|  |  |  |
| --- | --- | --- |
| S.No | Description | Page No |
| 1. | INTRODUCTION   1. Purpose 2. Scope 3. Definitions 4. References 5. Overview | 3  3  3  3  4  4 |
| 2. | OVERALL DESCRIPTION   1. Product perspective 2. Product functions 3. User Characteristics 4. Constraints 5. Assumptions and Dependencies 6. Requirements Apportioning | 4  5  5  5  5  5  6 |
| 3. | SPECIFIC REQUIREMENTS   1. External Interface Requirements    1. User Interface    2. Hardware Interface    3. Software Interface    4. Memory Constraints    5. Operations 2. Functional Requirements 3. Use Cases    1. Move    2. Undo    3. Resign | 6  6  6  7  7  7  7  9  9  9  9  10  11 |
| 4. | GROUP | 12 |

1. INTRODUCTION

This section gives a scope description and overview of everything included in this SRS document. Also, the purpose for this document is described and a list of abbreviations and definitions is provided.

**1.1 Purpose**

This document specifies all the requirements for the Chess Game software. It will illustrate the purpose and complete declaration for the development of this software. It will also explain system constraints, interface and interactions with other external applications. These requirements specified relate to the functionality, constraints, performance, attribute and the system interface. The Chess program is a program used to play chess game. First goal will be that the program should be working and allow the users to play the game. Second goal will be to store the user data.

**1.2 S**co**pe**

This document describes the software requirements for the Chess program. This document will be used by the end-users, tester, and developers of the game.

**1.3 Definitions**

**Bishop**: one of two pieces of the same color that may be moved any number squares diagonally, as long as no other piece blocks its way. One piece always remains on White squares and the other always on Black.

**Castling**: to move the king two squares horizontally and bring the appropriate rook to the square the king has passed over.

**Check:** to make a move that puts the opponents King under direct attack.

**Checkmate**: a situation in which an opponent’s king is in check and it cannot avoid being captured. This then brings the game to a victorious result.

**Chess Board:** a board you need to play Chess. Have 64 black and white square.

**Chess:** a game played by 2 people on a chessboard with 16 pieces each.

**King**: the main piece of the game, checkmating this piece is the object of the game. It can move 1 space in any direction.

**Knight**: this piece can move 1 space vertically and 2 spaces horizontally or 2 spaces vertically and 1 space horizontally. This piece looks like a horse. This piece can also jump over other pieces.

**Pawn**: one of eight men of one color and of the lowest value usually moved one square at a time vertically and capturing diagonally.

**Player or user**: a user or a player will be the person that is playing the chess game.

**Queen**: it can move any number of spaces in any direction as long as no other piece is in its way.

**Rook**: one of two pieces of the same color that may be moved any number squares horizontally or vertically, as long as no other piece blocks its way.

**Stalemate**: a situation in which a player’s king is not in check, but that player can make no move. This then results in stalemate, which is a draw.

1.4 References

[1] IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, October 20, 1998.

[2] Karlsson J, “A Cost-Value Approach for Prioritizing Requirements”, Norges Teknisk Naturvitenskapelige Uni. 1997

1.5 Overview

The rest of this document describes the system requirements for the Chess program.

2 OVERALL DESCRIPTIONS

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce the basic functionality of it. It will also describe what type of stakeholders that will use the system and what functionality is available for each type. At last, the constraints and assumptions for the system will be presented.

2.1 Product perspective

University students need an entertainment tool to enjoy and play with friends. As described in section 1.2, of this document, CHESS intend to fill this need by providing them with software to do the same. It allows 2 players at a time to play chess. It provides a simple GUI where both the players will be able to select their moves alternatively.

2.2 Product functions

CHESS system will provide the following functions:

1. Record the time remaining to make the next move
2. Records of moves made in this game so far.
3. Records of pieces that each player killed.
4. Records of valid moves around selected pieces.
5. Records the user statistics and display them on request

2.3 User characteristics

The user of CHESS needs experience and should be able to play chess. Furthermore user needs to be very familiar and comprehend chess rules.

