|  |  |
| --- | --- |
|  | Software Requirements Specification |
|  |  |
|  | BubblePipe Chess Player v1.0  Steve Calabro, Mark Koh, Alex Mann, and Eric Most  1/20/15 |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Change purpose** | **Version** |
| Steve Calabro, Mark Koh, Alex Mann, Eric Most | 1/20/2015 | Initial version | 1.0 |

Table of Contents

[1. Introduction 3](#_Toc409690912)

[1.1 Purpose 3](#_Toc409690913)

[2. Description 4](#_Toc409690914)

[3. Specific Requirements 5](#_Toc409690915)

[4. Use Cases 6](#_Toc409690916)

[4.1 Use Case Flow 6](#_Toc409690917)

[4.1.1 Create an Account 6](#_Toc409690918)

[4.1.2 Logging In 6](#_Toc409690919)

[4.1.3 Create a Game 6](#_Toc409690920)

[4.1.3 Waiting for an Opponent to Join 6](#_Toc409690921)

[4.1.4 Joining a Game 7](#_Toc409690922)

[4.1.5 Playing a Move 7](#_Toc409690923)

[4.1.6 Opponent’s Turn 7](#_Toc409690924)

[4.1.7 Game Over 8](#_Toc409690925)

[4.1.8 View Stats 8](#_Toc409690926)

[4.1.8 Rejoin Game 8](#_Toc409690927)

[4.2 Use Case Diagram 9](#_Toc409690928)

[5. System Evolution 10](#_Toc409690929)

# Introduction

The Introduction section will discuss the motivation for the software system as well as provide a preview of the rest of the document.

## 1.1 Purpose

The prupose of this document is to give a detailed description of the requirements for the “BubblePipe Chess” software.

# Description

# Specific Requirements

## 3.1 Functional Requirements

### 3.1.1 Login Screen

R1.1.1 The application will allow the user to create an account. **Priority 1**

R1.1.2 The application will display a prompt for the user to input their username and password. **Priority 1**

R1.1.3 The application will authenticate the user's credentials with the server. Upon success the user will be directed to the home screen. If verification fails, the application will display a failure message and allow the user to try again. **Priority 1**

R1.1.4 The login screen will feature branding that distinguishes the software. **Priority 3**

### 3.1.2 Home Screen

R1.2.1 The home screen will have a menu that allows the user to:

R1.2.1.1 Create a new game. **Priority 1**

R1.2.1.2 Join an existing game. **Priority 1**

R1.2.1.3 Join an existing game by game id number. **Priority 2**

R1.2.1.4 View their game history and replay through old games. **Priority 3**

R1.2.1.5 View performance statistics. **Priority 3**

### 3.1.3 Server

R1.3.1 The server will be implemented using Java sockets. **Priority 1**

R1.3.2 The server will maintain a collection of games, complete with game id and user id numbers. **Priority 1**

R1.3.3 The server will store moves for individual games so if the connection is lost the user will be able to resume play where they left off once reconnected. **Priority 1**

R1.3.4 The server will receive and transmit moves between players. **Priority 1**

### 3.1.4 Game Design

R1.4.1 Each game will have a game id number and user id numbers for the players. **Priority 2**

R1.4.2 There will be a ChessBoard class which has the following properties:

R1.4.2.1 Board:ChessPiece[8][8] - An 8x8 array of ChessPieces. **Priority 1**

R1.4.2.2 state:ENUM - An enumerated state (white to move, black to move, white win, black win, draw). **Priority 1**

R1.4.2.3 validateMove(ChessMove):boolean - Checks that the user's move is legal. If the user enters an illegal move the game should display a message to the user, but otherwise do nothing. **Priority 1**

R1.4.2.4 update(ChessMove):void - Send the move to the server. **Priority 1**

R1.4.2.5 history:List<ChessMove> - The game history. **Priority 3**

R.1.4.3 There will be a ChessPiece abstract class which has the following properties:

