

CA314 - Analysis

MY SCRABBLE

Student Name	Student Number
Kevin Cogan	18421694
Alex O'Neil	18414882
Cameron Fitzpatrick	18493006
Niall Dagg	17311161
Conor Mckeon	17386616
Jeff Thomas	18105319

Table Of Contents

Table of Contents

1. Introduction
 - 1.1. Purpose
 - 1.2. Intended Audience & Uses
 - 1.3. Scope
 - 1.4. Definitions and Acronyms
2. Overall Project Description
 - 2.1. Product Perspective
 - 2.2. Product Constraints & Dependencies
 - 2.3. User Characteristics
3. System Features and Requirements
 - 3.1. Functional Requirements
 - 3.1.1. Product Features
 - 3.2. Non-functional Requirements
 - 3.3. User Interface
 - 3.4. Hardware Interface
 - 3.5. Communication Interface
4. Scenarios
 - 4.1. User sets up a local game with 4 player
 - 4.2. A new user sets up an online game
 - 4.3. User plays the word “face”
 - 4.4. User exchanges 3 of their rack tiles
5. Primary List & CRC Cards
6. Class Diagram
7. Use Case Diagram
8. Use Cases
9. Structured Walkthrough
 - 9.1. Playing MyScrabble Online
 - 9.2. Playing MyScrabble locally
10. Group Minutes and Agenda

Refined Requirements Specification

1. Introduction

1.1 Purpose

This document outlines the requirements specification for MyScrabble, an internet-based strategy game. We were tasked to create the game for our CA314 OO Analysis and Design module.

1.2 Intended Audience & Uses

The intended audience for this document is:

- Product Users: People who play the game, and who would like to know more about the process behind creating it, and how we implemented each feature.
- Product Testers: Testers who comb through our game looking for bugs and glitches can use this document to more easily search for potential problem areas in our software.
- Product Devs: Product Devs can use this document to grasp how we went about creating this game, the functions and classes we used, and our goals in order to make better decisions regarding game alteration.

1.3 Scope

We have been tasked with creating an internet-based strategy game called “MyScrabble” which takes inspiration from the classic board game. Players must be able to play against each other on a local connection, and also on a remote connection over the internet. The application must use a client/server structure, with each player seeing their own individual view of the game while the server carries out events and communicates these events to the players. The main goal of our take on this classic game is to include rules and events that keep the player interested.

1.4 Definitions and Acronyms

<u>Term</u>	<u>Definition</u>
System	The program we are developing
Board	The 15x15 board on which the game is played.
Tile	Playable lettered squares which give the player points when played.
Tile Bag	A bag containing all not-in-play tiles. Players draw from the bag when their rack contains <7 tiles.
Rack	The shelf in which a player's currently usable tiles reside (must always contain 7 tiles unless the bag is empty.)
Blanks	Tiles that may be used as any letter in the alphabet (stated by the player).
Premium Squares	Squares located on the board that grant extra points when tiles are placed on them.
Turn	A player plays a few tiles, then takes tiles from the letter bag until their rack is full.
BINGO!	A score premium of 50 points that occurs when 7 tiles are played on 1 turn.



2. Overall Project Description

2.1. Product Perspective

The product we are creating is called MyScrabble, a game derived from the classic board game Scrabble. There are very many products in a similar vein on the internet, although we hope to make ours stand out by implementing new features to keep the user hooked in, such as customizable rules to keep the game interesting for players. Another attraction is that we record players' results such as wins and losses so we can match them with users of the same skill set in scrabble creating a fairer and more enjoyable gaming experience.

2.2. Product Constraints & Dependencies

Experience: Our product will be constrained due to the fact that our team's experience is primarily in Python programming language, although we do hope depending on time constraints to implement the Flask framework as it is compatible with Python.

Time: Our product will be constrained by the amount of time each of our team members has available. We must strike the correct balance between working on this project and our other college work.

Deliverables: The brief given to us for this project outlines strict due dates for project deliverables such as our Analysis and Product Design documents. This will require us to have frequent team meetings and assigned tasks for each team member.

2.3. User Characteristics

There are not many user characteristics needed for a concept such as this, other than a basic understanding of the game's rules and elements.

3. System Features & Requirements

3.1. Functional Requirements

3.1.1. Product Features

- Move: The system should be able to initiate a player move, allowing them to choose which letters to play and in what direction and area to play them in.
- Board Creation: The system must be able to generate a blank 15x15 playing board on start-up of a game, inserting correct premium squares.
- Awarding Points: System must be able to calculate and award correct points to players after a move.
- Rack & Bag capacity: System should always be checking that all racks have 7 tiles and should prompt the player to collect more unless the tile bag is empty.
- Word Verification: System should be able to verify using a dictionary that a word being played is in fact a valid word.
- Profile Creation: A log-in system should be used in order to create individual player profiles which can track statistics and total scores.

