Software Requirements Specification

BubbleChess 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
Eric Most	2/26/2015	Added move validation requirements.	1.1

Table of Contents

1. 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 purpose of this document is to give a detailed description of the requirements for the "Bubble Pipe Chess" software. This document shows what the need of the project is and will show an extensive description of how the system will be developed. It will also show requirements of how the system will operate and interact with the users. This document is intended to be a detailed description so that is can be reviewed for customer approval and can be used by the development team as a guideline to start the project.

1.2 Scope

This document, which describes the requirements for "Bubble Pipe Chess," is intended to be used by the development team, testers, and users of this application.

1.3 Overview

This document will elaborate on the details and information dealing with the "Bubble Pipe Chess" software. It will also discuss the functional and non-functional requirements of the application.

2. Description

2.1   Product Perspective

"Bubble Pipe Chess" is an application which allows users to play an interactive game of chess over a network. The system will allow these users to select which game to join or to create a new game. The users will also be allowed to view their statistics and game history.

2.1.1   Desktop User Interface

The web interface of the application is the main interface that the users will interact with. Here, they will first see a page that allows them to create an account. Once they have created an account, they will be able to start a game or join another users game via the game id. During a game and after a game has been completed a user will be able to review their statistics on the home screen as well as game history by game id.

2.2   Product Functions

"Bubble Pipe Chess" will have the following functions:

- Allow users to create an account
- Play a game against an opponent over the web
- Ability to review user statistics
- Ability to review game history

2.3   User Description

This application has only one type of user. The user is a chess player. This user will utilize the application to find opponents to play a game of chess against, or to link up with friends who are challenging them to a game. The user will also be able to see statistics of the games as well as game history.

2.4   Assumptions and Dependencies

For this application the assumption is that the user has internet connection and the java runtime environment installed. If network connectivity creates an issue, we will provide the user with an error message letting them know that they will need to connect to a network to continue. If the user has an issue with the java runtime environment not being installed we will provide them with a link to the environment download from the java team. In addition to this, we assume that users will be able to play the game of chess following standard rules of chess.

2.5   Requirements Apportioning

Since some requirements are more critical than others, we assign them different priority levels. These levels are defined as:

Priority Level	Description
1	This is the highest priority level. Requirements of this level are absolutely essential to the product's functionality and must be fully satisfied in order for the software to be released.
2	Requirements of this level are not critical to the software's underlying functionality but are highly desirable in order to provide users with a complete experience. These requirements should be satisfied.
3	Requirements of this level are additions that will make the software competitive among other chess applications. They will be fulfilled after priority 1 and 2 requirements.

3. 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 Play

R1.4.1   The chess board will be displayed to each user with their respective pieces oriented at the bottom of the board. **Priority 2**

R1.4.2 The user will be able to:

R1.4.2.1   Offer a draw. **Priority 2**

R1.4.2.2   Resign. **Priority 2**

R1.4.2.3   Request to abort the game. **Priority 3**

R1.4.2.4   View the game notation. **Priority 3**

R1.4.3 The game will end automatically in the case of:

R1.4.3.1   The players agree to a draw. **Priority 2**

R1.4.3.2   Checkmate. **Priority 1**

R1.4.3.3   Stalemate. **Priority 1**

R1.4.3.4   Insufficient mating material. **Priority 2**

R1.4.3.5   Three-fold repetition. **Priority 3**

R1.4.3.6   A user runs out of time. **Priority 3**

R1.4.4   The game will implement a clock accurate to 0.1 seconds. **Priority 3**

3.1.5   Move Validation

R1.5.1   No move may be made that leaves the player's king under attack. This has priority over all other requirements. **Priority 1**

R1.5.2   A piece may not move to occupy another friendly piece's location. **Priority 1**

R1.5.3   A piece may capture, but not move past an enemy piece. **Priority 1**

R1.5.4   King

R1.5.4.1   The king moves one space in any of eight directions. **Priority 1**

R1.5.4.2   Castling. If the king and rook have not moved, the king is not attacked (including the squares it passes through), and R1.5.1 and R1.5.2 are met, then the player may castle by moving the king two spaces toward the rook. **Priority 1**

R1.5.5   The rook moves horizontally or vertically in four directions. **Priority 1**

R1.5.6   The knight moves in a "L-shape", one space horizontally or vertically, and then the two spaces in the other direction, and vice versa. The knight jumps so squares in between the start and end squares are irrelevant. **Priority 1**