**2.4 Constraints**

CHESS may experience hardware limitations constrain for graphics and Java language requirements if installed on a non-compatible computer.

**2.5 Assumptions and dependencies**

CHESS is not platform dependent and can be installed in any operating system capable to run Java Run time environment. One more assumption about the software is that it will be used on a system with enough hardware resources or else there may be scenarios where the software do not work as intended. CHESS also requires Java installed on the user’s system and a small amount of storage space to save data.

2.6 Requirements Apportioning

The priority levels for the requirements are:

|  |  |
| --- | --- |
| **Priority 1** | All items of this level must be implemented and verified or the program is unacceptable! |
| **Priority 2** | Requirements of this priority are expected to be implemented, but their omission will not result in an unacceptable program; so long as their omission does not affect higher priority components. |
| **Priority 3** | Lowest priority and not expected to be implemented in the current release. |

3 SPECIFIC REQUIREMENTS

This section contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

3.1 External Interface Requirements

This section contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

**3.1.1 User Interfaces**

A first-time user should be able to register as a new player. A new user should not be able to play the game without registering. An old-user should be able to select his/her name from the list of available users. Once the users are selected, they will be showed the Mainboard with all the pieces placed at their starting position. CHESS includes an interface resembling a common chessboard. The CHESS is portable and the clients will just have to run an executable file that will be used to play. An interface is provided to the game players which display player information .As players make moves the screen is updated to reflect the moves made in the game.

Figure 1 will show the same default screen with chess pieces in their starting positions.

Figure 2 shows the game partially in progress with pawn pieces moved.

User Interface is subject to change while in development

3.1.2 Hardware interfaces

CHESS users the hardware mainly to obtain the inputs from the user using input devices CHESS runs on any computer hardware meeting the following criteria:

* Includes a Keyboard
* Includes Memory Storage
* Includes a Mouse / Touch Screen
* A Display Device for the GUI

3.1.3 Software interfaces

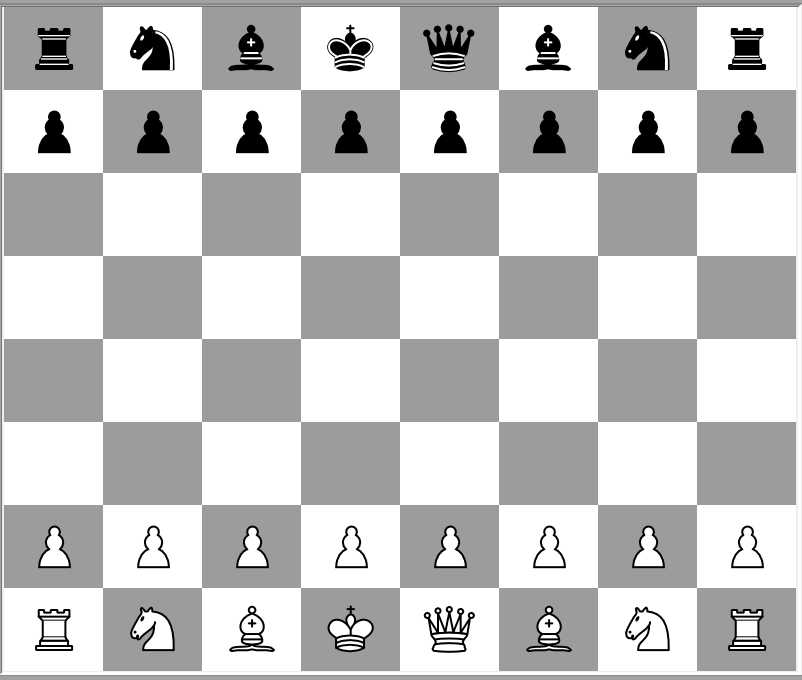
CHESS software integrates some external software like Java Runtime Environment to provide some of its functionalities.

3.1.4 Memory constraints

CHESS software requires 512 MB of RAM memory or more. It also requires a free space of 500 KB on secondary storage

3.1.5 Operations

CHESS does not provide backup or recovery operations.



