- Enable the functionality of En-passant move in the game.

- In an event of invalid move the system displays notification.

- In the event of check, checkmate and stalemate the system displays relevant notifications.

- The system offers the last move undo option.

- The system offers the option to quit the game anytime.

- The system records all the moves from both players.

- Enable the time keeping functionality for each move.

- Displaying the captured pieces on both sides.

- Enable the points calculation for the entire chess game. Such as Win, draw, lose, each captured pieces and the moves.

- The system keeps the win and loss results records of the games played.

- The system offers the ranking of players according to the level of intelligence.

2.2 Non Functional Requirements

- Reliability.

- Performance.

- Response time.

- Throughput.

- Maintainability.

- Scalability

2.3 User Stories

- As a user I want a start menu so that I can choose to play against another human player or a machine player.

- As a user I want a menu so that I can choose between at least three intelligence levels of the machine player.

- As a user I want the program to have a 2D chessboard with visually appealing pieces and board layouts that follow standard rules of chess so that I can play a simulated game of chess.

- As a user I want the program to keep track of the results of each game so that the user can see a ranking of human players based on how many matches they have won.

- As a user I want the program to stop me from making illegal moves and to carry out legitimate features of standard chess rules.

- As a user I want the program to contain a timer to show how long a game has lasted.

- As a user I want the program to end a game of chess when a player is in checkmate.

- As a user I want the program to have a menu selection where I can get an overview of the game rules.
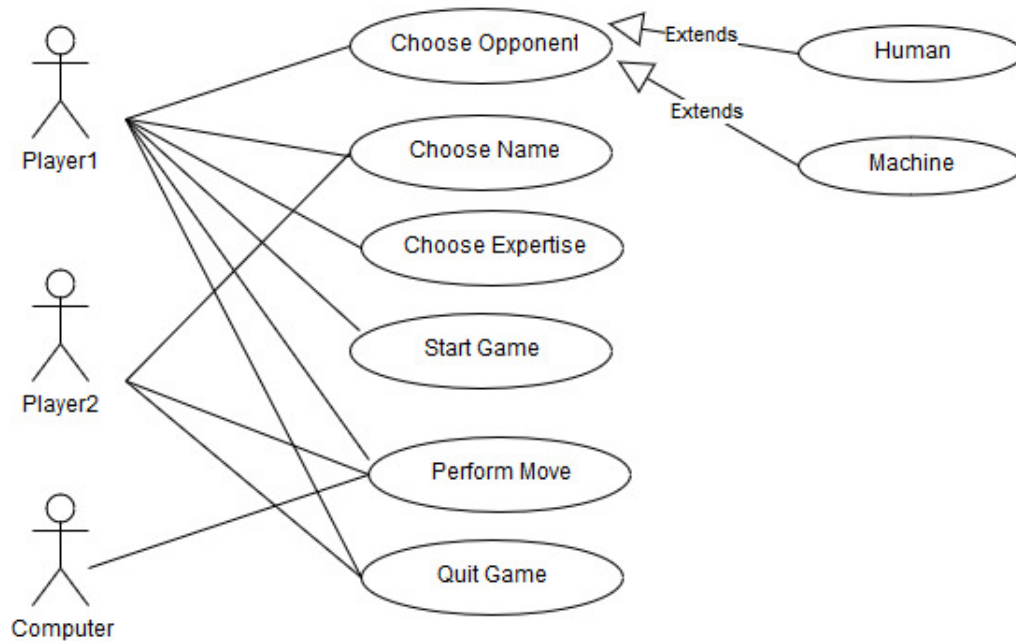
2.4 Use case diagrams

Figure 1: Use case diagram

2.5 Fully dressed use cases

2.5.1 Use case for Multiplayer

Use Case Name: Multiplayer chess game
Scope: Chess Game
Primary actor: Player
Stakeholders and interests:

- Player: Wants to play a game with an opponent. Only legal moves. Score should be saved. Time is kept track of.

- Opponent: Wants to play a game against player. Only legal moves. Score should be saved. Time is kept track of.

- High score holders: All scores are safely kept track of.

Precondition: Player has access to game and is identified if in high score list.
Postcondition: Player successfully played a game, score is recorded in system.