R1.5.7   The bishop moves diagonally in four directions. **Priority 1**

R1.5.8   The queen moves horizontally, vertically, or diagonally in any of eight directions. **Priority 1**

R1.5.9 Pawn

R1.5.9.1	The pawn moves one or two spaces forward on its first move, and one space forward otherwise. **Priority 1**
R1.5.9.2	The pawn moves diagonally one space forward when capturing. It cannot capture a piece directly in front of it. **Priority 1**
R1.5.9.3	En passant. The pawn can capture an enemy pawn if the enemy pawn moves two spaces forward in an adjacent file past a square that the friendly pawn is attacking. The friendly pawn moves into the space it is attacking and the enemy pawn is removed from the board. **Priority 1**
R1.5.9.4	Promotion. When the pawn reaches the end of the board, the player has the option to promote it into a queen, rook, knight, or bishop. **Priority 1**

3.2 Non-Functional Requirements

3.2.1 Extensibility

R2.1.1 The application will be built in a way that allows it to be easily modified and expanded. Additional functionality will be able to be added to the existing framework and will not require a redesign of the software. **Priority 2**
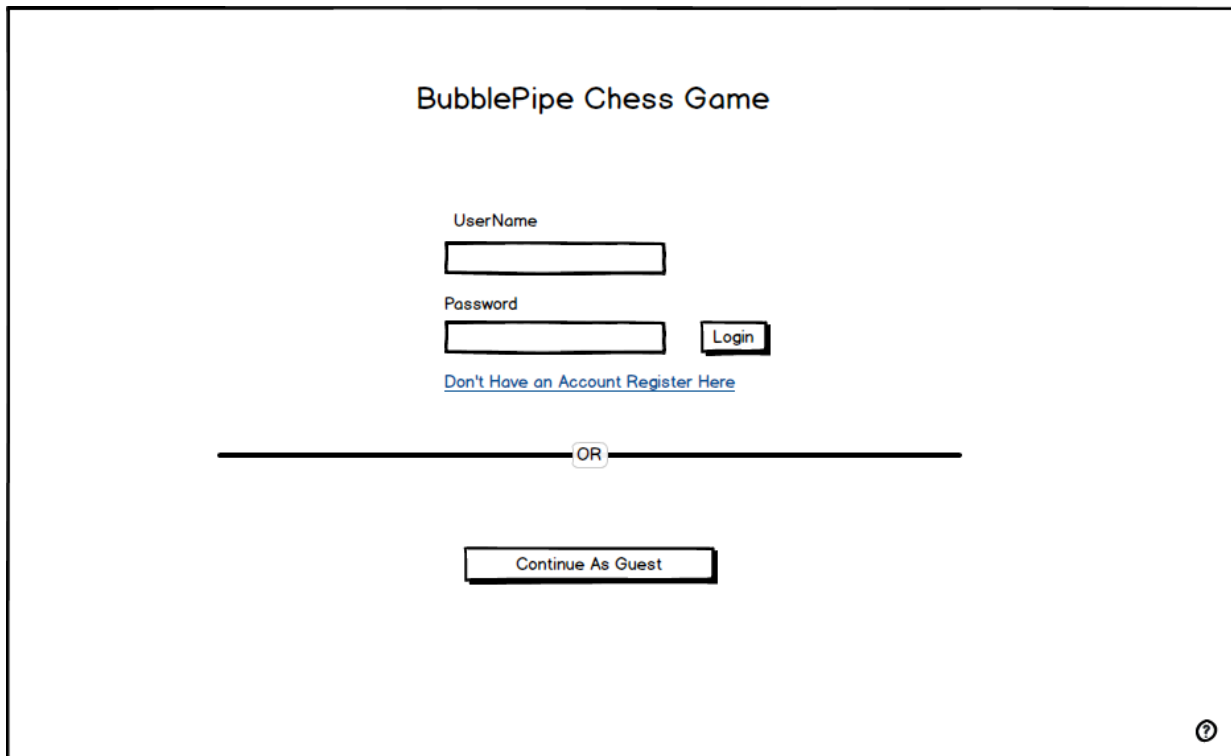
R2.1.2 The design will allow us to implement a chat window and Artificial Intelligence players if we choose to do so in the future. **Priority 3**

