CS 451: Software Engineering Term Project Requirements Specification

Group Members

Phillip Ryan
Daniel Fitzick
Troy Santry
Karishma Changlani

Revision History

Name	Date	Reason for Change	Revision
Phillip Ryan Daniel Fitzick Troy Santry Karishma Changlani	07/06/16	Initial template	0.0
Phillip Ryan Daniel Fitzick Troy Santry Karishma Changlani	07/10/16	First Draft	1.0
Phillip Ryan Daniel Fitzick Troy Santry Karishma Changlani	07/28/16	Final Draft	2.o

Table of Contents

1. Introduction

1.1. Purpose of Document

This document will provide all of the requirements for the Checkers game. It will serve as a reference for the developers and the customers for developing the final version.

1.2. Scope of Document

The document will contain enough information such that a developer will be able to translate the requirements to code without ambiguity.

1.3. Overview of Document

This document will contain functional and nonfunctional requirements for the Checkers game, as well as use cases and diagrams. The game contains a component that will allow a player to connect to another person remotely. The client server interaction will be abstracted form the player. In order to connect to another player someone must initiate a game and another player must send a request to their known ip address. The requirements will be provided for a desktop application version.

2. Description

2.1. Product Perspective

2.1.1. This application is a simple checkers game designed such that two people can play it remotely from their individual computers when they are on the same network. When a player starts the application they have a choice to either start a new session or join an existing session.

2.1.2. If the player chooses to make their own new session they will enter an empty game and the session will wait for some time for another player to join. If the player chooses to join a game the application asks the player to enter the IP address for the computer that started the session to join.

2.1.3. Each player is assigned a color randomly and we continue with a standard checkers game until either a winning/draw condition is met, or a player decides to forfeit or if both players mutually decide to call it a draw.

2.1.4. The game requires that the two players share a network and the players have Java 8 runtime on their computers in order to run the application.

2.1.5. The game itself is a 2-D checkers board. Both players see the entire board and all piece positions. The game highlights valid moves and only allows those moves to be selected by the player.

2.1.6. The starting pieces are represented by the color red and white for the other set of pieces.

2.1.7. There is a scoreboard in order to let players know of the number of games won by each of the players.

2.2. Product Functions

The program will allow users to:

- Select a username
- Connect directly to another instance of the program remotely
- Begin a checkers game (this requires a request and an accept)
- Make valid moves
- Automatically detect when one player has won the game
- Output notation files

2.3. User Description

The ideal users are pairs of people who wish to play checkers. They must know each other prior to play and have some means to communicate IP addresses.

2.4. Assumptions and Dependencies

2.4.1. Java Runtime

2.4.1.1. JRE 8 is the latest version of Java runtime which lets you run Java applets on your machine. This game is entirely a Java application and hence if there is a crash while running JRE or if Java crashes for any other reasons the game will crash as well.

2.4.2. Network connections

2.4.2.1. The checkers game utilizes UDP protocols to populate a list of other users that are available to play a game of checkers

2.4.2.2. Once a user selects a particular game then a TCP connection is established where moves are sent from player to player.

2.4.3. Platform

2.4.3.1. Any operating system that can use JRE and has a basic graphical display and point and click support should be able to run this application.

2.4.3.2. However, bug checking is done mainly on standard Windows PC's and hence this is the most reliable platform for using the game.

2.5. Requirements Apportioning

Priority Level	Description
1	**Priority 1** requirements that are essential to product and must be completed for the final build.
2	**Priority 2** requirements that are committed for the delivery of the final build but these are not required for the overall functions of the product
3	**Priority 3** requirements that are not required for the final build and should be considered part of the design of the overall system. If there is time remaining then these requirements will be incorporated.

3. Functional Requirements

3.1. Moves

3.1.1. (Priority 1) There are three fundamentally different types of moves: the ordinary move of a man, the ordinary move of a king, the capturing moves.

3.1.2. Ordinary Move Of A Man

3.1.2.1. (Priority 1) An ordinary move of a man is its transfer diagonally forward left or right from one square to an immediately neighbouring vacant square.

3.1.2.2. (Priority 1) When a man reaches the farthest row forward (known as the “king-row” or “crown-head”) it becomes a king, and this completes the turn of play. The man can be crowned by either player by placing a man of the same colour on top of it before the next move is made. (It may be necessary to borrow from another set if no captured man is available for the purpose).

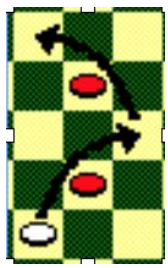
3.1.3. Ordinary Move Of A King

3.1.3.1. (Priority 1) An ordinary move of a king (crowned man) is from one square diagonally forward or backward, left or right, to an immediately neighbouring vacant square.

