SOFTWARE REQUIREMENTS SPECIFCATION

FOR

<CHESS ENGINE>

PREPARED->

<051-BSCS-19(ABDULLAH FAROOQ), 006-BSCS-19(AHMAD ABDULLAH)>

DATED

<14-11-2021>

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 purpose of this document is to provide a detailed overview of the requirements for chess engine, explanation of the purpose and features. Chess Engine will provide the user with a reliable, well-designed game of chess.

## Document Conventions

**CE**: Chess Engine.

**JRE**: Java Run-time Environment.

**AI**: Artificial Intelligence.

**SP**: Single Player

**MP**: Multiplayer

**VS**: Versus

**JDK**: Java Development Kit

**GUI**: Graphical User Interface

## Intended Audience and Reading Suggestions

The document is intended for teachers, users and gamers having a limited knowledge of software development. The rest of the document contains tools and strategies and the benefits and differences from the entire **CE** known. The reader should start from the description then to interfaces, system features and at last read the requirements of the product for knowledge purposes.

## Product Scope

This **CE** provides entertainment for players of all skill levels. This game also provides some useful feature including, indicating the valid moves, **SP** mode and **MP** (duo) mode.

There will also be difficulty levels for all types of players.

## References

This **SRS** does not refer to any other source.

# Overall Description

## Product Perspective

**CE** is a java based application with AI. The application will provide users with a simple way to play against a computer or a human opponent. The following main features will be added to the game: **Opponent:**

Player can play against **AI** or a human player (provides options to play against another player).

**Difficulty:**

Different **AI** difficulty settings will be available.

**Tips / How to Play:**

A section that will explain the basics of **CE** as well as some tips to get the user started.

## Product Functions

Following options will be available in the menu:

(Human vs. Computer) User can select the difficulty level before starting the game. Difficulty levels can be changed on a click.

(Human vs. Human) This is the default setting for **CE**. Two human players will be each-others rivals.

(Computer vs. Computer) At any time in the game, the computer can take hold of any user if option is selected.

There is option to change the scene for the game board and user interaction.

There will be information panel informing user about the moves and ranking the moves as user proceeds playing.

The engine will also provide hints to the user for a move.

## User Classes and Characteristics

The prospective user will be anyone with access to a computer that can run JRE. There is no privilege level / educational level / skills and experience required. The user is not expected to know chess, as the **CE** will include rules and suggestions / strategies.

## Operating Environment

It will be a java based desktop application so JRE will be used.

## Design and Implementation Constraints

**JDK** will be required for users to have them installed in their systems to run the application as the engine uses **JRE,** for its building. If user uses high difficulty levels and user does not have enough processing power then Computer will take some time to make its move against the user as computer has to think about the possible outcomes for each move it makes. It will also use parallel computing for some specific functions.

## User Documentation

If the user is a beginner in computer functions including clicking, moving the cursor. He/she can consult online about how to use a mouse. But, however if the user is a beginner for chess, he/she won’t need any consultation of documentation online as everything will be provided in the software itself.

## Assumptions and Dependencies

Since this will be a standalone application, there shall not be any other dependencies aside the requirement of having the **JDK** installed on the user system.

# External Interface Requirements

## User Interfaces

* This **CE** has **GUI** that takes input from the user as mouse click.
* Images of chess piece will follow the following theme:
* On the front page there are two buttons one for options and the other for new game.
* From option button user can select the theme and difficulty level game and mode of game.
* From the new game button new game will start with the option user has selected.
* By default, the game will start with Human vs. Human mode with Easy difficulty.
* User can move his piece by clicking on piece and direct it on the board.
* When a piece is selected, the square box where piece can move is highlighted to help the user.
* All possible boxes where piece can move are highlighted with different colors user can click on the box directly and generate move.
* If a selected move contains an opponent's piece, a different color will be used to highlight the square, indicating that it can be caught.
* If a player takes a move that captures an opposing player’s piece, that piece shall be removed from the board.
* A message appears on the board who wins the game or if the game has been drawn.

## Hardware Interfaces

Chess does not require any hardware other than a mouse for input, a monitor to display the game, and a computer to support JRE, a processor with high processing power, which is required to play game files.

## Software Interfaces

Chess software interfaces with the user computer and expect that it have **JDK** 14 environment installed on the system.

## Communications Interfaces

Communication between the client software and the network software is facilitated by common network protocols.

# 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, or combinations of these, whatever makes the most logical sense for your product.>

## 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 Nonfunctional Requirements

## Performance Requirements

Game should run smoothly.it should not be crashed. There is no specific performance requirement.

The time and space complexity are not explained. The overall goal is to run the program without any bugs or user issues.

## Safety Requirements

There are no safety requirements to run the engine.

## Security Requirements

There are no safety requirements. There is no database, no user information so no security is needed.

## Software Quality Attributes

Chess will move smoothly and will give the user a somewhat enjoyable experience of playing chess at a basic level or at advanced level.

## Business Rules

There are no business rules.

# 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.>

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.>