R1.4.3.1 x:int, y:int - The piece's location on the board. **Priority 1**

R1.4.3.2 create():void - Instantiates a new piece on the board. **Priority 1**

R1.4.3.3 destroy():void - Removes a chess piece from the board. This occurs when an enemy piece moves to occupy the square this piece occupies. **Priority 1**

R1.4.3.4 generateMoves(ChessBoard):List<ChessMove> - For move validation. Each piece will be able to return a list of legal moves. **Priority 1**

R1.4.3.5 There will be individual chess piece classes (Rook, Bishop, Pawn, etc.) that extend the ChessPiece class. These classes will override the above functions where necessary (move generation) and provide additional functionality if required (promotion). **Priority 1**

R1.4.4 There will be a ChessMove class which has the following properties:

R1.4.4.1 id:int - The move number and id (ply). **Priority 1**

R1.4.4.2 fromX:int, fromY:int - The starting location on the board. **Priority 1**

R1.4.4.3 toX:int, toY:int - The ending location on the board. **Priority 1**

# Use Cases

## 4.1 Use Case Flow

### 4.1.1 Create an Account

**Preconditions**

* The user has opened the Application
* The user does not currently have an Account

**Main Flow**

1. The user is presented with “User Name” and “Password” and “Email” fields
2. The user enters their desired information and clicks “Submit”
3. If the desired username is already taken, the user is prompted to enter another one

**Post-conditions**

* The application returns to the Main Menu
* The user is now logged in

### 4.1.2 Logging In

**Preconditions**

* The user has opened the Application
* The user is not currently logged in

**Main Flow**

1. The user is presented with “User Name” and “Password” fields
2. The user enters their login info
3. If the login info is incorrect the user is re-prompted

**Post-conditions**

* The application returns to the Main Menu
* The user is now logged in

### 4.1.3 Create a Game

**Preconditions**

* The user has opened the Application
* The user is at the main menu

**Main Flow**

1. The user clicks the “New Game” button
2. The user is given a popup with the GameID on it

**Post-conditions**

* The game is in the “waiting for opponent” state

### 4.1.3 Waiting for an Opponent to Join

**Preconditions**

* The user has “Created a New Game”
* The opponent has not entered the game yet

**Main Flow**

1. The user observes a “Waiting for opponent…” message

**Post-conditions**

* The user waits for the opponent to join

### 4.1.4 Joining a Game

**Preconditions**

* The user has the GameID for the game they want to join
* The user is at the Main Menu

**Main Flow**

1. The user clicks “Join Game” button
2. A popup is displayed with a field called “Game ID”
3. The user enters their gameID
4. The user clicks the “Join” button
5. If the GameID is invalid, the user is re-prompted

**Post-conditions**

* The user will now see the game board and has joined the game

### 4.1.5 Playing a Move

**Preconditions**

* The user has joined a game
* It is the user’s turn

**Main Flow**

1. The user clicks the piece they intend to move
2. The user sees a list of available moves for that piece highlighted on the board
3. The user clicks the desired square
4. The piece moves to the clicked square
5. The game automatically removes any necessary pieces from the board

**Post-conditions**

* It is now the opposing players turn

### 4.1.6 Opponent’s Turn

**Preconditions**

* The user has joined a game
* It’s the opponent’s turn to move

**Main Flow**

1. The user observes a “Waiting for opponent…” message

**Post-conditions**

* It is now the user’s turn or game over if in end state

### 4.1.7 Game Over

**Preconditions**

* The user has joined a game
* A move was played that caused an end state

**Main Flow**

1. A “Game Over” message is displayed
2. The user’s stats are showed on the screen

**Post-conditions**

* The users may play again or return to the home screen

### 4.1.8 View Stats

**Preconditions**

* The user has opened the Application
* The user is logged in
* The user is at the main menu

**Main Flow**

1. The user clicks the “View Stats” button
2. A “User Stats” window pops up with the user’s stats

**Post-conditions**

* The user is viewing the “User Stats” window

### 4.1.8 Rejoin Game

**Preconditions**

* The user has opened the Application
* The user is logged in
* The user is at the main menu
* The user was previously disconnected from a game which their opponent is still in