3.2.2 Maintainability

R2.2.1 The application will be developed using Github as our source control tool so that it will be easy to enforce version control, view code history and revisions, file bugs, and evaluate the software against the requirements. **Priority 1**
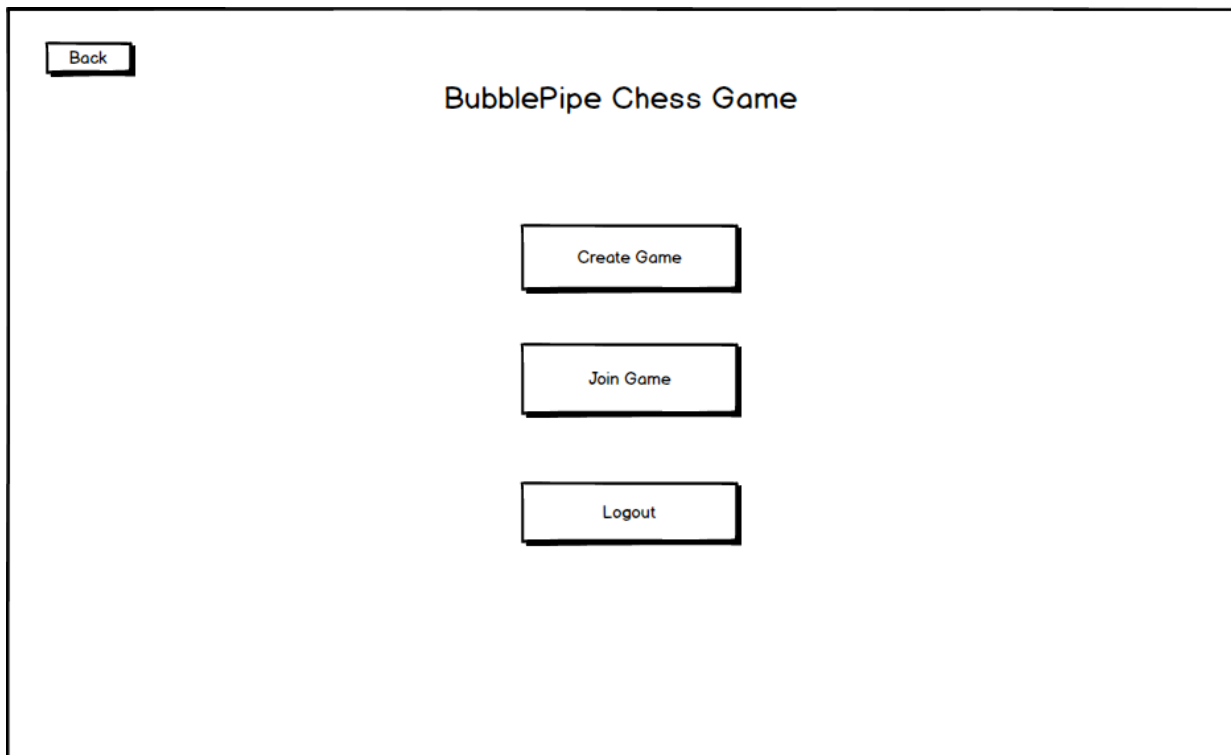
3.3   User Interface

Figure 1: The BubbleChess Account screen. This is the first screen that a user will see if they have not logged in previously.

Figure 2: The Main Menu. From here a user can choose to create or join a game as well as log out of their account.

Figure 3: The Create Game screen. A user can choose whether to make their game public or private.

