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

## 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

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

[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

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>

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

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

None OR we will create a user manual

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

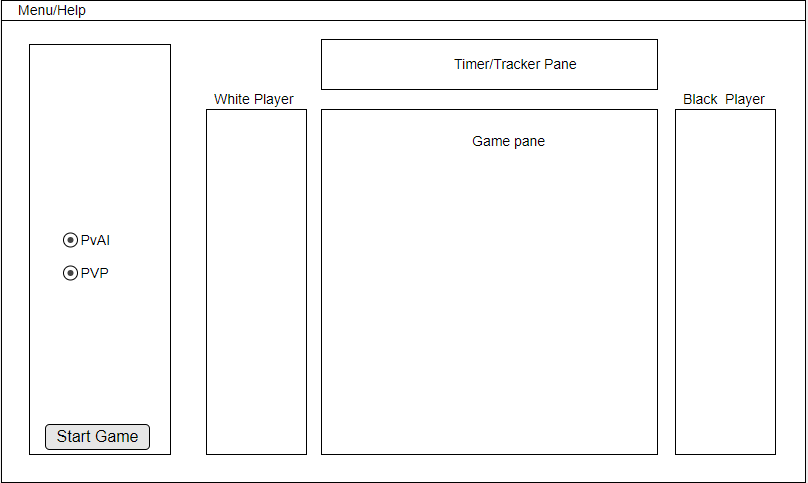
<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions 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

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, The game or combinations of these, whatever makes the most logical sense for your product.>

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.
6. The game shall allow for a player vs player format (PVP).
7. The system shall keep track of which players turn it is using arrows to indicate which player is going.
8. The system shall track moves and check states of the game in a text field once the game has been started.

## System Feature 1

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

4.1.1 Description and Priority

<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).>

4.1.2 Stimulus/Response Sequences

<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.>

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:

REQ-2:

## System Feature 2 (and so on)

# 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.>