Software Requirements Specification

for

CentipedeArmy Chess Game

Version 1.0 approved

Prepared by Emmanuel (Tobi) Afolayan, Drew Grubb, Jed Hutto, Jesse Miara, Ryan Weeks

CS 3398.264

February 2, 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

2. Overall Description 1

2.1 Product Perspective 1

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 User Documentation 2

2.6 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 4

3.3 Software Interfaces 4

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 5

4.3 System Feature 3 5

4.4 System Feature 4 6

4.5 System Feature 5 6

5. Other Nonfunctional Requirements 7

5.1 Performance Requirements 7

5.2 Safety Requirements 7

5.3 Security Requirements 7

5.4 Software Quality Attributes 7

5.5 Business Rules 7

Appendix A: Glossary 8

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Ryan Weeks | 2/16/18 | Format inconsistencies | 1.1 |
|  |  |  |  |

# Introduction

## Purpose

The purpose of this project is to simulate a game of Chess using the Waterfall model. The goal of this game is to allow a user play a game of Chess with an AI. Another goal is to allow two users to play against each other.

## Document Conventions

The priorities for each requirement is stated in the Description and Priority section.

## Intended Audience and Reading Suggestions

This document is intended for developers, testers, documentation writers, and project managers. This document contains the feature requirements for the game of Chess. Section 2 contains the overall description of the project. Section 3 contains information on what interfaces will be used in this project. Section 4 contains system requirements for the project. Section 5 contains the nonfunctional requirements.

We suggest that developers begin reading at section 2 to grasp the overall product perspective. Then, read section 3 and 4 to get an understanding of the interfaces that the project will use and the feature that the project will include.

We suggest that the testers start with section 2 for the overall product perspective. Then read section 4 and 5 for the requirements of the project.

We suggest that the project managers begin by reading sections 1 and 2. Then read sections 4 and 5 to understand the project's requirements.

## Product Scope

The intended purpose of this software is to allow a player to play the game of chess against an AI or another player. The goal is to implement a working game of chess using the waterfall methodology.

# Overall Description

## Product Perspective

This chess simulation is a new, self-contained product that attempts to virtually replicate the standard rules and gameplay of chess, with a few additions to add variety. For example, with the implementation of an AI vs. AI mode, entire matches of the game can be observed for self-teaching. In addition, turn timers add a sense of urgency and force experts to modify their strategy.

## Product Functions

There are several key functions that the product must perform as well as let the user perform in this project.  
  
The project must:

* Provide a Graphical User Interface.
* Provide a menu system that changes game settings without taking too long or being too complex.
* Provide a functioning game of chess that follows the basic requirements and features laid out in Section 4.

The user must be able to:

* Interact with the Graphical User Interface with Mouse input.
* Easily understand and execute important project functions.
* Perform a complete start to finish game of chess without encountering compromising issues.

## User Classes and Characteristics

There is a single user type for this project. The system shall implement all the settings the user requests for their match as well as the type of game they want to play.  
The only situation where there are multiple "users" are in player vs player mode, where a turn system shall be implemented to avoid a multiple user problem.

## Operating Environment

One of the main selling points of Java is the user’s ability to “write once, run anywhere”. Therefore, this program shall be able to run in the Windows environment, the Mac OSX, and the Linux operating System. This project will be a Java standalone and shall not have any OS compatibility troubles resulting from the language itself because are no outside plugins or application components.

Due to Team Centipede writing the GUI for this project in the AWT library, it is not likely there will be any JRE version inconsistencies in any environment that will conflict with our version. AWT has been a part of Java since 1995 and has not been edited enough to create problems for the surface level of this project.

## User Documentation

For this project, we will be using the following the standard Javadoc documentation style and basic comments to provide simple and readable online assistance via the Java Platform API. These Javadoc comments will be present above every class and every non-setter-getter method.

The Javadoc above each class will provide a description explaining the purpose of the class, as well as the author(s), version, and date. The Javadoc above each method will be explaining the specific function of the method, as well as describing any parameters or return variables that need explaining beyond their name.

## Assumptions and Dependencies

Time constraint is the only possible factor that will prevent the implementation of any feature stated in the SRS.

# External Interface Requirements

## User Interfaces

The primary interaction between the software product and the user comes from a menu system that varies based on where the user is in the simulation. For example, the options available to the user upon opening the application shall significantly differ from the menu options available to the user when they've paused in the middle of a game.

Our project shall contain three main Graphical User Interfaces.

1. Menu GUI:

This is where the user selects their settings or game type. The menu shall consist of at least 3 parts:

The “initial” screen: [Play vs Player | Play vs AI | Watch AI vs AI]

The “settings” screen: [Casual | Timed]

The “difficulty” screen: [Easy | Medium | Hard]

The game then begins with the determined settings.

2. PvP or PvAI GUI:

Layout:  
 Chess board on the left 65% of the screen, and the in-game settings on the right 35% of the screen.

Buttons:

Clicking on a piece “selects” it.

Clicking on a tile moves a selected piece to that tile (if allowed)

Forfeit button

Restart button

Pause button (in timed mode only)

Turns alternate. AI obviously does not need any mouse input to act.

3. AI vs AI GUI:

There is no human player in the AI vs AI game state. Therefore, the only role of the Graphical User Interface is to give the user a “Spectator” role.

