CA314 Assignment 1 - Scrabble Analysis

Group 05

Table of Contents

Refined Requirements Specification

Our implementation of the board game of “Scrabble” called “MyScrabble”, will be closely modelled after the original game. Players will be able to play against one another and against a machine, as such it will take on a client/server structure. In this case each client shows one player’s view of the game while the server processes the game state and synchronises the clients.

It is our goal that our version of the game will include our variation of the rules and movesets in order to attempt to keep the player interested.

Game Overview

Scrabble traditionally consists of a 15x15 square board, where each single square is filled with a letter tile. Typically, there exist special tiles which award you with double or triple scores depending on the word and letters and in the middle of the board consists of the initial starting point where the first letter of the game must be placed in that tile.

At the beginning, each player is given 7 tiles from the letter bag randomly.
Each player is supposed to construct a word from the 7 tiles and attempt to create it on the board.
Game ends when each player has no remaining tiles and the tile bag is empty.
Player with the highest score wins.

As our implementation is web based and is loosely based on the traditional game of Scrabble, as we intend to implement ways to keep the game interesting there will be some additional requirements needed in addition to the base game.

Requirements

- ☐ Users may create an account with a unique username, non unique usernames may be possible by adding a “#” and 3 digits behind it.
- ☐ Minimum number of players in order to start a game is 2
- ☐ Maximum number of players in a game is 4, players may add “bots” to fill in empty spaces.
- ☐ User created instances contain a unique join code, which can be shared.
- ☐ Board size can be customised to be greater than 15x15 in the lobby.
- ☐ Starting letter bag size is 100 tiles, and can be changed in the lobby.
- ☐ Specific game rules can be selected in the lobby, such as specific dictionaries to use.
- ☐ Allows the player to challenge the opponents play, if they believe it’s not in the dictionary, if the word is invalid, it is removed from the board and the player who played it loses a turn, vice versa.

- ☐ First tile placed must be on the centre empty tile, named the “Anchor tile”.
- ☐ Audio cues in order to keep player attention and notify them of important events occuring
- ☐ No tile may be moved or replaced during the game.

- [ ] Clients should send their move information to the server and show what the server validates to be the game state.
- [ ] Server will serve to process the moves made by each player and validate that they are possible, it will also serve to synchronise all the clients.
- [ ] There should exist specific “blank tiles” which allow the player to freely choose what letter the tile is.
- [ ] Game UI should be intuitive and allow the tiles to be played via drag and drop.
- [ ] There should be a leaderboard which shows in real-time what each player’s scores are.
- [ ] Games should be light to run in order to maximise player base ,game instances and efficiency.
- [ ] When an invalid word is placed, a pop up may notify the user that word doesn’t exist in the dictionary.
- [ ] Score calculation occurs at each and every word placed on the board.
- [ ] Turns may be skipped, however if there are 6 (the number may be adjusted in lobby) consecutive skips, the match ends.

Scenarios (User Stories)

Connecting to Website

System: Server is waiting for a user to join
Informal Scenario: User1 accesses website.ie and the home page is displayed with the UI and game options.
Next Scenario: Inputting user-name.

Input user-name

System: Displaying username field and game options.
Current Scenario: User1 chooses the username human66 and clicks the create game button.
Next Scenario: Creating a game.

Creating Game

System: Displaying game creation page.
Informal Scenario: User1 selects private game and is given a code so User2 can join the game directly. User1 selects start game.
Next Scenario: Joining a game.

Joining Game

System: Displaying join game page.
Informal Scenario: User2 enters the code given by User1 and presses join game. User2 is placed in User1's game.
Next Scenario: Taking the first turn.

First Turn

System: Displaying scrabble board with the letter C in the star tile.
Informal Scenario: User1 uses all 7 tiles given at the start of the game for the word CABBAGE with the C covering the star tile. User1 one submits their turn and receives 7 tiles to refill their hand.
Next Scenario: A Standard turn during the game.

