Requirements and Analysis Document for Mindchess

Elias Carlson, Elias Hallberg, Arvid Holmqvist, Erik Wessman

October 23rd, 2020
1.0

1 Introduction

Mindchess is a desktop client of the internationally renowned and ancient game of chess. The purpose behind the application was to create a standalone version of chess built with object-oriented design. Its functionality revolves around playing locally against another player or against the computer. The rules of chess will be applied automatically and users will only have to worry about strategy, rather than debating over rules. The users the application targets are people who want a local version of chess where they can play against friends or family or practice their chess skills.

1.1 Definitions, acronyms, and abbreviations

GUI - Graphical User Interface: the visual interface of an application, with which users interact.

Object-Oriented Design: rules and patterns used in object-oriented languages such as java to create solid applications.

User Stories: a tool that's commonly used and plays a large role in agile workflows/SCRUM to divide and distribute work as well track progress.

Domain Model: a domain model is an abstract overview of a domain's objects and their relations with each other. Commonly used to communicate and plan between different divisions of a project.

MVC - Model-View-Controller: MVC is a way to divide an application in three components. Its main intention is to separate the application data and logic, the model from the presentation and interaction of the application to the user.

Computer player (AI): A virtual player that can be chosen as your opponent and makes moves as if you were playing against another player online.

2 Requirements

For Mindchess to be considered finished it will need to meet multiple requirements. The User stories used in development, that describes the desired functionality, each have to reach a certain acceptance level to be considered finished. The application will only be considered complete when all User stories are implemented.

2.1 User Stories

Start Program [Implemented]

ID: #001

Description
As a User, I want to be able to start the program for the game so that I can begin playing.

Confirmation
Functional:
- Can I open a window for the application?

See Board [Implemented]

ID: #002
Description
As a User, I need to be able to see the view of the chess board so that I can plan my next move.

Confirmation
Functional:
- Can I see the chessboard?

See/use Main Menu [Implemented]

ID: #003
Description
As a User, I want to be able to see/use a Menu to have control over the application.

Confirmation
Functional:
- Can I use the Menu to start a new game of Chess?
- Can I use the Menu to exit the program?

Move chess pieces [Implemented]

ID: #004
Description
As a User, I need to be able to move the chess pieces so that I can play the game.

Confirmation
Functional:
- Can I select a piece?
- Can the chess pieces be moved according to my input?

Start game [Implemented]

ID: #005
Description

As a User, I want to be able to start a new game of chess, so that I can play.

Confirmation
Functional:
- Can I change the duration of the game?
- Can I enter the names of the players who will be playing?
- Can I start a game with all the chess pieces?
- Is a standard game of chess prepared when I switch to the game screen?

See Timer [Implemented]

ID: #006
Description
As a User, I want to be able to see a Timer for each player in my games so that I know how much time I have left.

Confirmation
Functional:
- Can I see timers for both players during the game?
- Does the Timer count down only when it's my turn?

Alternating turns [Implemented]

ID: #007
Description
- As a user, I want me and my opponent to have alternating turns so that we can play the game.

Confirmation
Functional::
- Can I only move my pieces?
- Can I make my move after the opponent?
- Is it impossible for one player to make two moves in a row?
- Is it my opponent's turn after I have made my move?

Taking pieces [Implemented]

ID: #008
Description
As a User, I want to be able to take my opponents pieces, so that I can win.

Confirmation
Functional::
- Does the pieces that are taken disappear from the board?

Change Duration [implemented]

ID: #009

Description

As a User, I want to be able to change the Duration of my games.

Confirmation

Functional

- Can I choose the duration for the game I'm starting from a set of different durations?

Draw/forfeit Game [implemented]

ID: #010

Description

As a , I want to be able to propose a draw or forfeit a game, so that I can avoid playing for infinity.

Confirmation

Functional:

- Can I offer a draw to my opponent?
- Can my opponent accept or decline my offer?
- Can I choose to forfeit the game?

End the game [implemented]

ID: #011

Description

- As a user do I want to be able to win the game so that there is a reason to play.

Confirmation

Functional:

- Can you surrender?
- Does the game end when someone wins or draws?
- Does the game end when the timer runs out for either player?

Show possible movement options [implemented]

ID: #012

Description

As a player, I want to see all possible moves I can make, so that I know where I can move my pieces

Confirmation

Functional:

- Can I see where on the board a piece can move when I click on it?

Apply Basic Rules [implemented]

ID: #013
Description
As a User, I need the pieces to follow the rules of chess so that I can play the game.

Confirmation
Functional:
- Do the chess pieces move according to the rules of chess?

Special Rules [implemented]

ID: #014
Description
As a User, I want to make special rules that exist in chess so that I use all my talents.

Confirmation
Functional:
- Can I perform castling when it is permitted?
- Is En Passant possible when it’s available?
- Can I promote a pawn to a knight, bishop, rook or queen when it reaches the end of board?
- Am I prohibited from moving my king into check?