Main Success Scenario (or Basic Flow)

1. Player wants to play a multi player chess game

2. Player opens the program

3. Player gets a menu displayed

4. Player chooses to play multi player game

5. Screen with choosing of names and white/black for player and opponent

6. System keeps track of time passed for Player and Opponent

7. White makes a move

8. Black makes a move
   (Repeats step 7 and 8, until game is done.)

9. System saves the scores of Player and Opponent

10. System displays end menu, with options of new game, ranking and end

Extensions:
     6 a. Time runs out for Player/Opponent
     6 a1. The Player with time left wins. Game ends and scores are saved. Go
to step 9
     7 a. Player makes illegal move
     7 a1. Program shows a message, and lets Player try again
     7 b. Player quits
     7 b1. System goes to step 9

Technology and Data Variations List: Different computers (MAC, PC)
Frequency of occurrence: Very often
Special Requirements: Should run smoothly

2.5.2 Use case for Single Player

Use Case Name: Single player chess game

Scope: Chess Game

Primary actor: Player

Stakeholders and interests:

- Player: Wants to play a game with an opponent. Only legal moves.
  Score should be saved. Time is kept track of.

- High score holders: All scores are safely kept track of.

Precondition: Player has access to game and is identified if in high score list.

Postcondition: Player successfully played a game, score is recorded in system.

Main Success Scenario (or Basic Flow)

1. Player wants to play a single player chess game

2. Player opens the program

3. Player gets a menu displayed

4. Chooses to play a single player game

5. Screen with choosing of name, color and level of intelligence.

6. System keeps track of time passed for player

7. White makes a move

8. Black makes a move
   (Repeats step 7 and 8, until game is done.)

9. System saves the score of Player to the system

10. System displays end menu, with options of new game, ranking and end.

Extensions:
6 a. Time runs out for Player
6 a1. Player loses game. Go to step 9
7 a. Player makes illegal move
7 a1. Program shows a message, and lets Player try again.
7 b. Player quits
7 b1. Player presses quit
7 b2. System goes to step 9

Technology and Data Variations List: Different computers (MAC, PC)
Frequency of occurrence: Very often
Special Requirements:
    - Three level of intelligence, novice, intermediate and expert
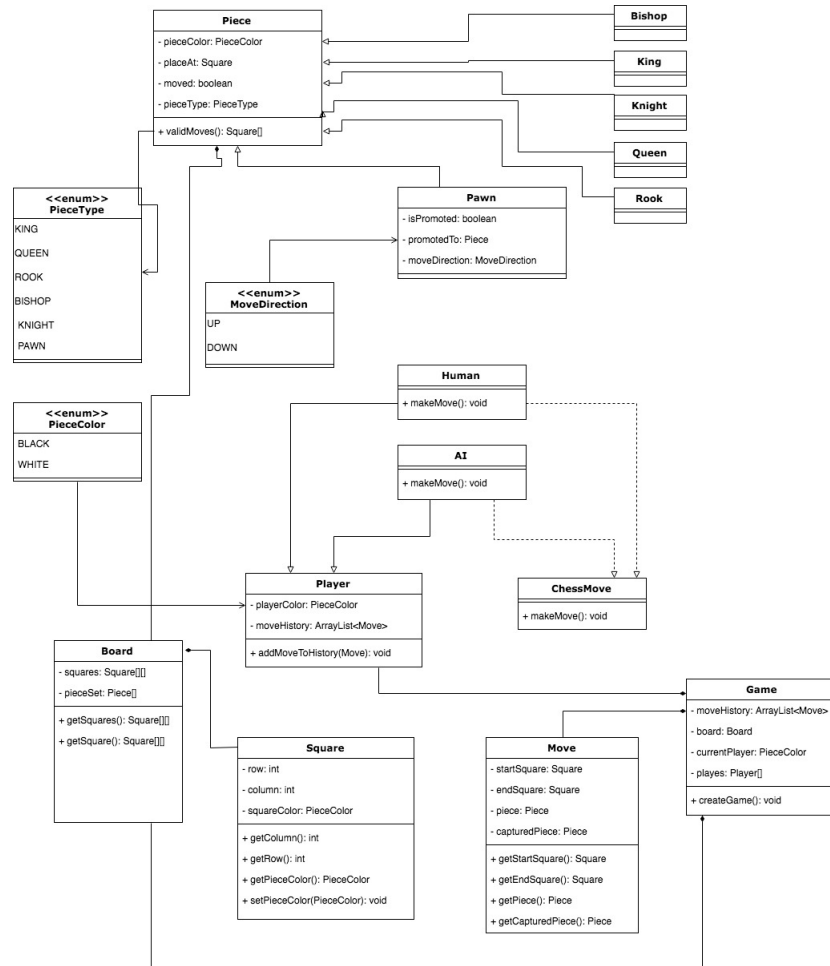   - Should run smoothly

3. Domain Model

3.1 Class Diagram

Figure 2: Class Diagram.