Standard Turn

System: Displaying game board from previous turn and a 1 minute timer.
Informal Scenario: User2 drags and places their tiles to create a valid word using a letter from an existing word on the board. A word needs to be from the game dictionary to be valid. User2 is given tiles until they have 7 in their hand. The game calculates User2's score based on how long the word is and which letters were used.
Next Scenario: Submitting an invalid turn**.**

Invalid Move

System: Displaying turn 3 with words CABBAGE and BAT on the board and a 1 minute timer.
Informal Scenario: User1 creates the word TABLE but doesn't connect it to either of the words. User1 submits their turn and receives an error message "Invalid placement" and a try again message.
Next Scenario: Submitting an invalid turn**.**

Invalid Move #2

System: Displaying turn 3 with words CABBAGE and BAT on the board and a 30 second timer.
Informal Scenario: User1 creates the word CULOUR with C connected to the C in CABBAGE. User1 submits their turn and receives an error message "Invalid word" and a try again message.
Next Scenario: Running out of time**.**

Timer reaches 0

System: Displaying turn 3 with words CABBAGE and BAT on the board and a 5 second timer.
Informal Scenario: User1 takes more than 1 minute to submit a valid turn. User1's turn gets skipped automatically.
Next Scenario: Exchanging tiles**.**

Tile Exchange

System: Displaying turn 3 with words CABBAGE and BAT on the board.
Informal Scenario: User1 decides to exchange 2 tiles, "U" and "C". User1 selects exchange tiles and selects the 2 tiles he wants to exchange. User1 selects submit and receives 2 random tiles back and their turn ends.
Next Scenario: Skipping a turn.

Skip Turn

System: Displaying turn 4 with words CABBAGE and BAT on the board.
Informal Scenario: User2 can't think of any valid words so they decide to select skip turn. User2's turn ends.
Next Scenario: Finishing the game.

End Game

System: Displaying rematch button and return button.
Informal Scenario: Both User1 and User2 have run out of tiles or 3 turns have been skipped which ends the game. User1 wins the game as they have a higher score than User2.

Primary class list

ID	NAME
1	CLIENT
2	PLAYER
3	SERVER
4	GAME
5	GAMEBOARD
6	TILE

Class Responsibility Collab

Class Name: Client	**ID:** 1	**Type:** Object
Description: Hosts the scrabble game. The client class translates the users' input into messages which are sent to the server for processing		**Associated Use Cases:** Creating a game, user joining a game, user makes a move, user swap tiles, end game

Responsibilities	Collaborators
connectGame: This will connect the client and player to the game	- Player
sendData: The client sends data from the server to the player and vice versa	- Server
renderGame: This will render the game board and User Interface for the client	
encodeData: Encodes data into a suitable format	
decodeData: Decodes data into a suitable format	

Class Name: Player	**ID:** 2	**Type:** Object
Description: This class is in charge of the moves a player is able to make. This class also holds the users' information like their name and their score.		**Associated Use Cases:** Creating a game, user joining a game, user makes a move, user swap tiles, end game

Responsibilities	Collaborators
getName: Takes the name of the user playing	
getID: Takes the user's unique ID	
getTiles: Gets the tiles for the player	
getScore: Gets the score for the player to view	- Client

swapTiles: Swaps the current tiles for new tiles	
skip: Allows the player to skip their turn	
concedeMatch: Allows the player to forfeit the game	
placeTiles: Allows the player to place a tile	

Class Name: Server	**ID:** 3	**Type:** Object
Description: Handles the interactions between the client and game classes. It retrieves and sends data like the moves a player makes and updates board states.		**Associated Use Cases:** Creating a game,user joining a game, user makes a move, user swap tiles, end game
Responsibilities	**Collaborators**	
encodeData: Encodes the data into a suitable format		
decodeData: Decodes the data into a suitable format		
sendData: The server sends data from the client to the player and vice versa	- Game, Client	

Class Name: Game	**ID:** 4	**Type:** Object
Description: Works closely with the GameBoard class. It is in charge of all logical operations in the program. We can check the validity of a word and a player's move in general. Calculates the score of each player. We keep track of the state of the game and keep the game board updated through this object. Sends this updated state of the game to the server.		**Associated Use Cases:** Creating a game, user makes a move, user swap tiles, end game
Responsibilities	**Collaborators**	

tileChecker: Determines if a moved tile made was valid	- GameBoard
wordChecker: Determines if the tiles placed together form a valid word	- GameBoard
updateGameBoard:	- GameBoard
updateTileSack: Changes the tile sack with new letters after the previous letters have been placed on the gameboard.	- GameBoard
updatePlayer: Updates the player's score in the game	- GameBoard
calculateScore: Takes the sum of the score from the tiles placed	
sendData: Sends data to and from the server	- Server

