***Mosaic***

A system for networked Scrabble-like gameplay

Software Requirements Specifications

***Department of Computer Science,***

***Loyola Univeristy Maryland***

***Baltimore, Maryland, US***

*Alex Campbell – Programmer, Beat maker (the album drops 2015)*

*Rachel Fraczkowski – Marketing Administrator, Smooth talker*

*Armando Benavidez – Programmer, time-challenged lackey*

**1. Introduction:**

**1.1 Purpose:**

This document expands on the Software Requirements Specification (SRS) for the Mosaic game. Its purpose is to describe the scope, the functional and non-functional software requirements, and outlines, in detail, the design of the game logic and underlying infrastructure in its implementation. The infrastructure includes both the data sharing mechanics between entities of the system. The design of the game interface and view are also outlined within.

**1.1 Scope:**

The MOSAIC game logic system is an implementation of the American classic board game Scrabble to be used as the game logic component of the final product. This system dictates and enforces the rules of the basic Scrabble game; the doubling and bidding systems are not within the scope of this system. The game logic system contains a doubling system that defines and enforces all doubling events and conflicts; doubling allows for the weight or amount of points of a game to be doubled. The bidding system defines and enforces all the bidding events and conflicts; this system allows for some specified amount of points to be assigned to a placement of a word. Conflicts are resolved by the highest bidder wins. The overall network system allows for simultaneous game play; in other words, the cumulative moves of both players are shown at the same time. The MOSAIC invitation system allows for players to add individuals as their friends and allow for quick matching with them. All user data is contained within central database, the networking system is involved with interacting with the database system. The any logic or conflict handling related to the display or user interactions are not within the scope of any of the aforementioned systems.

**1.1.1 Within Scope:**

**1.1.1.1 Base Scrabble Game**

Follows the established rules of Scrabble, by Hasbro. This includes special letters, the ability to Doubling, letter placement, number of tiles, letter and special tile values, and end-game conditions.

**1.1.1.2 Bidding system:**

The game logic that deals with the allotment of points to

signify weight on some Player Move.

**1.1.1.3 Doubling system**

The game logic that coincides with the Bidding System; but handles specifically the case when a Player doubles a bid.

**1.1.1.4 Stat system**

The infrastructural system that houses User Statistics.

**1.1.1.5 Account system**

The infrastructural system that houses subscribed Users to the game server.

**1.1.1.6 PC platform**

Main targeted platform for the Mosaic Game.

**1.1.2 Out of Scope**

**1.1.2.1 Lobby / Invitation system**

A organizational space for users to view and interact with other users currently logged on to the server.

**1.1.2.2 User Pictures associated with accounts**

**1.1.2.3 Multiplatform**

Includes Android, iOS, and Windows Phone

**1.2 Stakeholders:**

This is the list of individuals that have stakes in the development of Mosaic, and details what relationship they have with the software and its development.

|  |  |
| --- | --- |
| **Dr. Glenn International Inc.** | The client for the developed Mosaic Game. Main Body |
| **Groovy Software Inc. Consultation** | The main developing firm tasked to implement agreed upon features that where detail by Dr. Glenn International Inc. |
| **Users** | Targeted audience of the final product, between the ages of 5 and up. These individuals have no direct attachment to before mentioned parties. |

**2.1 GLOSSARY**

The following table describes common terminologies in this document.

|  |  |
| --- | --- |
| **Mosaic** | The base game by which Mosaic’s gaming logic is defined. “Mosaic” is the name we will use to refer to the software project. |
| **Dictionary** | A sub-component of the system that contains both the data needed to define English words and the logic to search desire words. |
| **Home Screen** | The screen that allows the user to do 3 things: receive an invite, check their stats, and invite another user to play. |
| **Player** | An individual of the targeted audience. They are the individuals that interact with the completed game. They have no hand in the implementation of the system. They are the main focus of discussion; there is another later defined entity that plays along with the Player. |
| **Opponent** | Another individual interacting with the game alongside the Opponent. They are not the main focus of discussion and are thus named the Opponent of the Player. |
| **Player action** | Is some interaction with either the view or interface of the game. |
| **Player Move** | Includes the placing/removing of letters, adding/removing bets, and any other game related action that affects the Opponent or the outcome of the match. |
| **Bid** | An allotment of some amount of points on a particular Player Move. |
| **Confirm Word** | Telling the system that the tiles placed by a player form a word that the player wishes to submit |
| **Doubling** | The act of asking the player if they want the game to augment each person’s record by double the amount that it would normally. For example, if one player asks the other to double the game and the other player accepts, the winner will receive 2 wins added to their record, while the loser will receive 2 losses added to their record. A game can be doubled more than once (i.e., 2 accepted doubles will make the game worth 4 wins and losses). |
| **Draw** | The state of a game match in which both the Player and the Opponent have reached a stalemate, defined by the equal amount of points at the end of a Game Match. |
| **Forfeit** | When either Player or Opponent ends a game match before the normal end time. |
| **Place** | The Player Move that puts a letter piece onto the game board. |
| **Turn** | The three(3) minute or less allotted time used by a Player or Opponent to math series of Player Moves. |
| **Round** | Consists of a Player’s Turn and a Opponent’s Turn. |
| **Points** | The non-standard currency of a game. Player and Opponents receive points by doing the appropriate Game Move. |
| **Special tiles** | Specified tiles on a game board that depict some bonus attribute that leads to a beneficial allotment of points. |
| **Double Letter Tile** | A special tile on the board that doubles the amount that the tile is normally worth |
| **Double Word Tile** | A special tile on the board that doubles the amount that the word would normally score |
| **Triple Letter Tile** | A special tile on the board that triples the amount that the tile is normally worth |
| **Triple Word Tile** | A special tile on the board that triples the amount that the word would normally score |
| **Server** | Refers to the Mosaic game server. Handles logic and interaction between Player and Opponent. |
| **Master** | Is a mode in which the Player hosts the game state and changes it according to the received Opponent’s action. |
| **Slave** | Is a mode in which the Player merely received the associated Opponent’s actions and game state. |
| **Message** | The communication packets of data transmitted between Player to Server or Server to Player. |
| **Game Board** | Is the specific area in which the game pieces or letter can be placed. |
| **View** | The overall display or window of the game application. |
| **Interface** | Subsection of view that person interacts with. Includes objects that a User interacts with directly, such as game pieces or navigation buttons. |
| **Invite** | The invitation to a game by a User to another. |
| **User Statistics** | Includes data of past games played; wins, loses, 10 recently played Opponents. |