Buttons:

Cancel/quit the game (in which case no winner is selected, and all pieces or game states are reset.)

Pause/resume the playing field

Change the delay between each move (in milliseconds).

Error Handling:  
Errors shall be handled by simply not allowing the user to do something that would cause an error.  
  
Our software shall have user input in order to navigate through the GUI.

## Hardware Interfaces

This application shall be supported on most PC devices that contain a Java Virtual Environment. This application shall be compatible with a touch screen interface but will primarily work with a mouse or a trackpad.

## Software Interfaces

For this project, we will need some software components other than what is included in Java 1.8. We will be using an external library for the AI component of the chess game, and plan on using the included libraries for the GUI, the integrated timer, and the two-player mode. Everyone will be using Windows 10 as an OS and will be using Eclipse as an IDE.

# System Features

## The following section goes over each system feature of the chess application in detail, including a general description, response sequences, and the functional requirements.

## Timer Implementation

### Description and Priority

In each game, there shall be two distinct timers visible to players.

The primary timer shall display how long a player has until they automatically lose the game for using up all their allotted time (Timed mode only). It shall pause each time a turn ends, and control passes to the next player.

The secondary timer available shall display how long the game has lasted. When the game starts up, the timers shall start at 00:00 and count as each second passes in real time.

This feature will have priority 4.

### Stimulus/Response Sequences

The primary timer and the secondary timer shall begin once the game begins. The primary timer shall start counting down at the beginning of each player's turn. Once the game has ended, the secondary timer shall stop counting and display how long the game lasted.

### Functional Requirements

REQ-1: Timers will run independent of one another.

REQ-2: Once the primary timer goes to 00:00, the player shall lose the game

## Move Suggestions

### Description and Priority

When players select a piece they wish to move, the game shall display several paths that can be taken.

This feature will have priority 8.

### Stimulus/Response Sequences

The game shall allow the user to click on the piece they want to move, then highlight every possible square the selected piece can move to. If the user clicks on a non-highlighted spot, nothing will happen.

### Functional Requirements

REQ-1: The program shall highlight all squares on the board a selected piece can move to.

## Difficulty Modes

### Description and Priority

This feature shall be available if players want to challenge AI opponents. Each skill

level shall determine the skill level of the AI opponent. The lowest difficulty setting shall provide human players with a completely randomized AI while the highest difficulty shall be an imported chess AI.

This feature will have priority 6.

### Stimulus/Response Sequences

If the user picks the single player mode, the program shall ask them which difficulty they would prefer.

### Functional Requirements

REQ-1: The program shall offer two difficulty modes that the user can choose to compete with.

## Player versus Player

### Description and Priority

The chess game shall allow two players to play chess against each other on the same computer, in a turn by turn fashion.

This feature will have priority 9.

### Stimulus/Response Sequences

The chess game shall allow the user to choose a button on the home screen, and that button activates the player vs player mode.

### Functional Requirements

REQ-1: The program shall randomly assign each user to a specific color.

REQ-2: The interface shall clearly state which player's turn it is.

REQ-3: The interface shall clearly state if a player has a check, and which player it is.

REQ-4: The program shall declare a winner when there is one and ask the users if they want to play again.

## AI versus AI

### Description and Priority

This mode allows two AI players to compete a game that can be observed. Can use this mode to learn techniques or strategies they might not currently use.

This feature will have priority 5.

### Stimulus/Response Sequences

The speed of the AI's moves shall be adjusted based on the selected speed from the user.

The AI shall pause if the user selects pause.

### Functional Requirements

REQ-1: The AIs shall play against each other without user interaction affecting their moves.

REQ-2: The difficulty modes of each of the AI shall be selected before the game selected.

# Other Nonfunctional Requirements

## Performance Requirements

Due to the simplicity of this project, performance is not an issue. The system shall not cause any noticeable lag on a standard computer.

## Safety Requirements

This project shall be created with meticulous care to prevent any possible harms that could be caused from seizures or eye strain.

## Security Requirements

The chess game will not collect or handle any personal data of the users leaving the game with no security or privacy concerns.

## Software Quality Attributes

There are many quality characteristics in our chess game. Our game shall not crash due to user error or a display failure. Our game shall be portable and playable on any major Operating System.

## Business Rules

There are no operating principles for the chess game. Every user has equal rights and can perform similar functions under every circumstance.

Appendix A: Glossary

* AI- Artificial Intelligence: a software developed replication of human intelligence, allowing the software to operate without direct, manual control
* API- Application Programming Interface: an interface that communicates with software components regarding how to build a specific software, usually providing definitions and tools
* GUI- Graphical User Interface: an interface that allows users to interact with software with textless, visual tools, such as windows, scroll bars, and buttons
* IDE- Integrated development environment: a component of software development that allows programmers and users to analyze the software in question to looks for defects and attempt to debug the software
* Java- The programming language that was used to code this chess application, Java relies on an object-oriented design and a general-purpose implementation of various software components
* JRE- Java Runtime Environment: an application that allows users to develop and run coded software that relies on the Java programming language
* OS- Operating System: a part of the computer that provides both the hardware and software with common features
* UML- Unified Modeling Language: A language used to develop diagrams that visualize the design of software.