Class Name: GameBoard	**ID:** 5	**Type:** Object
Description: This class describes the board of the game. It is in charge of the position of tiles on the board and the placement of tile on to said board		**Associated Use Cases:** Creating a game,user joining a game, user makes a move, user swap tiles, end game

Responsibilities	Collaborators
getPos: Gets the position of the tiles on the gameboard	- Tiles
placeTile: Allows the tiles to be positioned on the board	- Tiles - Game

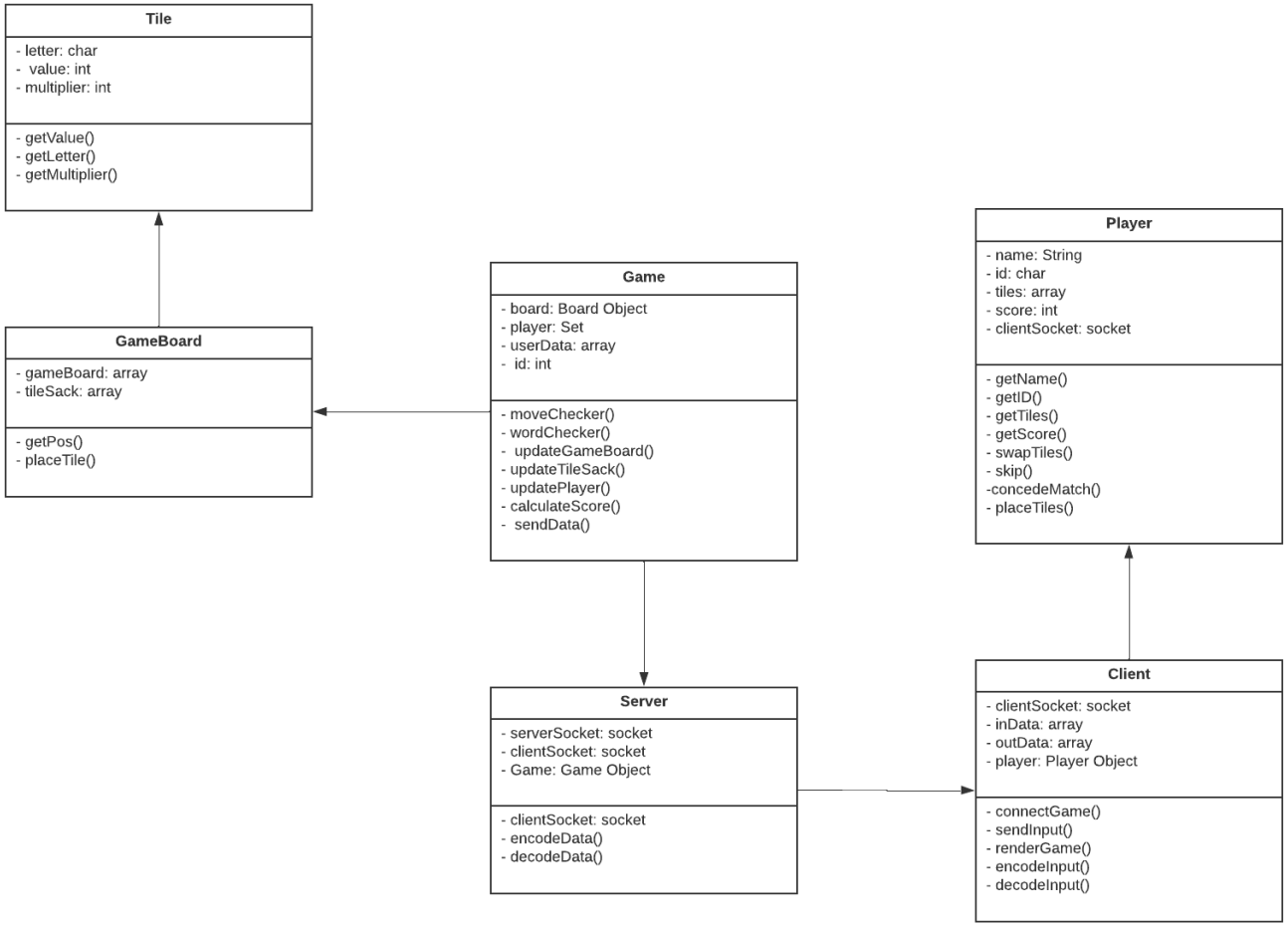
---

Class Name: Tile	**ID:** 6	**Type:** Object

Description: Letters used to make the words in our scrabble game are created using this class. Each tile object has a letter, value, and a multiplier.	**Associated Use Cases:** Creating a game, user joining a game, user makes a move, user swap tiles, end game

Responsibilities	Collaborators
getValue: Finds the value of a tile	
getLetter: Assigns the letter to the tile	

Class Diagram



Use Cases

USE CASE 1	Creating a Game	
Goal in Context	Player creates the games	
Scope & Level	System, Core.	
Preconditions	Player accesses Scrabble on website	
Success End Condition	Scrabble game starts	
Failed End Condition	Scrabble game does not start	
Primary, **Secondary Actors**	Players. Client, Server, Game, Gameboard, Tile	
Trigger	Player presses the ‘Create Game’ prompt on the website	
DESCRIPTION	**Step**	**Action**
	1	Player inputs name into ‘Input User name’ prompt
	2	Player selects the ‘Play Game’ button
	3	Client checks if a name has been inputted before sending a request to the server
	4	Server checks if there is a game available
	5	Server will generate a unique code
	6	Server will then send back that unique code that will be used as the ‘Invite Code’
	7	Client will then create a Scrabble Board
	8	Client will display the unique code to the user that can be used for a second player
	9	Game state will not change until another player has used the invite code to join the game
EXTENSIONS	**Step**	**Branching Action**