**3.1 Actors:**

|  |  |
| --- | --- |
| **3.1.0 Game system** | The system that contains all of the elements of the game |
| **3.1.1 User** | Interacts with the game system |
| **3.1.2 Tile system** | Manages the exchanging and distribution of user tiles |
| **3.1.3 Board System** | Manages the state of the board: tiles places, words, etc |
| **3.1.4 Dictionary System** | Checks words that have been placed on the board, validates |
| **3.1.5 Bidding System** | Manages and compares user bids on tiles, distributes points |
| **3.1.6 Doubling System** | Augments the “worth” of the game, relating to the user record: if the user wins, it will earn them 2x  wins, where x is the amount of times the game has been doubled |
| **3.1.7 Server** | Computes game logic, sends updates of game logic to users |
| **3.1.8 Database** | Holds user account info: username, password, email, etc |
| **3.1.9 Account System** | Allows a user to access their account, or create an account |
| **3.1.10 Invitation System** | Allows a user to select another user to play against |
| **3.1.11 Interface System** | Allows the user to interact with the software, depending on the state of the software |

**3.2 Use cases:**

This section contains usage scenarios of the software we are planning to implement. We outline the essential scenarios that will need to be tested to ensure that basic functionality of the software is working correctly.

**3.2.1 Login: UC1**

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Stakeholders and their interests** | User: wants fast, responsive way to gain access to their Mosaic account. Database system: wants organized, efficient way to store and retrieve user credentials |
| **Preconditions** | User has already created an account |
| **Successor Guarantee** | 1. Username and password match their respective fields in the database  2. User now has access to information for their account.  3. The user can now initialize the game |
| **Main Success Scenario** | 1. User is presented with fields to enter their username and password  2. User types their username and password in these fields  3. Both fields are sent to the account database  4. IF the username exists in the database and the password is correct for that user:  return a response that lets the Login system know that the  credentials match  5. User is taken to their Home Screen |
| **Extensions** | 1. If username doesn’t exist: inform user that the username doesn’t exist  2. If password is incorrect: inform user that the password is incorrect |
| **Special Requirements** | none |
| **Open Issues** | none |

**3.2.2 Create Account: UC2**

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Stakeholders and their interests** | 1. User: wants fast, responsive way to create an account for the game  2. Database system: wants organized, efficient way to store and retrieve user credentials |
| **Preconditions** | none |
| **Successor Guarantee** | 1. Username and password are non-zero length strings, email address is valid.  2. User now has a way to keep track of wins / loses, contact other players for games, etc  3. They can now play the game |
| **Main Success Scenario** | 1. User is presented with fields to enter their username, password, and email address  2. User types their username, password, and email address in these fields  3. All three fields are sent to the account database  4. IF the username does not exist AND the email address is valid:  Return a response that indicating that the account was created  Login system permits the user to play the game |
| **Extensions** | 1. If username exist: tell user to pick another username  2. If email address is not valid: Tell user to verify email address  3. If any field is blank: do not submit data to database, tell user that no field can be |
| **Special Requirements** | none |
| **Open Issues** | Are there limits to password / username lengths, symbols allowed / prohibited? |

**3.2.3 Send Invite: UC3**

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Stakeholders and their interests** | 1. User: wants a way to let other users know that they wish to play against them, while, adhering to established design standards when doing so |
| **Preconditions** | User has already entered their login credentials, and been successfully logged in. |
| **Successor Guarantee** | 1.User is notified that they have successfully sent an invitation  2. The user that they invited has been notified of this invitation  3. The user may play the game if the invitation has been accepted |
| **Main Success Scenario** | 2.User is presented with a field to enter the name of the user they wish to play  3. This username is checked against the Account Database.  4. IF the username exists in the Account Database and is logged on:  return a response that lets the Login system know that the  username exists  Let the User know that the invitation was sent  Let the user with the entered username know that the User  wishes to play with them |
| **Extensions** | 1. If username of the “invitee” doesn’t exist: inform user that the username doesn’t exist |
| **Special Requirements** | none |
| **Open Issues** | none |