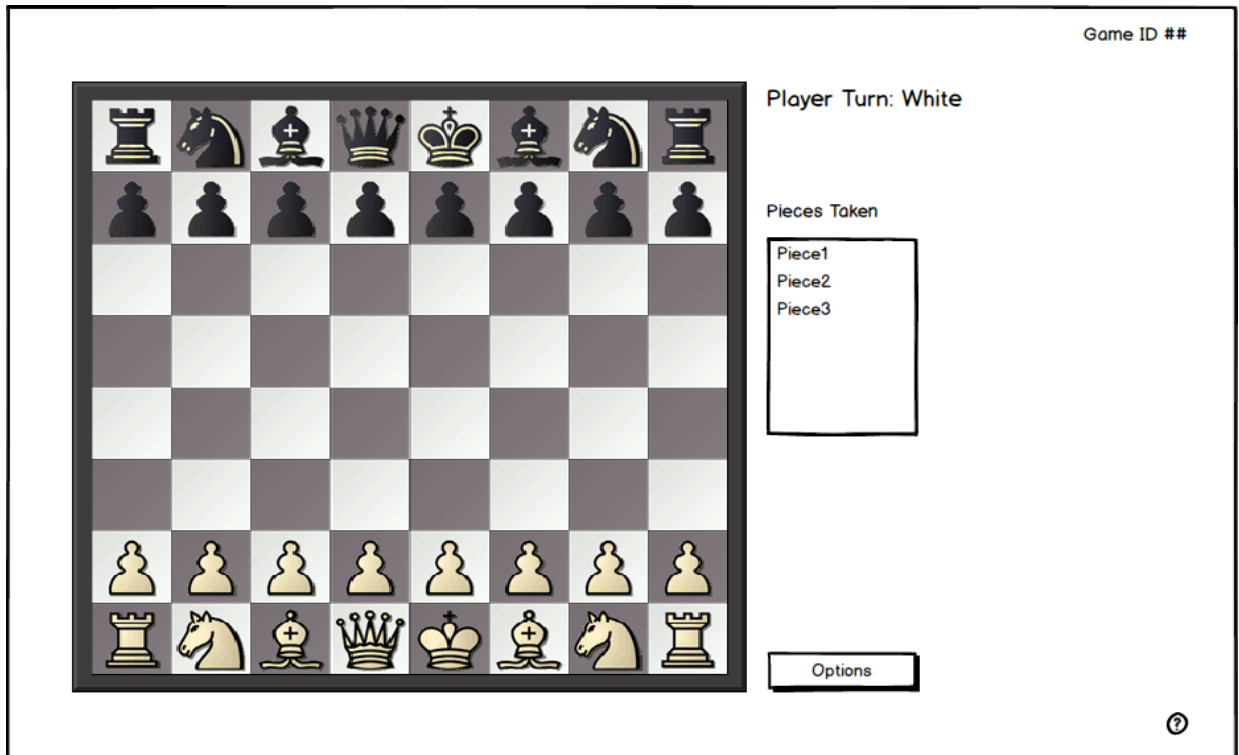*Figure 4: The Join Game screen. A user can choose a public game to join or input a GameID to join.*

Figure 5: The Game Play screen. The screen will initally look like such before any moves are made.

Figure 6: The Game Over screen. A player will see this screen after a game has ended.

4. 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
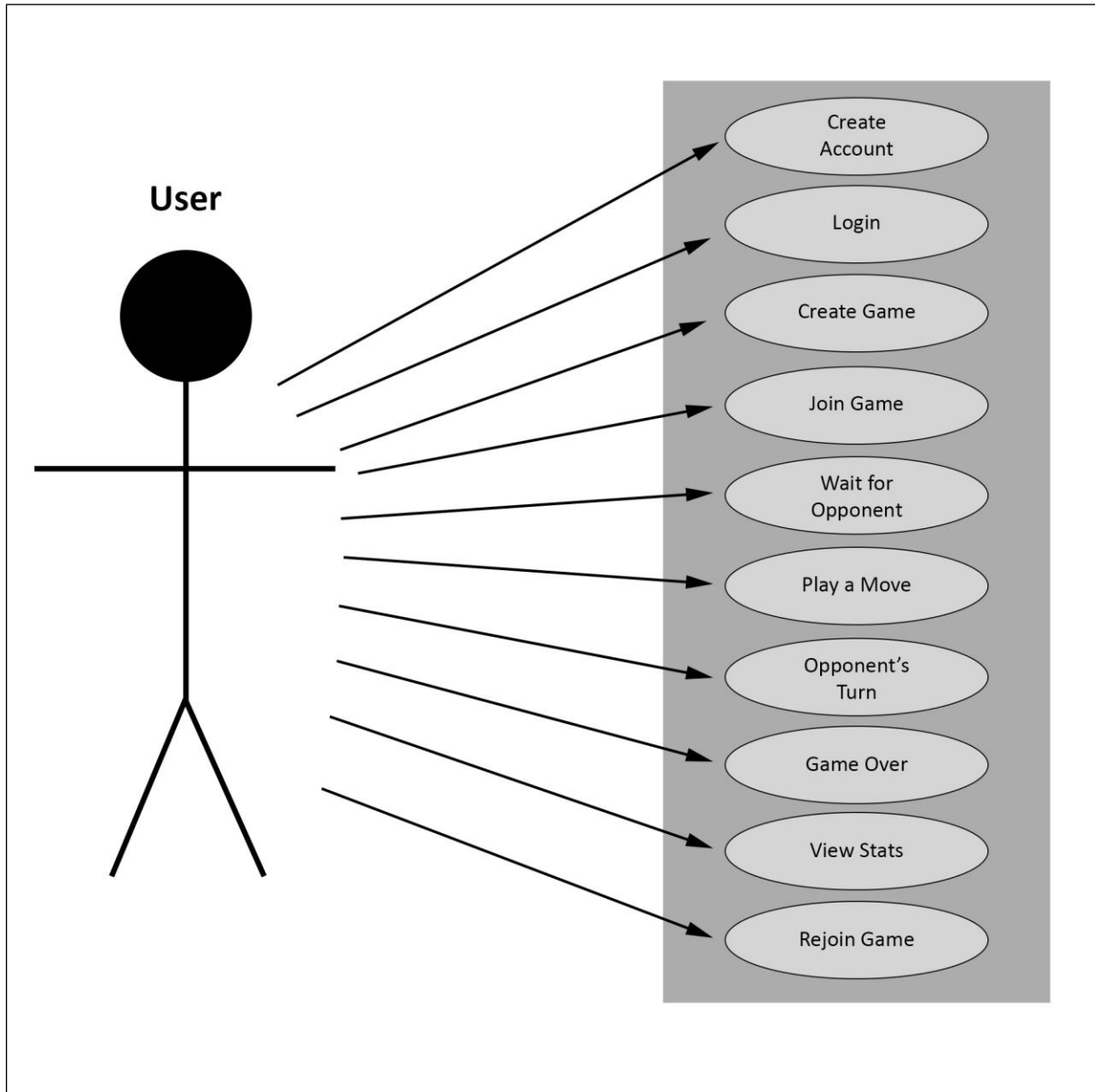
4.2 Use Case Diagram