	3a	Name not provided: Error message prompt. It will ask to enter a name

VARIATIONS		**Branching Action**

USE CASE 2	User joining a game	
Goal in Context	User joins a game lobby.	
Scope & Level	System, Core	
Preconditions	User created an account.	
Success End Condition	User successfully joins lobby	
Failed End Condition	User fails to join the game lobby.	
Primary, **Secondary Actors**	Players. Game, Client, Server, Gameboard, Tile	
Trigger	Player clicks join game button	
DESCRIPTION	**Step**	**Action**
	1	Player inputs name into ‘Input User name’ prompt
	2	User enters the lobby code in the provided text box.
	3	User presses join game
	4	Client sends lobby code to server.
	5	Server checks if code is valid.
	6	If valid, the server returns data for game state to the client.
	7	Player successfully joins the game lobby.
EXTENSIONS	**Step**	**Branching Action**
	3a	If the player is not logged in, prompt the player to create a username.
	6a	If the lobby code is invalid, return an error prompt.
VARIATIONS		**Branching Action**

USE CASE 3	User makes a move	
Goal in Context	User makes a standard move	
Scope & Level	System, Core	
Preconditions	User is in an ongoing game and it's their turn	
Success End Condition	The User submits a valid move.	
Failed End Condition	The User does not submit a valid move or time runs out	
Primary, **Secondary Actors**	Players. Client. Server. Board. Tile. Game.	
Trigger	Game Started. Other player's turn has ended.	
DESCRIPTION	**Step**	**Action**
	1	Server sends and displays the updated board to Client.
	2	User places their tiles on the board/User skips their turn.
	3	User presses the submit button.
	4	Server sends user board data to Game.
	5	Game checks board configuration and updates player data.
	6	Server sends updated board and player data to Client.
	7	Client displays the updated board to the player.
	8	User's turn ends.
EXTENSIONS	**Step**	**Branching Action**
	4a	User submits an invalid turn: Server asks for board resubmission.
VARIATIONS		**Branching Action**