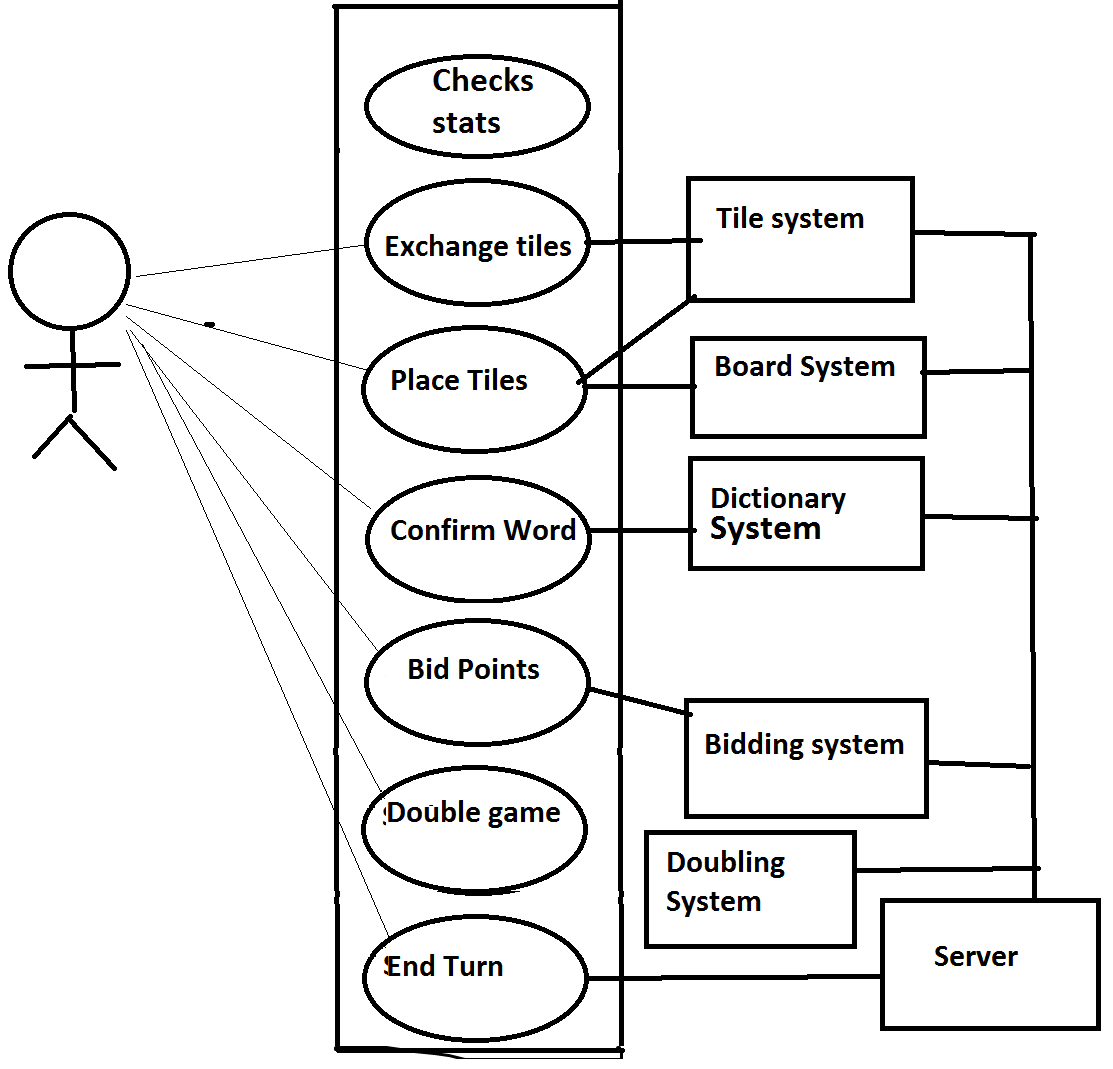
**3.2.4 Receive Invite: UC4**

|  |  |
| --- | --- |
| **Primary Actor** | User |
| **Stakeholders and their interests** | 1. User: wants a way to let other users know that they wish to play against them, while adhering to established design standards |
| **Preconditions** | 1. User has already entered their login credentials, and been successfully logged in. |
| **Successor Guarantee** | 1. User is notified that another player wishes to play against them  2. User can now play game against other player |
| **Main Success Scenario** | 1. User logs in  2. At some point, they receive an invitation to play against another player  3. IF the user accepts:  Change from login screen to Scrabble game screen. Start game with both players. Use case UC5(begin player turn) begins.  4. ELSE IF the user declines:  Reiterate to user that they declined the invitation |
| **Extensions** | none |
| **Special Requirements** | none |
| **Open Issues** | How are we accepting invites when the user is in game, or at the login screen, or at other various in-game states? |

**3.2.5 Begin Player Turn: UC5**

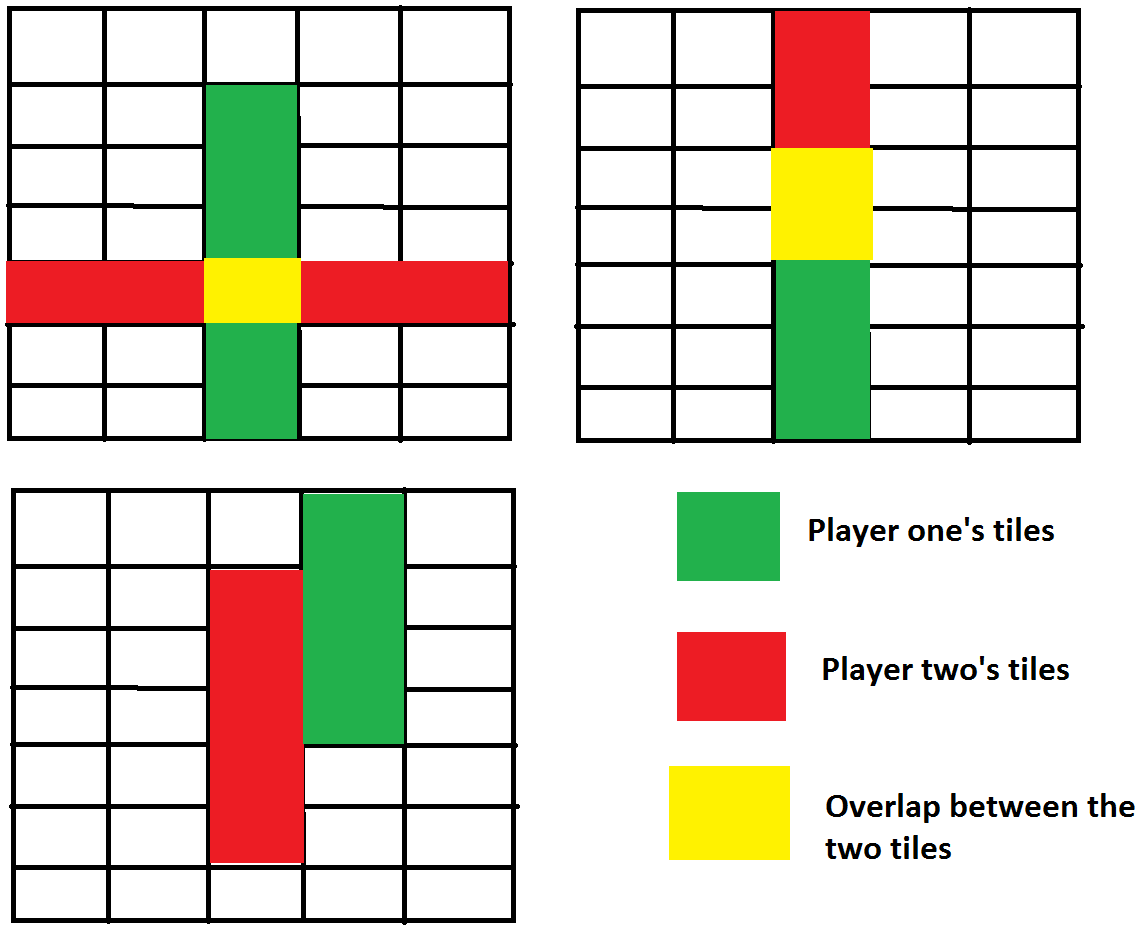
|  |  |
| --- | --- |
| **Primary Actor** | User, Game System |
| **Stakeholders and their interests** | 1. User: wants to play a fair game of Scrabble, and plays in such a way that is logical and adheres to the established game flow of Scrabble, and also being allowed to double the game and bid. |
| **Preconditions** | 1. It is the user’s turn, the Scrabble system knows that it is the user’s turn. |
| **Successor Guarantee** | 1. The player has peformed some combination of placing tiles, bidding, doubling, and exchanging tiles,  2. The round will be completed after |
| **Main Success Scenario** | 1. It is now the player’s turn.  2. If the user wishes to exchange a number of tiles for the same number of random different tiles, they may do so. This ends their turn  3. If the user wishes to place tiles to form a word, comprised of tiles that they own, they may do so.  4. If user wishes to double the worth of the game, they may do so. The other user is notified of the doubling at the end of the round  5. When the user is satisfied with the word comprised of their tiles, they can confirm their word  6. The word is a valid, English word.  7. The user can now bid, using a maximum of the points that they earn from the word, on the squares that they used.  8. The turn now ends. |
| **Extensions** | 1. If word is invalid, remove all tiles from board and give them back to the user  2. If the other player is not finished, the user must wait |
| **Special Requirements** | none |
| **Open Issues** | none |