3.1.4. Capturing Moves

3.1.4.1. (Priority 1) A capturing move of a man is its transfer from one square over a diagonally adjacent and forward square occupied by an opponent's piece (man or king) and onto a vacant square immediately beyond it. (A capturing move is called a "jump"). On completion of the jump the captured piece is removed from the board.

3.1.4.2. (Priority 1) If a jump creates an immediate further capturing opportunity, then the capturing move of the piece (man or king) is continued until all the jumps are completed. The only exception is that if a man reaches the king-row by means of a capturing move it then becomes a king but may not make any further jumps until their opponent has moved. At the end of the capturing sequence, all captured pieces are removed from the board.



3.1.4.3. (Priority 1) All capturing moves are compulsory, whether offered actively or passively. If there are two or more ways to jump, a player may select any one that they wish, not necessarily that which gains the most pieces. Once started, a multiple jump must be carried through to completion. A man can only be jumped once during a multiple jumping sequence.

3.1.4.4. (Priority 1) A capturing move of a king is similar to that of a man, but may be in a forward or backward direction.

3.1.5. Illegal moves

3.1.5.1. (Priority 1) The game only gives legal moves as options to the player hence negating the possibility of illegal moves.

3.2. Game Results

3.2.1. There are only two possible states to define: the win and the draw

3.2.2. The Win

3.2.2.1. (Priority 1) The game is won by the player who makes the final move. An opponent can be prevented from making moves in several ways:

- When all of the opponent's pieces have been captured
- When all of the opponent's remaining moves have been blocked

3.2.2.2. (Priority 1) A player also wins in the case that

- An opponent resigns at any point.
- Forfeits the game.

3.2.3. The Draw

3.2.3.1. (Priority 1) The game is drawn if at any stage both players agree on such result. A game shall also be declared drawn where:

3.2.3.1.1. (Priority 1) At any stage a player has:

- Neither advanced an uncrowned man towards the kin-row during their own previous 40 moves.
- No pieces have been removed from the board during their own previous 40 moves.

3.3. Board and Piece Layout

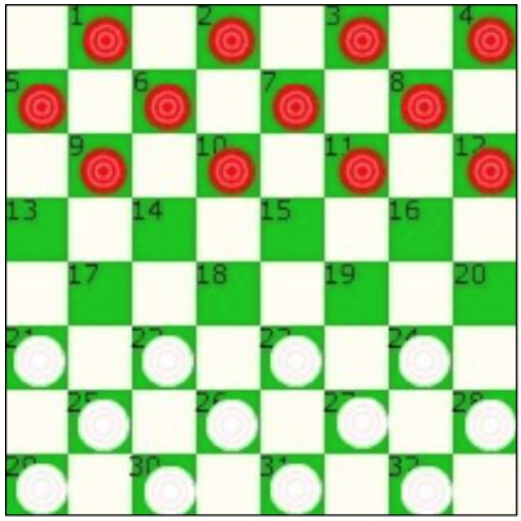
3.3.1. (Priority 1) The board is composed of 64 squares, alternately light and dark arranged in a square array of 8 rows and 8 columns and bounded by a neutral border.

3.3.2. (Priority 1) The board is placed between the players in such a way that a green square is to be found on the player’s near left side, and a white square to their right side.

3.3.3. (Priority 1) The 32 green squares used for play on the board shall be assigned numbers 1-32 for descriptive purposes. These numbers are the official reference system for notations and recording games.

3.3.4. (Priority 1) At the beginning of a game, one player has 12 dark-coloured discs (referred to as “pieces” or “men”), and the other player has 12 light-coloured discs.

3.3.5. (Priority 1) The Red pieces shall be set for beginning play on the first 12 squares of the player playing the Red pieces starting right to left, Nos. 1 thru 12. The White pieces will be on the last 12 squares, Nos. 21 thru 32, with No.32 being the nearest double corner square.



3.4. Order of Play

3.4.1. (Priority 1) To start the game through random selection, the color they will play is decided. In subsequent games the players alternate colors. The choice of a color is selected through a “toss of a coin”.

3.4.2. (Priority 1) The first move in each game is made by the player with the Red men. Thereafter the moves are made by each player in turn.

3.5. Peer to Peer Communication

Each instance of the program has the capability to act as either a client or a server.