**Main Flow**

1. The user clicks the “Rejoin Game” button

**Post-conditions**

* The user is presented with the game in the state it was left when disconnected

## 4.2 Use Case Diagram

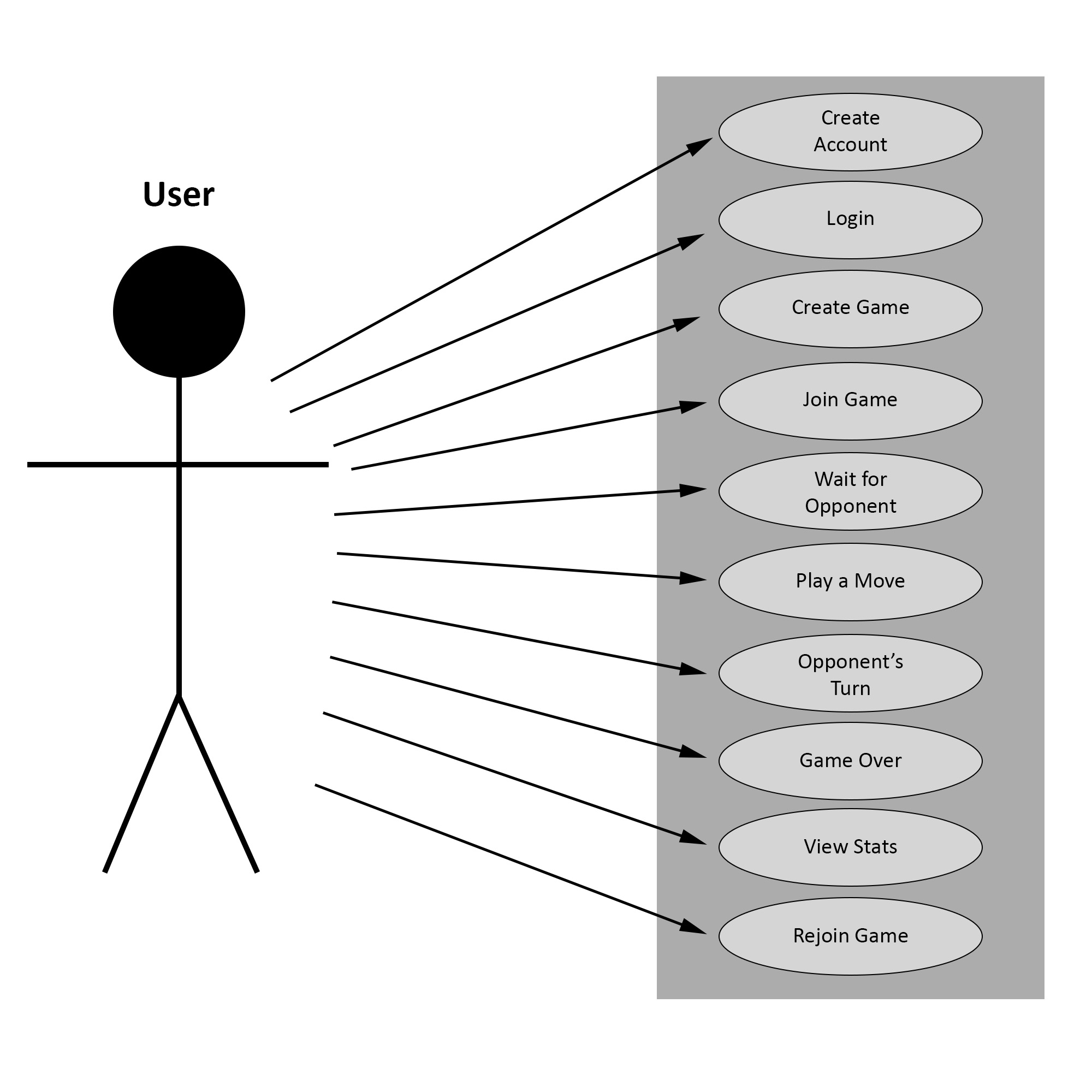


Figure : The Use Case Diagram for the application

# System Evolution