**Use Case diagram for “Begin Player Turn”**



**3.2.6 Resolve Bid: UC6**

|  |  |
| --- | --- |
| **Primary Actor** | User1, User2 |
| **Stakeholders and their interests** | 1. User1 and User2: wants a way for the highest bidder to receive points for their word |
| **Preconditions** | 1. Both users have already placed words, and the round is over. |
| **Successor Guarantee** | 1. One user is notified that they have lost the bid  2. The other user is notified that they have won the bid, and received points  3. The new round starts |
| **Main Success Scenario** | 1. User1 has bid X points for the squares that they have used  2. User2 has bid Y points for the squares that they have used.  3. The squares that each player placed have intersected by at least one square, OR the players have placed their tiles in such a way that the tiles form invalid words (see diagram for an illustration of examples of tile conflicts).  4. The user with the highest bid gets the points that their word scored, minus how much they had bid, and their word stays on the board  5. The user with the lowest bid gets their word removed from the board  6. The round ends |
| **Extensions** | If the users tie, then remove both words from the board and award 0 points for both.  If there are no tile conflicts, the players each get (the value of the word – the amount bided) added to their scores, and both words stay on the board. |
| **Special Requirements** | none |
| **Open Issues** | none |

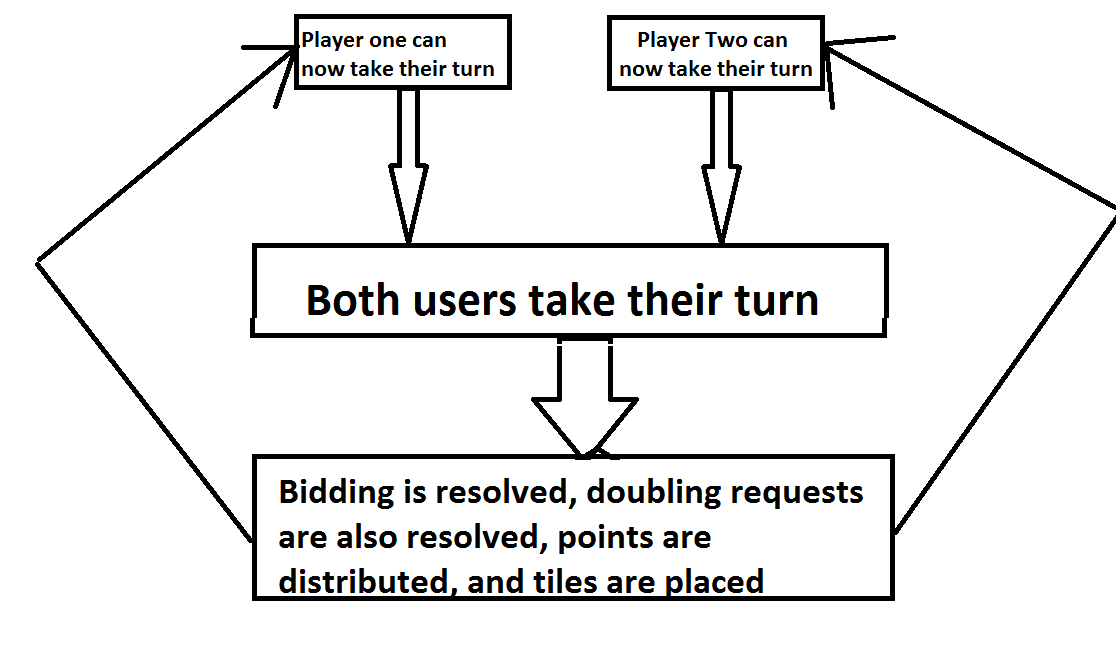
****

***Figure 3.1: Diagram of tile configurations that are considered “conflicts”***

**3.2.7 Play Round: UC7**

|  |  |
| --- | --- |
| **Primary Actor** | User1, User2, Server |
| **Stakeholders and their interests** | 1. User1 and User2: want a way to act out their turns for the game |
| **Preconditions** | 1. It is a new round |
| **Successor Guarantee** | 1. Player one has placed a word or has exchanged tiles  2. Player two has placed a word or has exchanged tiles  3. A bid has been resolved and the appropriate player has been awarded points  4. The appropriate words that have been placed remain on the board |
| **Main Success Scenario** | 1. User1 and User2 take their turns: they may exchange tiles, place letters, confirm a word, double the value of the game, and bid  2. The bid is resolved, doubling requests are either applied if it was accepted, or denied.  3. The winner of the bid’s word remains on the board, while the loser received their tiles back  4. Points are awarded appropriately (to the winner of the bid).  5. New random tiles are distributed so that each user has 7 tiles |
| **Extensions** | none |
| **Special Requirements** | none |
| **Open Issues** | Will accepting or doubling requests occur after a round? During? Anytime? |

**State diagram for Play Round:**



***Figure 3.2: state diagram for Play Round***

**3.2.8 Double Game: UC8**

|  |  |
| --- | --- |
| **Primary Actor** | User, Double system |
| **Stakeholders and their interests** | 1. User: wants a way to double the amount of wins and losses that will be applied to each user’s statistics at the conclusion of the game |
| **Preconditions** | 1. The user has not placed and confirmed a word yet, not have they exchanged any tiles. |
| **Successor Guarantee** | 1. The game is now worth 2x wins and losses, where x is the amount of accepted doubling requests. |
| **Main Success Scenario** | 0. The player begins their turn but does not exchange tiles or place any words  1. User uses Doubling system to double the value of the game, and sends a request to the opponent  2. A message is sent to the other user, notifying them of the Doubling  3. The other user can either accept or deny the Doubling.  4. If they accept, then the game will now be worth double the amount of wins and losses  5. If they deny, the win count and loss count will only be increased by one at the end of the game  6. Continue with the game as it was before the Doubling was sent |
| **Extensions** | none |
| **Special Requirements** | none |
| **Open Issues** | When exactly can the Doubling be issued? When will the other person be notified? |

**3.2.9 Exchange Tiles:**

|  |  |
| --- | --- |
| **Primary Actor** | User, Tile System |
| **Stakeholders and their interests** | 1. User: wants a way to exchange unwanted tiles for an equal number of random tiles |
| **Preconditions** | 1. The user has not placed and confirmed a word yet.  2. The user has at least 1 tile. |
| **Successor Guarantee** | 1. The user will no longer posses their unwanted tiles  2. The user will have new tiles that have replaced their unwanted tiles |
| **Main Success Scenario** | 1. User begins their turn  2. The user selects anywhere from one tile to N tiles that they wish to exchange, where N is the current number of tiles that the user has.  3. The tile system is informed that the user wishes to exchange their tiles  4. The tile system distributes X “random” tiles, where X is the amount that the user select, and “random” means a random selection from the list of remaining tiles  5. The user now has new tiles, and ends their turn |
| **Extensions** | none |
| **Special Requirements** | none |
| **Open Issues** | none |