Figure 7: The Use Case Diagram for the application
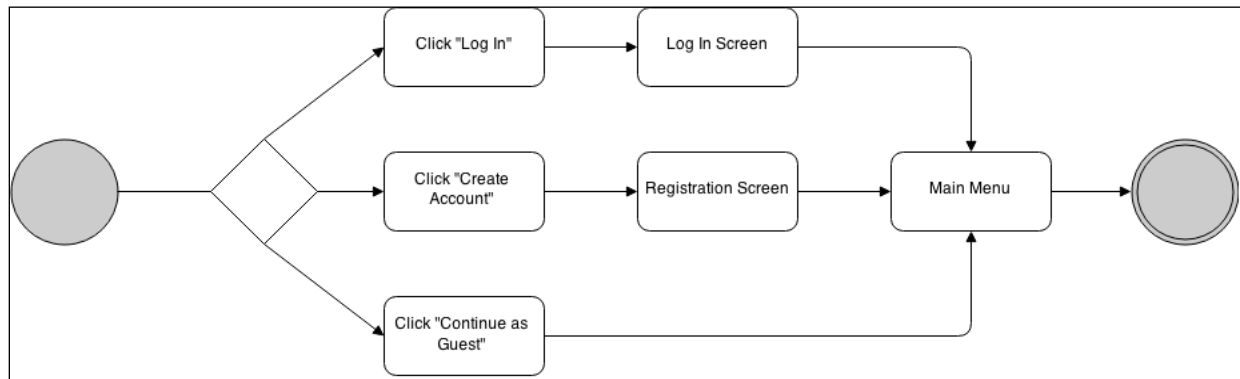
4.3 Activity Diagrams

