Software Requirements Specification

for

Project 1 – Chess Game

Version 1.0 approved

Prepared by Madison Kuyawa, William Widmer, Bunrith Seng, Robert (James) Castleberry, Scott Flolid

CS 3398

February 01, 2018

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

The product whose software requirements are specified in this document is a chess game. The chess game will allow player vs. player (extra requirement) or player vs. computer. The scope of this SRS covers the entire system.

## Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

\*\*Do later

Abbreviations:

* PVP: player versus player
* AI

## Intended Audience and Reading Suggestions

This document is intended for the developers of the chess game, testers of the game, and the grader of the project.

## Product Scope

The purpose of this product is a game to be enjoyed by users. The objective of the product is to allow users to practice their chess skills and have the option to play against the computer or an opponent.

## References

[1] Software Requirements Specification, 1st ed. Karl E. Wiegers, 1999, p. 6.

[2]

# Overall Description

## Product Perspective

This chess game is a new, self-contained product. It is not part of a larger system.

\*\*<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

## Product Functions

The major functions of the product are:

* User shall choose to play either the computer or another human opponent
* If user chooses the computer as an opponent, then the user shall choose a difficulty level
* Once the user chooses their opponent and difficulty level, if applicable, then the user shall begin moving chess pieces and playing the game

## User Classes and Characteristics

The chess game shall have two user classes. The first user class is the player class, which is a required user class that will handle the human. The second user class, which is optional, is the AI user class which will handle the side of the AI if that option is selected.

## Operating Environment

Our chess game shall operate within any operating system because of the platform independent nature of java. The jar should be able to operate within any operating system (OS) using any hardware version.

## Design and Implementation Constraints

Any devices, except a mobile like device (since this is not an app). The minimum requirements are required in accordance to each IDE's requirements.

Recommended hardware configurations for an IDE, for example:

* Ubuntu
  + Processor: 2.6 GHz
  + Memory: 2 GB or better
  + Disk space: 850 MB of free disk space or higher
* Window
  + Processor: 2.6 GHz
  + Memory: 2 GB or better
  + Disk Space: 1 GB of free disk space or higher
* MacOS
  + Intel-based core
  + Memory: 2 GB or better
  + Disk space: 850 MB of free disk space

Our chess game shall be developed using java graphical user interfaces and such libraries. To help design and implement the GUI certain graphical prototyping and fxml tools may be used such as scene builder in javafx.

## User Documentation

A user manual shall be included within the chess application. This manual will go over the basic controls of the graphical user interface as well as a section to teach the basics of the game of chess to a user.

## Assumptions and Dependencies

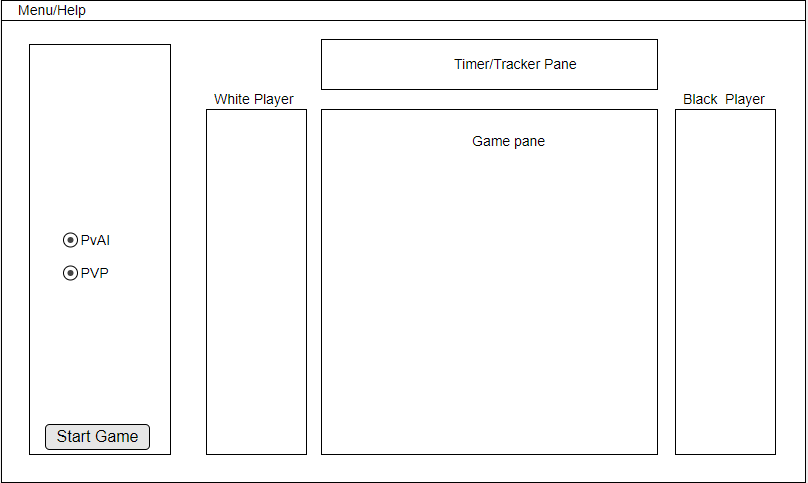
<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

Do research and come back to this

# External Interface Requirements

## User Interfaces

To allow for uniformity and robustness within the application our Graphical User Interface will be contained within a non-resizable window. The interface will contain 5 main sections

* A panel on the left wall that will handle setting up and difficulty selection of the game. Once a game has been started this pane will become a text are that displays the moves that are occurring within the game as well as when a player wins or loses. Once a game is finished the pane will switch back to set up mode
* A panel that will contain the actual chess game that will take up most of the space in the GUI. This pane will contain the board and the pieces that make up the game of chess. This is where players will be able to interact and actually play the game of chess.
* The timer and tracker panel will be directly above of the chess board and keep track of the game clock as well as an arrow that will indicate which player is currently taking their turn.
* The final two panes will be directly to the left and right of the board panel and be used as the player panels(One for white and one for black). These panels will also keep track of the corresponding players pieces as they become captured by the opponent. 

## Hardware Interfaces

The software shall be compatible with a connected mouse or keyboard that is connected to the system. The user shall be able to use these components to interact with the graphical user interface.

## Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

System clock for timer??

Out software shall be compatible with java fxml

# System Features

1. Software shall use a user interface
2. Software shall have two forms of AI implementation, a normal difficulty and a hard difficulty
3. System shall allow user input from mouse and keyboard.
4. The game shall begin from a normal starting position and will conclude when a "Check Mate" has been determined.
5. The game shall highlight legal moves for each piece once clicked (Highlight Legal Moves Feature).
6. The game shall allow for a player vs player format (PVP Feature).
7. The system shall keep track of which players turn it is using arrows to indicate which player is going (Player Turn Indicator Feature).
8. The system shall track moves and check states of the game in a text field once the game has been started.