**3.2.10 End Game:**

|  |  |
| --- | --- |
| **Primary Actor** | User1, User2, Game System |
| **Stakeholders and their interests** | 1. User: wants to know when the game ends, and who won the game |
| **Preconditions** | 1. There are no more tiles remaining in the Tile system  2. Neither user can form a valid word from their rack. |
| **Successor Guarantee** | 1. There will be no more rounds  2. A user will be chosen as the winner |
| **Main Success Scenario** | 1. User1 cannot form a word from their rack, and there are no more tiles  2. User2 cannot form a word from their rack, and there are no more tiles  3. A winner is chosen: the user with the most points wins  4. The game stops  5. Wins and losses are applied to each user’s record; the number depends on how many times the game has been doubled, if at all.  6. The users are taken to the Home Screen |
| **Extensions** | If there is a tie, no wins or losses are applied to the user’s record |
| **Special Requirements** | none |
| **Open Issues** |  |

**3.2.11 End Game:**

|  |  |
| --- | --- |
| **Primary Actor** | User, database |
| **Stakeholders and their interests** | 1. User: wants to check their wins, loses, and recent opponents. Info should be in a readable, logical format |
| **Preconditions** | 1. The user has an account that exists in the database |
| **Successor Guarantee** | 1. A user can now see their account statistics |
| **Main Success Scenario** | 1. A user chooses to view their account statistics  2. Their statistics are retrieved from the account database  3. The info is sent to the user’s interface  4. The user can now see this info. |
| **Extensions** | none |
| **Special Requirements** | none |
| **Open Issues** | none |

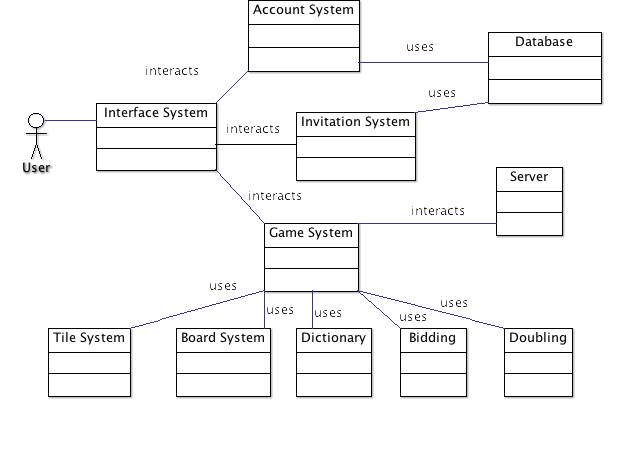
**3.2.12 Check Word:**

|  |  |
| --- | --- |
| **Primary Actor** | User, Dictionary System, Game System |
| **Stakeholders and their interests** | 1. User: needs to know if the word they placed on the board is a valid word according to an English Dictionary.  2. Game System: will behave in a way that is dependent on what the dictionary system determines. |
| **Preconditions** | 1. The user has placed a word on the board and has confirmed that they are done placing tiles. |
| **Successor Guarantee** | 1. The dictionary will have informed the user and game system whether or not the word placed on the board is a valid English word |
| **Main Success Scenario** | 1. A user places 1 – 7 tiles on the board  2. A user indicates that they are done placing tiles on the board  3. The dictionary system checks the word against its own list of valid words  4. The dictionary informs both systems whether the word is valid  5. If the word is invalid, the tiles the user had placed will be removed from the board and returned to the user.  6. If valid, the user will then bid, with points, for the squares that their word occupies. |
| **Extensions** | none |
| **Special Requirements** | none |
| **Open Issues** | What dictionary, or list of valid words, will we be using? |

**3.2.13 Forfeit Game:**

|  |  |
| --- | --- |
| **Primary Actor** | User, Game System |
| **Stakeholders and their interests** | 1. User: wants to be able to quit a game that has currently not been finished |
| **Preconditions** | 1. A game is currently in progress |
| **Successor Guarantee** | 1. A both users will no longer be playing the game |
| **Main Success Scenario** | 1. At any time during the game, the user can choose to forfeit the game  2. The other player is notified that the user does not want to continue playing the game.  3. The user that forfeits receives a loss on their record, while the other user receives a win on their record |
| **Extensions** | None |
| **Special Requirements** | none |
| **Open Issues** | If the game has been doubled, does this affect how each user’s record is affected? |

**3.2.12 Domain Model**



*Figure 3: Domain Model*

**3.3.1 Problem Domain**

This details what fields and disciplines the software relates to

**3.3.1.1 Scrabble**

This will relate to how the game of Scrabble actually works: the rules, the objects, and the limitations of the game

**3.3.1.2 Game Design**

This will influence the interface and “feel” of the game: what incentives will the user have to win a game? What do users normally expect from video games? What about the interface will video game players expect, and what will they not be familiar with?

**3.3.1.3 Interface Design**

The users will expect that our interfaces adhere to standardized interface design.

**3.3.1.4 Networking**

The game will allow two users to play over the internet.

**3.3.1.4 Database Systems**

The game will store user account details: usernames, passwords, game-playing stats

**3.3.1.4 Linguistics**

The game needs some way to check if a word placed on the board is in the English Dictionary

**3.4.1 List of Features**

This is a list of all the features that we intent to implement for the Mosaic software application. The features have numbers, from 12 to 0, indicating how important each feature is (12 being most important, 0 being least important).

**3.4.1.6 Game of Scrabble (12)**

Fully incorporates the game of Scrabble.

**3.4.1.7 Doubling system (11)**

Allows users to augment the effect of the game’s outcome: the wins and losses that will be added to a user’s account at a game’s conclusion can be doubled if both users agree to do so.

**3.4.1.8 Bidding system (10)**

Allows the users to bid on tiles that they want to use for the words that they place on the board. Whoever bids highest for the tiles will be allowed to place their tiles on the board, and will subsequently receive points for that word, minus the amount that they bid

**3.4.1.9 Interface (9)**

Allows the user to interact with the game via mouse clicks, mouse drags, and keyboard input

**3.4.1.10 Score (8)**

A quantitative representation of how skillfully a user is playing. It also the currency with which they bid for the tiles that their prospective words occupy.