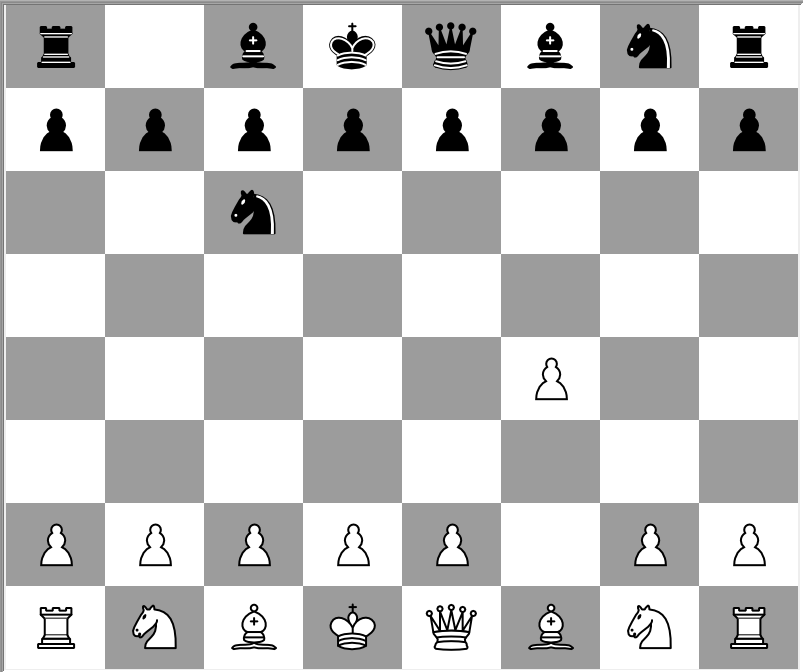


Figure 2 Prototype Screen with sample move

Figure 1 Prototype Default Screen

3.2 Functional Requirements

This section includes the requirements that specify all the fundamental actions of the software system.

* Users shall be able to download/obtain the executable jar and run it on any platform. **Priority 1**
* Users shall be able to start a game once two users are available. **Priority 1**
* Users shall be given the choice of who plays black and white. **Priority 1**
* The player playing white is first to move. **Priority 1**
* A player may forfeit at any time during gameplay. **Priority 1**
* A player must be given a confirm dialog before forfeiting. **Priority 2**
* Forfeiting shall end the game immediately. **Priority 1**
* The active player must receive information about the remaining time to make his next move. **Priority 1**
* The Player can check their statistics. **Priority 1**
* The active player shall select a piece by clicking it. **Priority 1**
* When a piece is selected, all legal moves for that piece are highlighted. **Priority 1**
* When a piece is selected, the active player may select another piece by clicking it. **Priority 1**
* A selected piece must always belong to the active player. **Priority 1**
* The active player shall move the selected piece by clicking on any legal square. **Priority 1**
* The active player shall capture a piece by moving onto a legal square containing an opposing piece. **Priority 1**
* Captured pieces shall be displayed in a captured pieces box. **Priority 3**
* The inactive player may request to undo the prior move. **Priority 2**
* There shall be no more than one undo request per turn. **Priority 2**
* A player shall be able to save a log of the moves at any time. **Priority 2**

**3.3 Use Cases**

3.3.1 Move

Precondition: During the Play Game state

Main Flow: The active player clicks a piece to select it. The game displays the positions it can move to. The player selects the new destination by clicking. The piece is moved there if it is a valid move. Their opponent becomes the active player.

Alternate Flow:

The active player may decide to select a different piece by clicking one of their own. If there are no valid moves and the active player is not in check the game ends as a stalemate. If there are no valid moves and the active player is in check the inactive player wins.



**3.3.2 Undo**

Precondition: During play game state, and an undo hasn't been requested this turn.

Main Flow:

1. The inactive player may request to undo their prior move.
2. The last move is then reversed and the inactive player becomes the active player
3. Time Remaining is changed accordingly

**3.3.3 Resign**

Main Flow:

1. At any time during gameplay, a player selects the forfeit option
2. The player then confirms the forfeit
3. The game is immediately ended with the forfeiting player as loser

Alternative Flow: The player decides not to confirm the forfeit. Game player resumes as usual.