	1	User skips their turn and their turn ends.
	2	User exchanges tiles and their turn ends.

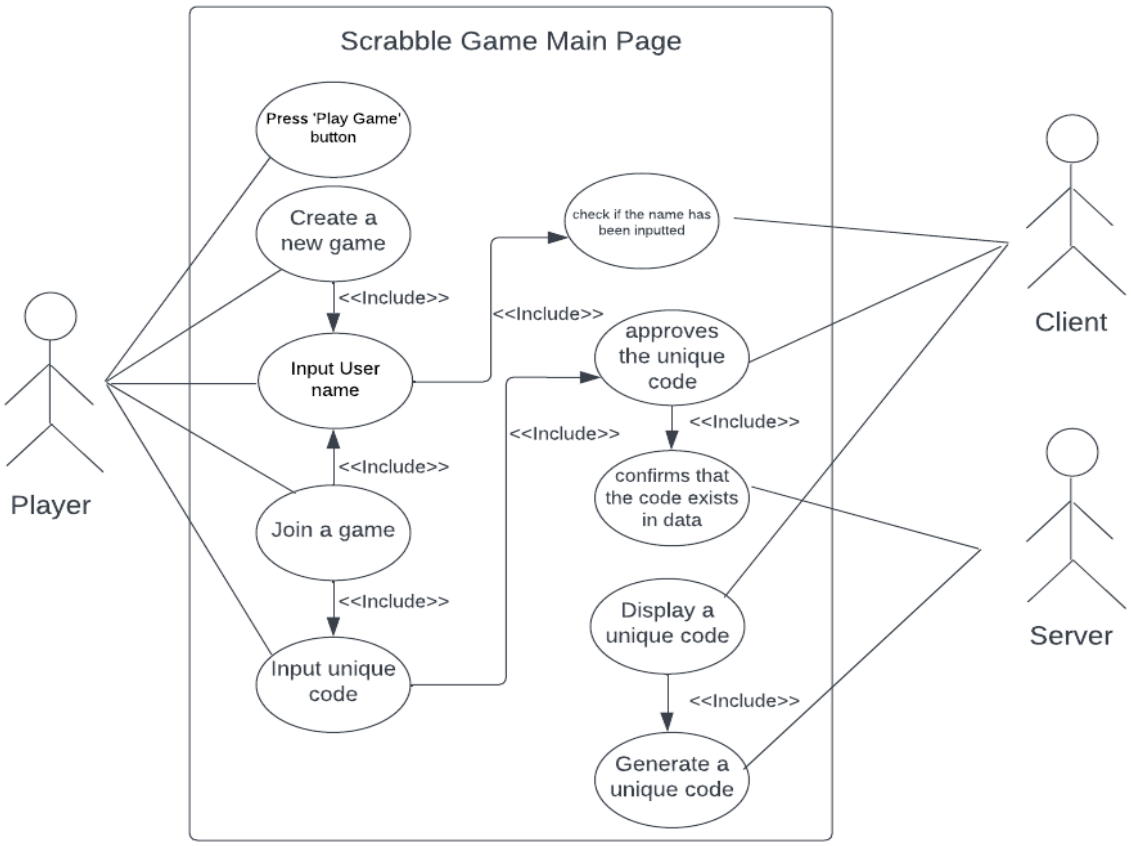
USE CASE 4	User Swaps Tiles	
Goal in Context	Player swaps the tiles in their hand	
Scope & Level	System, Core	
Preconditions	- Game has begun - It is the player's turn	
Success End Condition(s)	- The tiles the player wants to be swapped are swapped.	
Failed End Condition(s)	- Player's tiles requested tiles are not swapped at all - The user's tiles are	
Primary, **Secondary Actors**	.- Player - Client, Tiles, GameBoard, Game	
Trigger	The user presses the swap button.	
DESCRIPTION	**Step**	**Action**
	1	Player presses the "Swap Tiles" button
	2	Client shows the user a "Tile Swap" window
	3	Player chooses the titles that they want to swap
	4	Player presses "swap" button Pop up shows a message saying "Are you sure you want to swap this/these tiles?". The user will have a yes or no option.
	5	Client sends this request to the server (tile swap request)