**3.4.1.2 Database system (7):**

Contains user accounts, which include usernames, passwords, User Statistics, etc

**3.4.1.3 Account system (6):**

Holds all statistics and information of a user

**3.4.1.1 Login System (5):**

Allows users to access their account

**3.4.1.4 Invitation system (4):**

Allows users to communicate their intent to play a game

**3.4.1.5 Statistic / record keeping / viewing (3)**

Allows users to view and update their wins / losses, recent opponents

**3.4.1.11 [not implementing currently] Lobby system (2)**

This will be a system that will allow users to see who else is logged in to the Mosaic game system. From this list of users, a user can select a user to play and send an invite to. If the selected user accepts the invite, then a game will start.

**3.4.1.13 [not implementing currently] Account pictures (1)**

These will be pictures that will be associated with a User Account.

**3.4.1.12 [not implementing currently] Android (0)**

This will be an implementation of the Mosaic application on the Android mobile phone platform.

**4 Technology Used**

These requirements specify the technological aspects of the game software. Certain details of the exact systems to be used are out of scope of this document.

**4.1 Operating System**

Windows (PC) will be used as the base operating system for the word game software.

**4.2 Memory**

No more than 2 GB of memory usage will be used in order to run the system.

**4.3 Networking System**

A system of communication between the computers hosting the game and an established server will allow the game to sync between the clients.

**4.3.1 Dedicated Server**

The server system will be set up independently of the gaming computers

**4.3.2 Clients**

Client computers will also communicate with the server to send and receive data to the server.

**4.4 Database System**

A database system will keep user statistics, user account information, game logic and functionality. The server will communicate with the database to retrieve this information for game functionality.

## 5 Non-Functional Requirements

This section specifies the non-functional requirements of the word gaming system. Non-functional requirements exist in this system for Usability, Performance, Reliability, Availability, Security, and Maintainability.

**5.1 Usability**

The system will have a UI that follows standard conventions for Windows GUI. The further specification of the GUI like such as layout, view definitions, navigation, and accessibility is out of scope of this document.

**5.2 Performance**

The system should run fast enough so that 90% of transactions between the user and the server shall be processed in less than 5 seconds. The system should not crash during user interaction. System downtime may not exceed 2 minutes per day. The system should also support the interaction of at least 2 users, and have a maximum user count determined by the capacity of the production environment.

### 5.3 Reliability

The system should always work uncorrupted or produce the correct output, given the correct user-input from the GUI and network sub-components.

### 5.4 Availability

The system should be available to the GUI and network sub-components as long as the system is running.

### 5.5 Security

The system shall protect user data from unauthorized access. The system shall also provide authentication through password and username verification.

### 5.6 Maintainability

The system’s source code should maintain the following conventions:

* 1. Conventional Indentation for methods and java coding
  2. In-Line Comments
  3. Java Documentation
  4. All variable and class names should be clear, concise, and appropriate

**5.7 Scalability**

The overall performance of the system must increase if more powerful hardware is used for the host part of the system.

**5.8 Portability**

The system must be able to be converted to other platforms without edits to the conceptual structure, or gaming logic.

## 6 Design Constraints

The system has no particular design constraints, except for the scalability of the system. The system will not be hosting a large amount of users and the testing environment does not have the means to test extreme cases, high performance hardware, or multiple platforms.

**APPENDIX A:****Scrabble Rules**

Rules – Setup

You should have a game board, 100 letter tiles, a letter bag, and four racks.

Before the game begins, all players should agree upon the dictionary that they will use, in case of a challenge. All words labeled as a part of speech (including those listed of foreign origin, and as archaic, obsolete, colloquial, slang, etc.) are permitted with the exception of the following: words always capitalized, abbreviations, prefixes and suffixes standing alone, words requiring a hyphen or an apostrophe.

Place all letters in the pouch, or facedown beside the board, and mix them up. Draw for first play. The player with the letter closest to "A" plays first. A blank tile beats any letter. Return the letters to the pool and remix. All players draw seven new letters and place them on their racks. **Scrabble Word Game Playing Rules**

The Rules of the Game: Game Play

* The first player combines two or more of his or her letters to form a word and places it on the board to read either across or down with one letter on the center square. Diagonal words are not allowed.
* Complete your turn by counting and announcing your score for that turn. Then draw as many new letters as you played; always keep seven letters on your rack, as long as there are enough tiles left in the bag.
* Play passes to the left. The second player, and then each in turn, adds one or more letters to those already played to form new words. All letters played on a turn must be placed in one row across or down the board, to form at least one complete word. If, at the same time, they touch others letters in adjacent rows, those must also form complete words, crossword fashion, with all such letters. The player gets full credit for all words formed or modified on his or her turn.
* New words may be formed by:
  + Adding one or more letters to a word or letters already on the board.
  + Placing a word at right angles to a word already on the board. The new word must use one of the letters already on the board or must add a letter to it. (See Turns 2, 3 and 4 below.)
  + Placing a complete word parallel to a word already played so that adjacent letters also form complete words. (See Turn 5 in the Scoring Examples section below.)
* No tile may be shifted or replaced after it has been played and scored.
* Blanks: The two blank tiles may be used as any letters. When playing a blank, you must state which letter it represents. It remains that letter for the rest of the game.
* You may use a turn to exchange all, some, or none of the letters. To do this, place your discarded letter(s) facedown. Draw the same number of letters from the pool, then mix your discarded letter(s) into the pool. This ends your turn.
* Any play may be challenged before the next player starts a turn. If the play challenged is unacceptable, the challenged player takes back his or her tiles and loses that turn. If the play challenged is acceptable, the challenger loses his or her next turn. Consult the dictionary for challenges only. All words made in one play are challenged simultaneously. If any word is unacceptable, then the entire play is unacceptable. Only one turn is lost on any challenge.

The game ends when all letters have been drawn and one player uses his or her last letter; or when all possible plays have been made.  
  
**Scrabble Scoring Rules**

SCRABBLE Rules: Scoring

* Use a score pad or piece of paper to keep a tally of each player's score, entering it after each turn. The score value of each letter is indicated by a number at the bottom of the tile. The score value of a blank is zero.
* The score for each turn is the sum of the letter values in each word(s) formed or modified on that turn, plus the additional points obtained from placing letters on Premium Squares.
* Premium Letter Squares: A light blue square doubles the score of a letter placed on it; a dark blue square triples the letter score.
* Premium Word Squares: The score for an entire word is doubled when one of its letters is placed on a pink square: it is tripled when one of its letters is placed on a red square. Include premiums for double or triple letter values, if any, before doubling or tripling the word score. If a word is formed that covers two premium word squares, the score is doubled and then re-doubled (4 times the letter count), or tripled and then re-tripled (9 times the letter count). NOTE: the center square is a pink square, which doubles the score for the first word.
* Letter and word premiums count only on the turn in which they are played. On later turns, letters already played on premium squares count at face value.
* When a blank tile is played on a pink or red square, the value of the word is doubled or tripled, even though the blank itself has no score value.
* When two or more words are formed in the same play, each is scored. The common letter is counted (with full premium value, if any) for each word. (See Turns 3, 4 and 5 in the Scoring Examples section.)
* BINGO! If you play seven tiles on a turn, it's a Bingo. You score a premium of 50 points after totaling your score for the turn.
* Un-played Letters: When the game ends, each player's score is reduced by the sum of his or her un-played letters. In addition, if a player has used all of his or her letters, the sum of the other players' un-played letters is added to that player's score.

