Software Requirements Specification

for

Ferret Army Chess

Version 1.0 approved

Prepared by A. Maxwell, J. Guerrero, A. Romualdo, J. Cole, M. Yi

Ferret Army

February 2018

Table of Contents

Table of Contents ii

Revision History iii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Definitions, Acronyms, and Abbreviations 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** |
| A. Maxwell | 02/05 | Update sections 1.1, 1.2, 1.3, 1.4, and 1.5 | 1.0 |
| J. Cole | 02/06 | Update TABLE 2. section 2.2 | 1.1 |
| A. Maxwell | 02/07 | Update sections 2.1, 2.2, 2.3, 2.4, 2.5 | 1.2 |
| A. Maxwell | 02/08 | Update sections 2.6, 2.7 | 1.3 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

The purpose of this document is to provide a detailed description of the requirements for the "Ferret Army Chess (FAC)" software. It will illustrate the purpose and declaration for the development of the FAC software. It also details the system constraints, interface, and expected interaction between the user(s) and the system.

## Document Conventions

## This document is written in accordance with the IEEE Software Engineering Standards Committee document titled "IEEE Recommended Practice for Software Requirements Specifications". This calls for the use of 12pt Times New Roman font, spaced 2.0 points separation between lines, with top margin: 0.6", bottom margin: 0.5", mirror images, inner margin: 0.75", outer margin: 0.75", Gutter: 0.5 ", header and footer 0.3" from edge. Due to readability this document has been formatted with 1” top, bottom, inner, and outer margins with 1.5 points separation between lines.

## Definitions, Acronyms, and Abbreviations

The intended readers of this document are the client, project managers, and marketing personnel associated with the development and deployment of the Ferret Army Chess (FAC) software. Due to this section 1.3 has been included to ease in the reading of this document. Important terms, definitions, acronyms, and abbreviations have been included in detail.

TABLE 1.

Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Computer | The system that represents the artificial intelligence of which a User can compete against. |
| User | A person interacting with the Ferrer Army Chess software. |
| Player | A User who has initiated a game against either another User or the Computer. |
| Bystander | A User who is observing a Computer vs. Computer game without making a Game Move. |
| FAC | Refers to Ferret Army Chess the software under development. |
| AIE | Refers to Artificial intelligence engine that makes moves for the Computer. |
| UI | Refers to User Interface the system by which the Player interacts with the FAC software. |
| GP | Refers to any of the Game Piece(s) which may be a Pawn, Rook, Bishop, Knight, Queen, or King |
| GM | Refers to Game Move which is the act of moving a Game Piece on the Board. |
| GB | Refers to the Game Board which is comprised of an 8 square by 8 square board with alternating colors which total 64 possible squares a Game Piece may occupy. |
| GE | Refers to the Game Engine which is collectively the code that runs the Game Pieces, Game Moves, and Game Board. |
| TP | Refers to the Test Plan used to test the functionality of FAC. |
| Capture | The act of a Player removing another Player’s Game Piece by replacing their opponent’s Game Piece with the attacking Game Piece thus Capturing said Game Piece. |
| Check | Refers to a Game Move where a Player’s King is under attack from another Player whether a User or the Computer. |
| Checkmate | Refers to a Game Move where a Player’s King has no remaining moves where said Game Piece is not under attack from another Player whether a User or the Computer. |

## Product Scope

The "Ferret Army Chess (FAC)" software is Ferret Army's take on the classic turn-based game of chess founded in northern India around 6th century AD. It will feature traditional chess moves between the user(s) and the games artificial intelligence engine.

The FAC software will initially only support local gameplay on desktop/mobile computers with the eventual goal of transitioning it into a web-based application to support online gameplay and platform independence.

Furthermore, the software shall feature local user vs. user, user vs. computer, and computer vs. computer gameplay with varying levels of support for beginning chess players to master the game. Features to support new chess players include highlighting potential moves, captures, checks, and checkmate.

## References

# [1] "830-1998 - IEEE Recommended Practice for Software Requirements Specifications - IEEE Standard", Ieeexplore.ieee.org, October 20, 1998. [Online]. Available: http://ieeexplore.ieee.org/document/720574/?reload=true&arnumber=720574. [Accessed: 04- Feb- 2018].

# Overall Description

## Product Perspective

The FAC software will be a new self-contained application comprised of three primary components the *user interface* (UI), *game engine* (GE), and *artificial intelligence engine* (AIE). Of these, the UI is comprised of two sub-components the *game board* (GB), and several *game pieces* (GP) which a player uses during the course of a game. The general relationship between each major component and sub-components can be seen in figure 1.