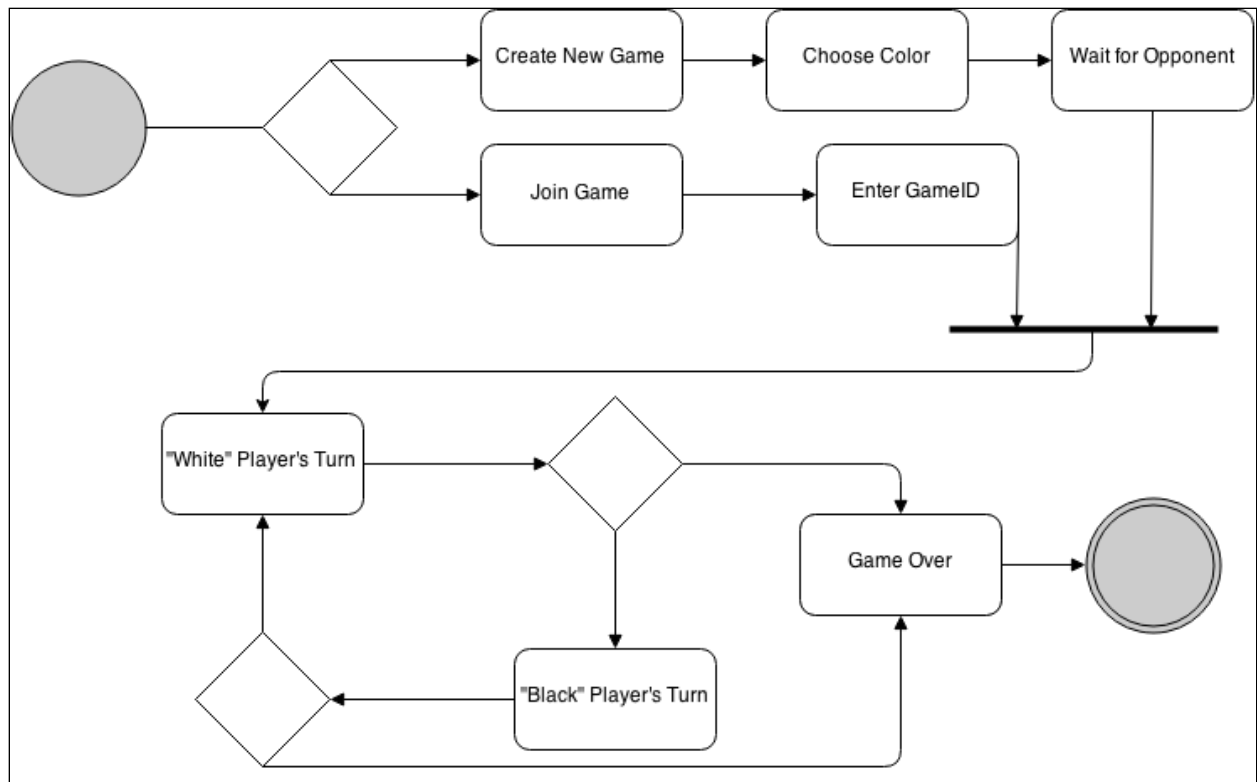*Figure 8: Activity Diagram for choosing an account option*

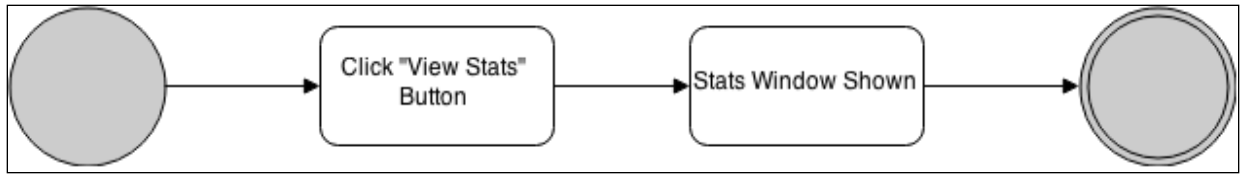*Figure 9: Activity Diagram for the game play process*

Figure 10: Activity Diagram for viewing user stats

5. System Evolution

While currently BubbleChess is designed only to handle two players over network with basic statistic keeping, the game could potentially do much more down the road. The system is designed for scalability in that while right now the functionality is not planned, BubbleChess could very well include a "Spectators" feature, Artificial Intelligence opponents, chess strategy tips, and much more. The addition of these features would be prioritized by user demand.

6. Appendix

6.1 Glossary

- **Abort** - A player may request to abort a game at any time. If their opponent agrees then the game is immediately ended with no result.
- **Checkmate** - A player is in checkmate when their king is attacked and they have no way to escape the attack, either by blocking the attack, moving away, or capturing the attacking piece.
- **Clock** - Most games are played with a chess clock that keeps time for both players. The players agree on a time and then they each have that amount of time to make all of their moves for the game. A player loses the game when he runs out of time unless his opponent lacks sufficient mating material.
- **Draw** - A tie game.
- **Insufficient mating material** - There are several endgame states in which it is impossible to achieve checkmate. When one of these cases arises the game is automatically ended in a draw. These cases are KN vs. K, KB vs. K, and KNN vs. K.
- **Notation** - A way to record the chess moves of a game. There are several different forms but in this project we will use starting square - ending square. Example: e2-e4.
- **Promotion** - When a pawn reaches the final rank of the board the player must promote it to either a queen, rook, bishop, or knight. These options are not dependent on the current pieces on the board. For example, it would be theoretically possible to have nine queens.
- **Resign** - A player may resign at any time and forfeit the game.
- **Stalemate** - A player is in stalemate when they have no legal moves but their king is not attacked.

- **Three-fold repetition** - When the same exact position arises on the board three times either player has the option to claim an immediate draw by the rule of three-fold repetition.