3.5.1. Server

Acting as a server, the program must:

- Receive a request for a game
- Accept or deny this request based on user input
- Randomly decide which side is going first
- Receive, verify, and display move data from the client
- Verify and send move data based on user input
- Detect wins and ties
- Send win and tie data (sometimes based on user input)

3.5.2. Client

Acting as a client, the program must:

- Send a request for a game
- Receive the acceptance or denial of this request
- Receive data about which side is going first

- Receive, verify, and display move data from the server
- Verify and send move data based on user input
- Receive and verify win and tie data
- Send win and tie data based on user input

4. Non-Functional Requirements

4.1. Operating System Requirements

4.1.1. Must have Java Runtime Environment 8 or greater
4.1.2. Must have Internet connection

4.2. Accessibility

4.2.1. Players should have basic knowledge of networking and be able to find their local IP address

4.3. Playtesting

4.3.1. For every build candidate playtesting will occur. Playtesting is when a group of people (in this case, the development team) play several games using the program. With the following things.
4.3.1.1. Game connection
4.3.1.2. Moves offered.
4.3.1.3. Crowing of pieces
4.3.1.4. Capturing of pieces
4.3.1.5. Forfeit and Draw condition
4.3.1.6. Winning condition

5. User Interface

5.1. Game Play Layout

5.1.1. Centered in the middle of the window will be an 8x8 green and beige checkered board. The board is situated so that a colored green square is situated in the bottom left corner.
5.1.2. Draw, forfeit, and quit buttons will be under the game board in a row from left to right.
5.1.3. There will be a button in the top left corner to allow users to view the move notation for the game. The screen layout will change to the “Move Notation” layout.

5.1.4. During gameplay, if a draw is requested, the other player forfeits, or if either player wins then an alert box will appear in the middle of the screen displaying the correct information.

5.1.5. Player Interaction

5.1.5.1. Players move by selecting a piece. Valid spaces to move to are automatically highlighted. The player then selects one of the highlighted spaces to complete the move.

5.1.5.2. To draw, forfeit, or quit the game you would click the appropriate button at the bottom of the screen.

5.2. Move Notation Layout

5.2.1. Centered in the middle of the screen will be a scrollable text box with all the moves separated by a comma.

5.2.2. In the top left corner of the screen will be a “Back” button to go back to the “Game Play” layout.

5.3. Player Interaction

5.3.1. Click the ”Back” to return to the “Game Play” layout.

5.4. Start Screen Layout

5.4.1. In a Centered horizontally and vertically in the middle of the screen will have a “Start Game” button.

5.4.2. There will be a “Connect to Game” label, a textbox and a “Connect to Game” button in a horizontal row below the “Start Game” button.

5.4.3. Player Interaction

5.4.3.1. Player can start a new game by selecting the “Start Game” button. The screen will change to the “Game Play” layout.

5.4.3.2. Players can connect to another player’s game by entering the ip address of the player’s computer in the text box and then clicking the “Connect” button. The layout will change to the “Game Play” layout if the client could connect to the game. The screen will display an alert if no game could be found at that ip address.

6. Use Cases

6.1. Use Case Flow

6.1.1. Starting a game

6.1.1.1. Open application, choose start new game

Precondition: No game has been started

Action: Player starts a game

Postcondition: Player is waiting for another player to join their game

6.1.2. Connecting to a game

6.1.2.1. Open application, choose connect to a game

Precondition: A game has already been started

Action: Player uses ip address to connect to game

Postcondition: Player has connected and a game begins

6.1.3. Moving a man

6.1.3.1. Player chooses a piece and its location to move

Precondition: There are moves available

Action: Player chooses a man to move and its new location

Postcondition: Move is made and notation is recorded.

6.1.4. Losing a man

6.1.4.1. Opposing player makes a move which causes loss of man

Precondition: Opponent makes a move

Action: Player's man is overtaken

Postcondition: Player's piece is removed from board

6.1.5. Crowning a man

6.1.5.1. Player moves man to opponent's side of the board, king row

Precondition: Player makes a move

Action: Player moves onto king row

Postcondition: Player's man is crowned

6.1.6. Game End

6.1.6.1. An end game condition is met

Precondition: Game has been started

Action: End game condition has been met, see 3.2

Postcondition: Game is over

7. References

[1] http://www.usacheckers.com

8. Glossary

8.1. Checkers game:
The game that is being made.

8.2. Piece:
Objects that belong to a player that are moved around the board.

8.3. Crown Piece:
A piece that has moved across the board and reached the opposing team's first row.

8.4. Board:
The layout of all the pieces for the players to view and perform actions on.

8.5. Move:
An action made by the player to move a piece on the Board.

8.6. Capturing Move:
A move that includes "capturing" a piece. It steps over an enemy piece and the enemy piece is removed from the Board

8.7. Turn:
A sequence of moves made by one player, ending when no further moves are legal.

8.8. User Datagram Protocol (UDP):
A protocol used for transmitting data that is known for it's speed but is not very reliable for data loss.

8.9. Transmission Control Protocol (TCP):
A protocol used for transmitting data that is known for it's reliability but lacks speed.

8.10. Java Runtime Environment (JRE):
Development environment which will be used for development.

8.11. Networking:
Tasks to enable communication between a client and server.