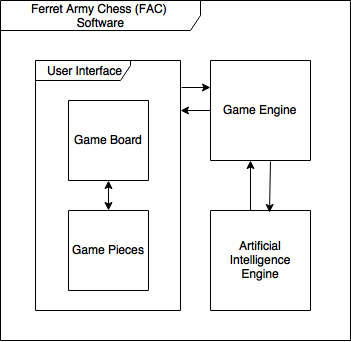


Figure 1. Component Overview

In the process of a user using the FAC software, there is a constant interfacing between the UI, GE, and AIE. When a user interacts with the GB this interaction is handled by the GE, which depending on the *game mode* (GM) may make calls to the AIE for a randomized move representing the *game move* (GM) of the Computer. In the event of a user. vs. user game the GE will initiate no calls to the AIE.

Since the FAC software is initially going to be designed for desktop/mobile platforms there will be no strong emphasis on restraining resources; however, when FAC is eventually adapted into a web-based application a series of optimizations will be applied to constrain resources according to hardware requirements of the web-server.

## Product Functions

With the FAC software the players, either a user or the computer, shall be able to have the ability to perform several key interactions. Those interactions are cataloged, in bulleted form, below in figure 2.

* User(s) shall be able to select from 3 game modes including computer vs. computer, user vs. computer, user vs. user.
* In addition to the standard chess moves the FAC software shall allow for 3 special moves including:
  + En passant – Special pawn Capture move.
  + Pawn Promotion – From a Pawn to any other Game Piece (GP)
  + Castling – Both Weak Side and Strong Side.

The specifics of each special move and what is entailed with these moves will be covered in-depth in section 4.

* User(s) shall have the ability to enable a timer to control the flow of the game. The time shall have functionality to support the following operations:
  + Game Time Limit – Default duration for the length of a single game
  + Turn Time Limit – Set a duration for the length of a single turn
  + Custom Time Limit ­– Change the Game Time Limit to a custom value
* User(s) shall be able to move any game piece (GP) according to that GP’s specific game move (GM) attribute. These moves are detailed in-depth in TABLE 2. shown on page 5.

TABLE 2.

*Game Pieces* and their associated *Game Moves*

|  |  |  |
| --- | --- | --- |
| **Game Piece** | **Game Move** | **Capture** |
| Pawn | Forward 1 space  Forward 2 spaces (Starting move only) | Left Diagonal 1 space  Right Diagonal 1 space |
| Rook | Forward 1-7 spaces  Backward 1-7 spaces | Same as Game Move until an opponent’s piece is captured. |
| Knight | Vertical 1 space and 3 spaces either horizontally left or horizontally right  Horizontally 1 space and 3 spaces vertically either up or down | Same as Game Move until an opponent’s piece is captured. |
| Bishop | Diagonal 1-7 spaces on the Game Pieces color of origin | Same as Game Move until an opponent’s piece is captured. |
| Queen | Diagonally 1-7 spaces Horizontally 1-7 spaces | Same as Game Move until an opponent’s piece is captured. |
| King | Diagonally 1 spaces  Horizontally 1 spaces | Same as Game Move until an opponent’s piece is captured. Must not be in check when Capture completed. |

* System shall include the functionality to provide color feedback for beginning chess players. This visual feedback shall include the ability to highlight:
  + Possible Moves – Based on User’s selected GP
  + Possible Captures – System shall show all possible captures for selected GP
  + Possible Checkmate – System shall show if a possible move will result in checkmate

In addition, the GB shall include game coordinates, so a User can submit moves using a coordinate on the GB. Furthermore, the User shall have the ability to enable or disable any of the aforementioned color feedback options.

* FAC software shall include functionality for limited artificial intelligence engine (AIE). Using randomization, the FAC’s AIE will move game pieces (GP) around the game board (GB).
* Users shall have the ability to change the appearance of the game board (GB) and the game pieces (GP) including support for traditional GP colors (Black/White) as well alternative colors.
* System shall implement a user interface (UI) allowing the User(s) to select game mode, display settings, and color feedback.
* Users shall have the option to quit an ongoing game at any time. It is not necessary for both Player’s to agree before an individual Player quits a game.

## User Classes and Characteristics

There are three users that will interact with the system: *Player(s)* of FAC, *Bystander(s)* of FAC, and AIE represented as the *Computer.* Each of these users have different interactions with the system and as such have their own unique requirements.