	6	The tile swap request is passed by the server to the game. Game handles the transaction and updates the player's tiles.
	7	The updated player data tile is sent to the client. The client displays these new updated tiles to the player.
	8	Once the tile(s) is/are swapped, the player's turn is skipped and it is the other player's turn now.
EXTENSIONS	**Step**	**Branching Action**
	4(a)	**-** The number of tiles the user chooses to swap is greater than the number of tiles in the tile sack. Therefore, they can't exchange the tiles
	4(b)	**-** The player presses miss-clicks and selects "yes" instead of pressing "no" when they wanted to cancel the swap.
VARIATIONS		**Branching Action**
	1	No more tiles left in the bag. The player cannot exchange any more tiles.

USE CASE 5	Game Ends
Goal in Context	The game finishes.
Scope & Level	System, Core
Preconditions	Users run out of tiles.
Success End Condition	-The game finishes and ends -There will be a winner
Failed End Condition	The game continues (does not end).
Primary, **Secondary Actors**	Player Client, Server, Game, GameBoard, Tile
Trigger	The user uses all of the tiles in their hand after one turn.

DESCRIPTION	Step	Action
	1	The game will calculate the final score.
	2	The Client will receive the winning data from the Server.
	3	The Client will show the winning information to the user.
EXTENSIONS	**Step**	**Branching Action**
		N/A
VARIATIONS		**Branching Action**
	1	The player clicks the forfeit button
	2	There are six consecutive skips (three for each user)

Use Case Diagram



Results of Structured Walkthrough

Step	Player	Use Cases	Use Case ID
0	Player 1	Create Game	1
0	Player 2	Join Game	2
1	Player 1	Makes Move	3
2	Player 2	Makes Move	3
3	Player 1	Swaps Tiles	4
4	Player 2	Swaps Tiles	4
7	Player 1	Skips Move	3
8	Player 2	Skip Move	3
9	Player 1	Makes Move	3
10	Player 2	Forfeits Game	5
11	Server	Game Ends	5

Minutes/notes of team meetings

1ST MEETING

Minutes for 19/09/2022, Monday 1:00pm

Members Present:

- Kline
- Effa
- Liam
- Mustafa

Members Absent:

- Ronghui, Covid
- Matt, doctor's appointment
- Vaidas, in Germany

What did we do/discuss?

- Decided that we're meeting at 1-2pm or 3-4pm every week, in L128 or L129
- Rotating minute taker, Mustafa will be taking minutes next
- Scrum meeting
- Google Calendar reminder every week, say if you're going or not
- If you can't go in person, try go on Discord

What are we going to do?

- Research what we're going to do on Scrabble. Check GitHub or YouTube for any guides
- Check Discord for any updates

Any difficulties?

- No

2ND MEETING

Minutes for 27/09/2022, Tuesday 12:00pm

Members Present:

- Kline
- Mustafa
- Matt
- Ronghui
- Vaidas

Members Absent:

- Effa
- Liam

What did we do/discuss?

- Progress done so far by our members
- Editing the document
- Rotating minute taker, Matt will be taking minutes next
- Scrum meeting
- Google Calendar reminder every week, say if you're going or not
- If you can't go in person, try go on Discord

What are we going to do?

- Finish what is necessary for the analysis such as CRC, Use cases and structured walkthrough.
- Check Discord for any updates
- When you get something done, say it in the Discord.

Any difficulties?

- No

3RD MEETING

Minutes for 04/10/2022, Tuesday 12:00pm

Members Present:

- Kline
- Mustafa
- Matt
- Ronghui
- Vaidas
- Effa
- Liam

Members Absent:

- N/A

What did we do/discuss?

- Progress done so far by our members and what has to be completed
- Use Cases
- Editing the document
- Rotating minute taker, Liam will be taking minutes next
- Scrum meeting
- Google Calendar reminder every week, say if you're going or not
- If you can't go in person, try to go on Discord

What are we going to do?

- Complete Use Cases and Result of Structured Walkthrough before Friday October 7th.
- Have one Use Case done by each person in the group
- Draw up a Use Case Diagram
- Check Discord for any updates
- When you get something done, say it in the Discord.

Any difficulties?

- No