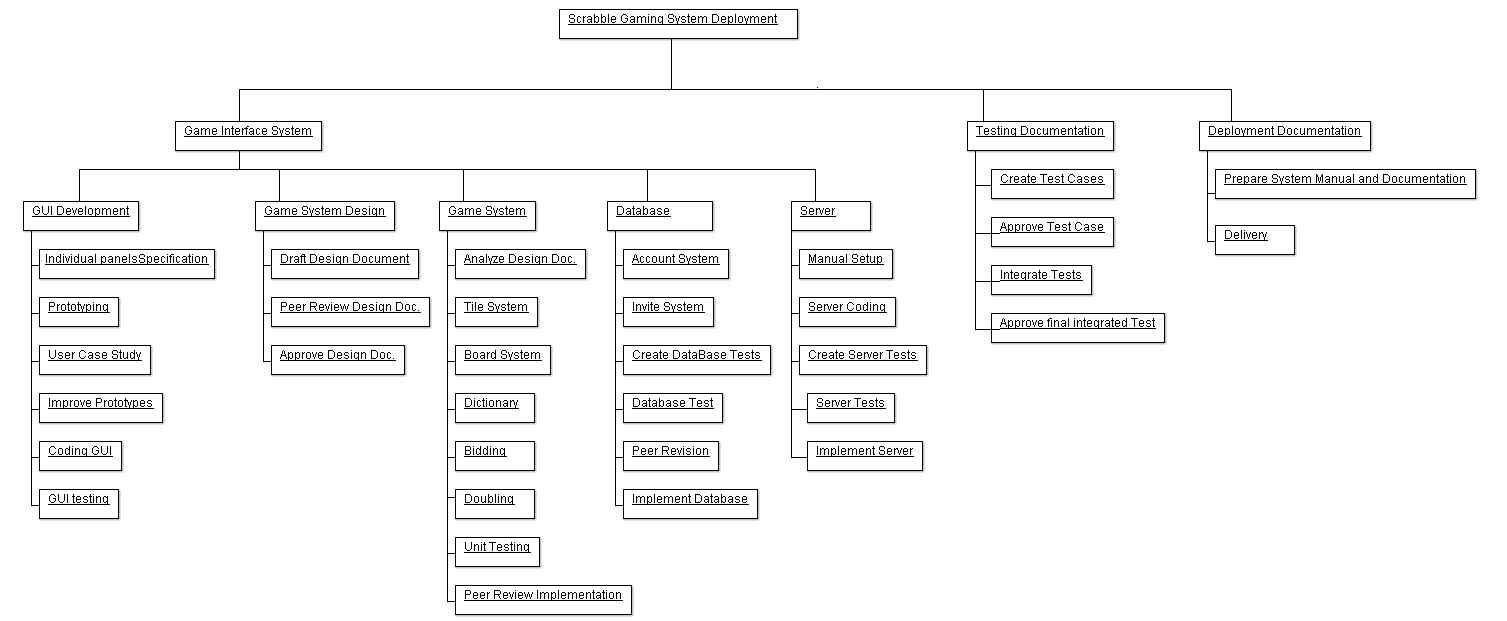
The player with the highest final score wins the game. In case of a tie, the player with the highest score before adding or deducting un-played letters wins. *Hasboro.com ©*

**Plan of Development**

**1. Software Development Process**

The Agile Process of Development will be followed for this project. In particular, we will be using the Trello software application to facilitate our process. Trello allows for our team to designate individual tasks by allowing team members to check out various cards and mark them finished when they have been completed. We plan to communicate with our customer as often as possible after each iteration in the process. All team members have access to the Trello application to allow for daily communication regarding the status of individual assignments (cards), the team’s time management, and the overall status of the project.

2. **Work Breakdown Structure:**



**3. Development Iterations:**

**1. GUI development:**

* Specifying individual panels or “windows” contained within the graphical user interface
* Specifying the contents of each panel: the buttons, interact-able objects, etc, contained within them.
* Prototyping, surveying, and further prototyping. We will create prototypes and send them to the client for feedback, and then make changes based on his reactions to our initial design.
* Coding the GUI: we will implement this interface using the Java GUI framework. We will have either indications or dummy methods associated with the interface.
* Graphical User Interface Testing: creating test cases and expected reactions to the GUI.

**2.** **Game system design**:

* Draft the Design Document: create a document that specifies the class-level implementation and class-level interactions of the Mosaic game system.
* Peer Review Design Document: All related development personal review the design documentation and make sure it accurately and cohesively aligns to Clientele specification.
* Approve Design Documentation: After all suggested changes are made to the design documentation all development person sign off on the final document.

**3. Game System:**

* Analyze Design Documentation: All development personnel review the design document before any implementation occurs. Any last minute design suggestions are to be made at this point; last point before any classes are coded.
* Tile System: All classes and associated methods regarding the Tile system of the Mosaic game system are implemented by the development team. The tile system includes the: interaction of tile with one another, updating of tile formation, and the opponent/player tile interaction.
* Board System: All classes and associated methods regarding the Board system of the Mosaic game system are implemented by the development team. The Board system includes the overall handling of groups of tiles and recognizing of words.
* Dictionary: All classes and associated methods regarding the Dictionary system of the Mosaic game system are implemented by the development team. This Dictionary system allows for the system to recognize English words send by the board system.
* Bidding: All classes and associated methods regarding the Bidding system of the Mosaic game system are implemented by the development team. The Bidding system is a game logic level of the implementation; it is implemented at this point.
* Doubling: All classes and associated methods regarding the doubling system of the Mosaic game system are implemented by the development team. The doubling system is a game logic level of the implementation; it is implemented at this point.
* Unit Testing: Each class and associated methods of the game system will be tested. Testing personnel create test methods and implement them.
* Peer Review Implementation: All functionalities stated above are peer reviewed by development and design personnel to make sure the design was implemented correctly.

**4. Database:**

* Account System: The implementation of this database functionality allows for users to talk to the server and for the server to talk back.
* Invite System: The implementation of this database functionality allows for user to join a game. Development personal will code or add functionalities to the database.
* Create Database Tests: Testing personnel will create test cases and methods to run various database functionalities.
* Database Tests: Testing personnel run designed database tests on the completed database.
* Peer Revision: Testing personnel review the result of all the tests and agree or disagree on implementing the database.
* Implement Database: The completed and peer reviewed database will be joined to the existing game system.