Show dead pieces [implemented]

ID: #015
Description
As a User, I want to see the pieces that have been taken during the course of the game so that I can see who’s in lead.

Confirmation
Functional:
- Can I see the pieces I and my opponent have taken next to the board?

Analyze game [implemented]

ID: #016
Description
As a Player, I want to be able to see all plies that were made during the game, after it has finished, so that I can analyze my performance.

Confirmation

Functional:
- Can I choose to analyze the game when it has finished?
- Can I see a list of all the moves/plies that were made during the game?
- Can I click on each ply and see what pieces moved in it, and a snapshot of the board at that stage?

AI [implemented]

ID: #017
Description
As a User, I want to be able to play against a computer player (or AI) so that I can practice my skills.

Confirmation
Functional:
- The AI can make moves when it is it’s turn
- The AI can take players

Load old games [implemented]

ID: #018
Description
As a Player, I want to be able to see a list of old and/or inactive games from the current session and load them to either continue or analyze them.

Confirmation
Functional:
- I can see a list of old/inactive games from the current session
- I can click on any of the games and load them to either continue or analyze them.

Intelligent AI [implemented]

ID: #019
Description
As a User, I want the AI to make calculated moves so that I get a challenge

Confirmation
Functional:
- The AI can evaluate moves according to certain parameters and choose the best move
- I can choose between different difficulties of AI when starting a game

Advanced win conditions [not implemented]

ID: #020
Description
As a User I want the game to recognize when advanced win conditions are met so I don’t have actually kill the king

Confirmation
Functional:
- Does the game recognize a mate/checkmate?
- Does the game recognize a stalemate?

2.2 Definition of Done

For a user story to be considered done, the following criteria must be met:
- The features listed in the user story must have been tested and confirmed to be working with the current master/dev branch
- The code must be shown, explained to, and reviewed by the other working group
- The code must have been merged with the master/dev branch without conflict

2.3 User Interface

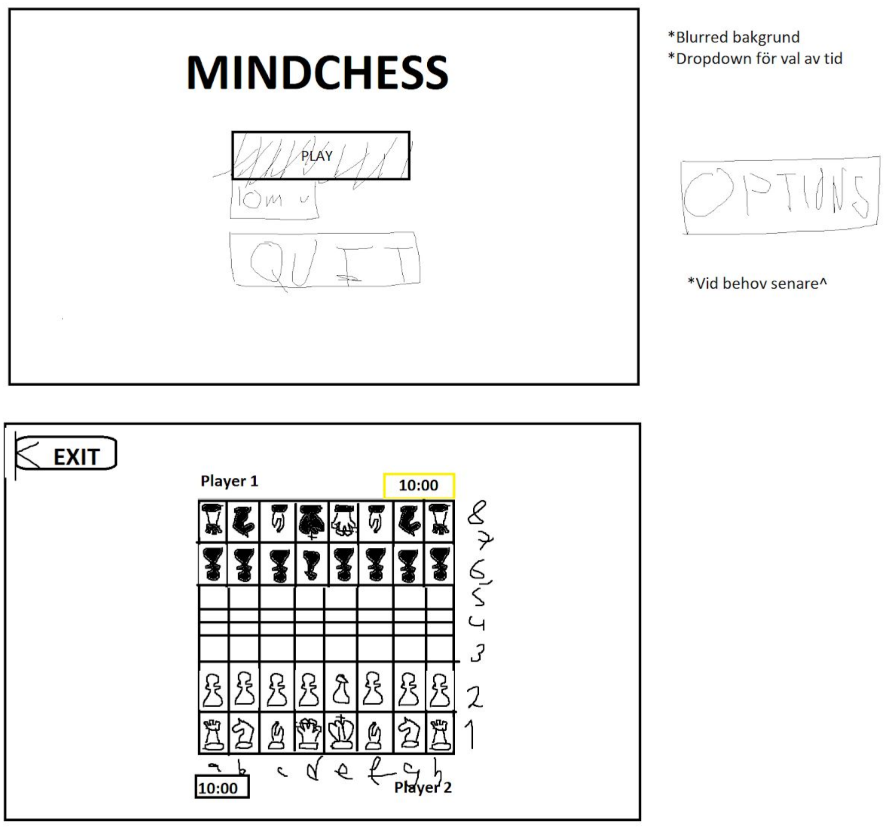

Figure 1. First sketch/prototype of the application GUI.

The application's GUI is very simple and consists of two views; a menu view and a game view. Both views have different purposes where the menu view allows users to start a game, and select the options for said game, or exit the application. The game view on the other hand is where the majority of the application's features are available and where users will spend the majority of their time using the program.

In the center of the game view the game's chess board can be found, complete with chess pieces. Above and below the board the names of both players as well as the timers indicating the amount of time they have left can be found. Along with this there are letters and numbers on the bottom and right side of the board, used for indicating the name of each square on the board. In the top left corner the button that takes the user back to the menu view can be found.