A *User* may interact with FAC as both *Bystander* and a *Player*. A *Player* may start a game with another *Player* or the *Computer*. The *Player* will have options to tailor a game to their skill level prior to starting a new game.

A *Bystander* is a *Player* that has chosen to setup and watch game between the *Computer* and another *Computer.* The *Bystander* will have no ability to interfere with the game and cannot make moves for the *Computer*.

The *Computer* is the manifestation of the AIE. It has no ability to change game settings, end the current game, or change a *Player’s* moves in any way. The *Computer* will interact with the FAC software only when requested by the *Game Engine* (GE).

## Operating Environment

Since the FAC software’s target platform is desktop and mobile computers the software must be able to run on any current operating system regardless of manufacturer. This means the FAC software must support both Microsoft and Apple Inc. operating systems.

Being operating system independent will promote a wider player base to draw from once the software is adapted into a web-based application. Once converted into a web-based application FAC will include support for mobile platforms in addition to the current support for desktop and mobile computers.

Due the scope of this document detailing the current build of FAC and not the future web-based application no description or detail of the web implementation will be included. The conversion documentation will be included in another software requirement specification once development reaches the refactoring stage.

## Design and Implementation Constraints

Due to the *Operating Environment* requirements laid out in *Section 2.5* the FAC software will need to be developed in a platform independent programming language to accommodate being operating system independent. The FAC software shall make use of any programming language libraries, API’s, SDK’s, or frameworks as long as the software scales to multiple platforms with no degradation in appearance or performance.

The efficiency of the *artificial intelligence engine* (AIE) is a potential constraint hindering performance of the *game engine* (GE). The AIE is responsible for providing all *Computer* moves as requested by the GE. Therefore, the need for clean efficient algorithms behind the AIE is important to overall system performance.

## User Documentation

Limited user support documentation will be provided with the FAC software upon delivery. However, to support refactoring, feature addition, and maintenance of FAC software by a future development team the software design document (SDD), test plan (TP), test cases and test result will be included as deliverables.

## Assumptions and Dependencies

One of the main assumptions made in this *software requirement specification* (SRS) document is that the chosen programming language will have and extensive library of tools to support rapid development through code reuse. A problem could arise if the selected language does not have extensive libraries for UI, data structures, and eventually web based interactions.

Additionally, for the *user interface* (UI) this document assumes that an off the shelf framework can be used to implement the *game board* (GB), *game pieces* (GP), and all necessary UI menus. This means the development team can have limit UI or *user experience* (UX) experience when implementing FAC. Should no UI framework exist to work with the chosen programming language this could potentially delay production until UI professionals are brought on board or another programming language is selected that supports such a framework.

# External Interface Requirements

## User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

## Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

## 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.>

## Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

# 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

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>

## Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

## Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the 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

This glossary is a duplication of TABLE 1. from section 1.3, which outlines the definitions, acronyms, and abbreviations used throughout this software requirement specification document.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Computer | The system that represents the artificial intelligence of which a User can compete against. |
| User | A person interacting with the Ferrer Army Chess software. |
| Player | A User who has initiated a game against either another User or the Computer. |
| Bystander | A User who is observing a Computer vs. Computer game without making a Game Move. |
| FAC | Refers to Ferret Army Chess the software under development. |
| AIE | Refers to Artificial intelligence engine that makes moves for the Computer. |
| UI | Refers to User Interface the system by which the Player interacts with the FAC software. |
| GP | Refers to any of the Game Piece(s) which may be a Pawn, Rook, Bishop, Knight, Queen, or King |
| GM | Refers to Game Move which is the act of moving a Game Piece on the Board. |
| GB | Refers to the Game Board which is comprised of an 8 square by 8 square board with alternating colors which total 64 possible squares a Game Piece may occupy. |
| GE | Refers to the Game Engine which is collectively the code that runs the Game Pieces, Game Moves, and Game Board. |
| TP | Refers to the Test Plan used to test the functionality of FAC. |
| Capture | The act of a Player removing another Player’s Game Piece by replacing their opponent’s Game Piece with the attacking Game Piece thus Capturing said Game Piece. |
| Check | Refers to a Game Move where a Player’s King is under attack from another Player whether a User or the Computer. |
| Checkmate | Refers to a Game Move where a Player’s King has no remaining moves where said Game Piece is not under attack from another Player whether a User or the Computer. |

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.>