**5. Server:**

* Manual Setup: Any manual setup of the server hardware will be done at this point. Any and all configurations will be set here.
* Server Coding: The functionality that allows the gaming system to interact with the server and for the server to talk back is implemented by development personnel.
* Create Server Tests: Any test cases for the server will be created by the development personnel. These include the design of stress tests and requirement tests.
* Server Tests: Testing personnel at this point implement the designed test cases and requirement tests. Furthermore, any failed tests are reported and any quick fixes are done.
* Implement Server: The server is then joined to the gaming system to allow multiplayer functionalities to the game.

**6. Testing Documentation:**

* Create Test Cases: Testing personnel will create tests cases to test the implementation of the interaction between the game system, database, and server.
* Approve Test Case: Testing personnel and development personnel meet to discuss if created system tests properly covers all requirements.
* Integrate Tests: All tests are implemented for full interactions of all sub-components are peer reviewed.
* Approve Final Integrated Test: Reports are made to developers and testers in cases of test failures. Bug fixes are to be made to ensure proper sub-component interaction before roll-out of system.

**6. Deployment Documentation**

* Prepare System Manual and Documentation: User guides for the interaction of the system for both users and for future developers are to be written.
* Delivery: Final setup of all subcomponents to be rolled out. Handing of the product to the client is to be made at this point. The handing over process takes some time; client developers to peer review completed documents. Making sure the client can properly maintain and upgrade the game system is a priority.

**4. Effort Estimations**

The project participation is divided by iterations, with one team member being the head of that development area, and will be responsible for most of the code, while the other members act as secondary programmers. The division is separated as follows:

**GUI Development:** Armando Benavidez will head the GUI development iteration. He will be mostly responsible for prototyping and coding the GUI. The individual panel specification, user case study, prototype improvement, and GUI testing will be shared amongst all members.

**Game System Desig**n: All participant members will be sharing responsibility for the game system design. This includes drafting, peer-reviewing, and approving the design document.

**Game System Development and Implementation:** Rachel Fraczkowski will head the Game System Development and Implementation Iteration. She will mostly be responsible for most of the coding implementation including the tile, board, bidding, and doubling systems. Alex will concurrently handle dictionary implementation while Armando will handle the unit testing. Analysis of the design document and peer review of the implementation will be shared amongst all team members.

**Database Development and Implementation:** Alex Campbell will head the Database Development and Implementation, specifically the account system and invite system. He will also decide the logistics of hosting the database. Database testing will be created and executed by Armando and Rachel. All team members will use peer revision for the code and assist in the implementation of the database.

**Server Development and Implementation:** (This will be the most difficult task for the entire team) At this time, Alex Campbell has volunteered to head the department, but because of the complex nature of this iteration, the iteration will require a more equal distribution of work amongst all team members. This will consist of designing the server, coding the server, creating tests for the server, and determining manual setup of the server.

**Testing Documentation:** Armando will head the Testing Documentation iteration of the system, specifically creating test cases and iterating through the tests. All members will approve test cases and approve the final integrated test of the system.

**Deployment Documentation:** Rachel will head the Deployment Documentation. This consists of preparing the system manual and documentation. All members will handle the delivery of the system.

**Table of tasks, relationships between tasks, and estimations of effort**

|  |  |  |
| --- | --- | --- |
| Task | Relationship Amongst Tasks | Required Time to Complete (Man Hours) |
| Individual Panels/Specification | Iteration 1: GUI Development | 10 |
| Prototyping | Iteration 1: GUI Development | 20 |
| User-Case Study/ Feedback | Iteration 1: GUI Development | 5 |
| Improve Prototypes | Iteration 1: GUI Development | 10 |
| Coding GUI | Iteration 1: GUI Development | 25 |
| GUI Testing | Iteration 1: GUI Development | 5 |
|  | Total Iteration Estimation: | 75 |
| Draft Design Document | Iteration 2: Game System Design | 20 |
| Peer Review Design Document | Iteration 2: Game System Design | 10 |
| Approve Design Document | Iteration 2: Game System Design | 5 |
|  | Total Iteration Estimation: | 35 |
| Analyze Design Documentation | Iteration 3: Game System Implementation | 5 |
| Tile System | Iteration 3: Game System Implementation | 5 |
| Board System | Iteration 3: Game System Implementation | 10 |
| Dictionary | Iteration 3: Game System Implementation | 10 |
| Bidding System | Iteration 3: Game System Implementation | 15 |
| Doubling System | Iteration 3: Game System Implementation | 10 |
| Unit Testing | Iteration 3: Game System Implementation | 15 |
| Peer Review Implementation | Iteration 3: Game System Implementation | 30 |
|  | Total Iteration Estimation: | 95 |
| Account System | Iteration 4: Database | 20 |
| Invite System | Iteration 4: Database | 15 |
| Create Database | Iteration 4: Database | 10 |
| Database Test | Iteration 4: Database | 10 |
| Peer Revision | Iteration 4: Database | 5 |
| Implement Database | Iteration 4: Database | 10 |
|  | Total Iteration Estimation: | 70 |
| Manual Setup | Iteration 5: Server | 40 |
| Server Coding | Iteration 5: Server | 30 |
| Create Server Tests | Iteration 5: Server | 10 |
| Server Tests | Iteration 5: Server | 10 |
| Implement Server | Iteration 5: Server | 10 |
|  | Total Iteration Estimation: | 100 |
| Create Test Cases | Iteration 6: Testing Documentation | 10 |
| Approve Test Case | Iteration 6: Testing Documentation | 5 |
| Integrate Tests | Iteration 6: Testing Documentation | 15 |
| Approve Final Integrated Test | Iteration 6: Testing Documentation | 5 |
|  | Total Iteration Estimation: | 35 |
| Prepare System Manual and Documentation | Iteration 7: Deployment Documentation | 20 |
| Delivery | Iteration 7: Deployment Documentation | 10 |
|  | Total Iteration Estimation: | 30 |
|  | Total Hours (All Iterations): | 440 |

**Iterables and Dates**

November 1st: First Iteration – GUI Development: Have a working (but not functional) demonstration of the user’s view of the game.

November 15th: 2nd Iteration – Game Development: Have a working AND functional program that demonstrates all aspects of the Mosaic software ASIDE from playing another opponent across a network and being able to access user accounts. The server implementation will not be functional at this time.

December 06th: 3rd Iteration – Completed Application: Have a working and functional program that demonstrates all aspects of the Mosiac software: the user account system, as well as being able to play other people across a network, will now be implemented.