3 Domain Model

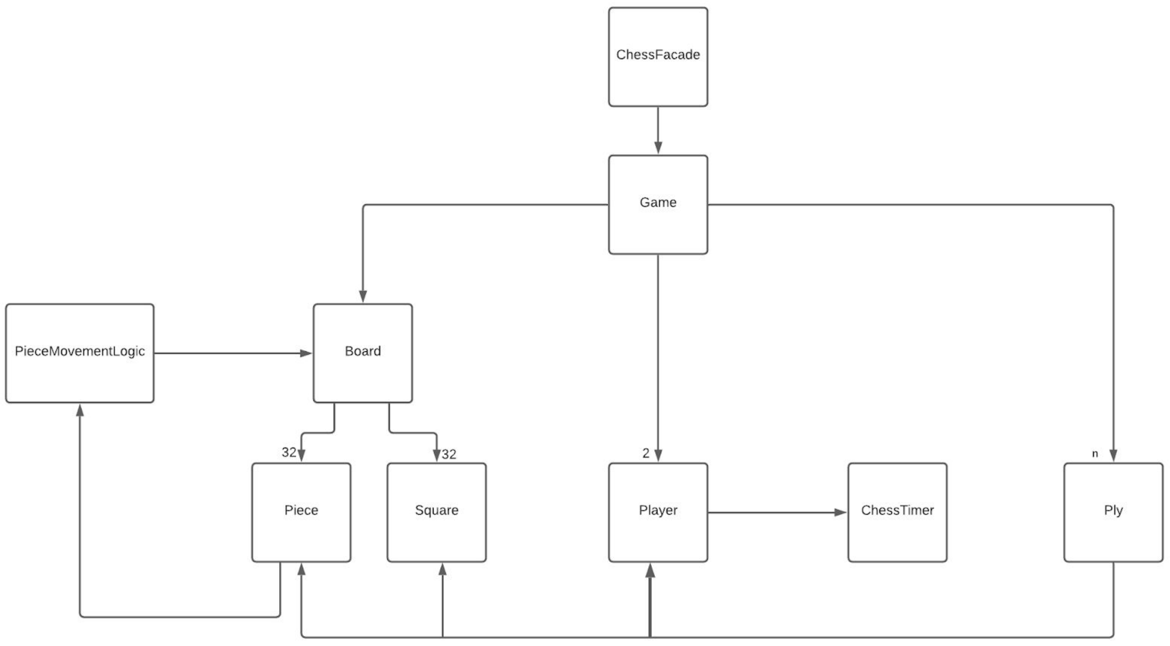

Figure 2. Picture of domain model. [1].

3.1 Class Responsibilities

ChessFacade

The *ChessFacade* class works as an entry point for the model. It hides unnecessary information from the user and makes communication with the model simpler and safer. *ChessFacade* does not contain any logic in itself other than keeping track of ongoing games, instead it calls other classes in the model to do the logic.

Board

Just like in real chess, our *Board* class contains the pieces on the board and their respective positions. When a game is started it creates the pieces and gives them their initial positions, and if a piece is taken it stores it as taken.

Game

The *Game* class gets created whenever someone creates a game and handles the logic during that game. If a player chooses a piece to move, it finds that piece's legal moves and displays them. If a player makes a move, it changes its position and takes eventual opponent pieces on the square it moves to. In addition to this it also switches between the player's turns and ends the game when it is finished.

Ply

The *Ply* class is a representation of a move that has been made. A list of plies are stored in a list in our *Game* class so that the information of every move that has been executed is saved. Certain rules of chess depend on what moves have already been executed and therefore our *Ply* class contains information about which player made which move.

PieceMovementLogic

The *PieceMovementLogic* class holds all logic on how every piece is allowed to move on the chess board. It is used by the *Board* class when it wants to move a piece which will return a list of possible moves the piece can make.

Piece

The *Piece* class is like the name suggests the class that represents each piece in the game. It holds information about the piece type, color, value and rules that decide its movement.

Player

The *Player* class represents the players in a given game of chess. It's created by the *Game* class and only 2 players should exist for each *Game*. It contains the player's timer and its name. It's used for controlling the timer and to make sure players can only move their own pieces.

ChessTimer

The *ChessTimer* class keeps track of the time left for each player. Only two instances should exist at the same time as there can only be two players playing at the same time. The *ChessTimer* counts down each second from a specified value down to 0. The countdown time can be set to a value in minutes and seconds. The timer is paused while the opposite player makes their move.

Square

The Square class represents a certain position on the gameboard. It keeps track of its position and type.

4 References

[1] LucidChart, “Online Diagram Software & Visual Solution”, 2020. [Online]. Available: https://www.lucidchart.com/pages/, accessed on: 2020-10-02