## Graphical User Interface

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

4.1.1 Graphical User Interface

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

This system feature is a high priroty as it will be the central interface that will put everything together and make it interactable for the user. A 10 on the priority scale because without it we would have no game.

4.1.2 Stimulus/Response Sequences for GUI

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

The list of sequences for this system will include the following input form the player:

* User clicks the start button - This sequence shall allow for the game to begin and will switch the settings pane into a play history pane that will track moves made.
* User selects either Player vs Player or Player vs AI radio buttons – This shall toggle the game mode that will be played once the player clicks the start button
* User selects help menu item- This shall display a text window that will display a user manual for the Graphical User Interface and also a section for the rules of chess that our game will be implementing.
* User chooses either easy or hard AI difficulty button – This shall toggle the difficulty of the AI if the Player vs AI radio button is selected.
* Player selects a piece of their own color – This sequence shall cause the board to highlight all possible tiles that are available for that piece to move to.
* Player selects a highlighted space after selecting one of their own pieces- This sequence shall cause for the selected piece to now occupy the selected tile. Then the turn will be passed to either the next user player or AI player.
* User selects a new game button once the current game has concluded- This shall reload the setting pane over the move history pane so that a player can then choose setting for another game.

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1: When the application is started or jar is executed the graphical user interface shall initialize and display.

## A.I.

4.2.1 Description and Priority

This feature is a high priority because it includes an A.I. for a user to play chess against, including two difficulty levels. Since player versus A.I. is our most basic game mode, this will be a 10 on the priority scale.

4.2.2 Stimulus/Response Sequences

* User selects Player vs AI and chooses a difficulty – This sets the max depth of the A.I.’s move maker algorithm to either 2 or 5. The algorithm will stop at that depth when searching a tree of possible moves, meaning the A.I. will only be able to “think” that many moves ahead.

4.2.3 Functional Requirements

REQ-1: Libraries for move generation and recursive tree exploration

(Chesspresso-lib.jar, minimax and alpha-beta pruning algorithms)

## Handling Game setup and Game Ending

4.4.1 Description and Priority of Game setup and Game Ending

This system feature is intended to make sure that the game is playable this means that the game will start with a standard chess board setup and finish once a player is put into the position of checkmate. This system feature is a 8 on the priority scale since it makes sure that the game starts and ends properly and that the game is played correctly

4.4.2 Stimulus/Response of Game setup and Game Ending

* When the game has been started the board shall load into the starting positon.
* When the game reaches a checkmate state the game shall declare the winer and end the game.

4.4.3 Functional requirements of Game setup and Game Ending

Req-1: The game only allows for the movement of pieces to spaces that they are legally allowed to move.

Req-3: When a king is in check and has no possible way to get out of check the game shall declare a checkmate and decided the win er and end the game.

## Highlight Legal Moves Feature

4.5.1 Description and Priority of Highlight Legal Moves Feature

This system feature is a low priority as it an extra feature to help the player view their possible moves in the game. On the scale of 1-9, this feature receives a 2 because without it, the game would still work.

4.5.2 Stimulus/Response Sequences for Highlighted Legal Moves Feature

The list of sequences for this feature are as follows:

* User clicks on the piece they wish to move. There are six pieces the user may select to move: pawn, rook, bishop, knight, queen, and king.
* Once the user has selected a piece, certain squares will be highlighted, indicating where the user can move that piece. Below are what will happen for each piece selected:

1. Pawn:
2. Rook:
3. Bishop:
4. Knight:
5. Queen:
6. King:

* Finally, the user may choose to either move their selected piece to one of the highlighted squares or they may choose to select a different piece to view other available options.

4.5.3 Functional Requirements for Highlighted Legal Moves Feature

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## Player vs Player

4.6.1 Description and Priority of Player vs Player Feature

This system feature is of medium priority as it is an extra feature that will allow two people to play against each other instead of only against the AI. On the scale of 1-9, this feature receives a 5 because while without it the game will still function as designed but would add more for the end user.

4.6.2 Stimulus/Response Sequences for Player vs Player Feature

The list of sequences for this feature are as follows:

* User chooses Player vs Player (PVP) mode on start screen.
* One PVP mode is selected the game will start and the two players will then take turns moving their respective pieces
* Once “Check-Mate” has been detected the game will end.

4.6.3 Functional Requirements for Player vs Player Feature

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## Player Turn Indicator Feature

4.7.1 Description and Priority of Player Turn Indicator Feature

This system feature is a low priority as it an extra feature to assist the user(s) in knowing who’s turn it is to move. On the scale of 1-9, this feature receives a 4 because without it, although the player may be confused, the game would still work.

4.7.2 Stimulus/Response Sequences for Player Turn Indicator Feature

The list of sequences for this feature are as follows:

* Once the game begins, an arrow will indicate which player’s turn it is. The options are: the user them self, the computer (AI), or the other human opponent (user 2 in PVP).
* After the first player has completed their turn, the arrow will indicate it is the opposite player’s turn. This pattern will continue until a player has won and the game has ended.

4.7.3 Functional Requirements for Player Turn Indicator Feature

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## Text Field Logging

4.8.1 Description and Priority

This feature is not core to the functionality of the chess game, but provides important information to the user such as moves and when the game has ended. This will be a 4 on the priority scale.

4.8.2 Stimulus/Response Sequences

* When a user or A.I. makes a move, the move is logged and displayed in the text field.
* When a user or A.I. puts the opponent in checkmate, the game ends and the winner is logged and displayed in the text field.

4.8.3 Functional Requirements

REQ-1: javafx libraries, including TextArea and TextField

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

I think we can cut out this section

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>