3.2. Non-functional Requirements

- Performance: The system must be able to run in a stable manner, with minimal lag/wait times. Depending on the user's internet speed, updates to the game board should take no longer than 3-4 seconds to appear after a turn timer has run out.
- Quality: The system must look polished and put together and should not have any off-putting or confusing interfaces.
- Usability: The system's UI should be clear, concise and easy to use for anyone who wishes to use it.
- Security: Users must be prompted to create a profile with login details if they wish to play online games. This profile will be password protected and will have a username that is displayed in-game.
- Reliability: The system should reliably connect our user to a game server once one is available, and should update the user's client with new game occurrences consistently and quickly.
- Maintainability: The system should be programmed in a way where code can be read through and followed without difficulty, thus making any potential need for bug fixes/system changes quick and efficient.

3.3. User Interface

The user should be met with a homepage and presented with two options that allow them to either choose to play locally or remotely. Should the player choose remote, they continue onto another menu where they can select which remote game they want to join. Should they choose local, they are prompted to input the number of players, select names for their players and assign player colours. The user can then click the start game button.

Once the game has begun, the interface should clearly display the 15x15 game board, the current scores of the players, the current rack contents of the players, a text box in which the player can enter the word they want to play and a method for selecting where that word is played on the board.



3.4. *Hardware Interface*

A personal computer using basic peripherals such as screen, mouse and keyboard is needed to run the interface for our system. The game is not very demanding, however, so hardware requirements are low. Should the user wish to play games online, a connection to the internet will be needed either via ethernet or modem.

3.5. *Communication Interface*

This system will use Wi-Fi to communicate with the game's server in order to participate in online games. Should the user not have access to a Wi-Fi connection, they will be limited to playing local offline games without the ability to log in to their account. Offline local games will be hosted on the device that is running the system.

Scenarios

User sets up a local game with 4 players

On the homepage, the user clicks on the “Play locally” button. The user is prompted to enter the number of players they wish to play with and is guided to a screen with 4 text boxes in which they enter the names “Cameron”, “Kevin”, “Conor” and “Jeff” in order to assign names to the players. Users are also prompted to assign colours to each player for identification purposes. User clicks on the “Start game” button, and the game begins.



A new user sets up an online game

From the homepage, the user chooses “Play Online”. The user is prompted to log in using their unique credentials or to create an account. The player is assigned automatically to an available server that has the same language preference and skill set of the player. One by one, other users begin to join the game. Once the server is full the game will begin.

User plays the word “face”

The user’s turn has begun. The user’s rack contains the letters [a, c, w, f, g, e, k]. The user types “face” into the word submission box and selects where they would like to play the word by inputting coordinates. The player enters the word they would like to play, and then for each letter of that word they must input an x-y coordinate in order to place it on the game board. The user has not played their word on any premium tiles, and so is not awarded bonus points. The scores of the letters in their chosen word “face” are tallied up for a total of 9 points. These 9 points are added to the user’s score, and the scoreboard is updated. The user has 4 new tiles added to their rack, and the capacity of the tile bag is reduced by 4.

User exchanges 3 of their rack tiles

During their turn, the user is unhappy with their tiles and so chooses to “Exchange” some of their tiles. The user is prompted to type the tiles they wish to exchange, and the user types in each of the first three tiles in their rack. The user confirms that they would like to exchange the tiles, and the chosen tiles are returned to the tile bag while 3 random new tiles are taken out and placed on the user’s rack.

Primary List & CRC Cards

Class Name: New Game	ID: 1	Type
Description	Associated Use Cases	
Acts as a “main menu” for the game. Allows players to choose between an online and local game, choose certain rule changes, let players log in and start the game	Choose game mode Register Account Login Contact admin	
Attributes	Methods	
Local - boolean value Number_of_players - integer	register_details() login_entry() rule_changes() contact_admin()	
Responsibilities	Collaborators	
Choose online or offline		
Player login	Player	
Choose game rules	Rules	
Number of players		

Class Name: Player	ID: 2	Type
Description	Associated Use Cases	
Player class keeps all details on the player. Such as Players name, favourite colour, log in details and win/loss record	Assign Player Order	
Attributes	Methods	
Name - string Record - two integers Login_details - dictionary Colour - string User_language - string Current_turn - integer	login_verify()	
Responsibilities	Collaborators	
Login Details	New Game	
Win/Loss Record		

Class Name: Board	ID: 3	Type
Description	Associated Use Cases	
This class is to initialise the board for the game	Display Board	
Attributes	Methods	
Board_size - integer Premium_squares - list	create_board()	
Responsibilities	Collaborators	
Check Board Size	New Game	
Creates Board		

Class Name: Rules	ID: 4	Type
Description	Associated Use Cases	
The Rules class will hold certain attributes (rules) that we will allow the player to change when they start their game.		
Attributes	Methods	
Board_size - integer Turn_time_limit - integer Game_time_limit - integer Rack_size - integer		
Responsibilities	Collaborators	
Check Board Size	New Game	
Checks time limit		
Checks rack size		

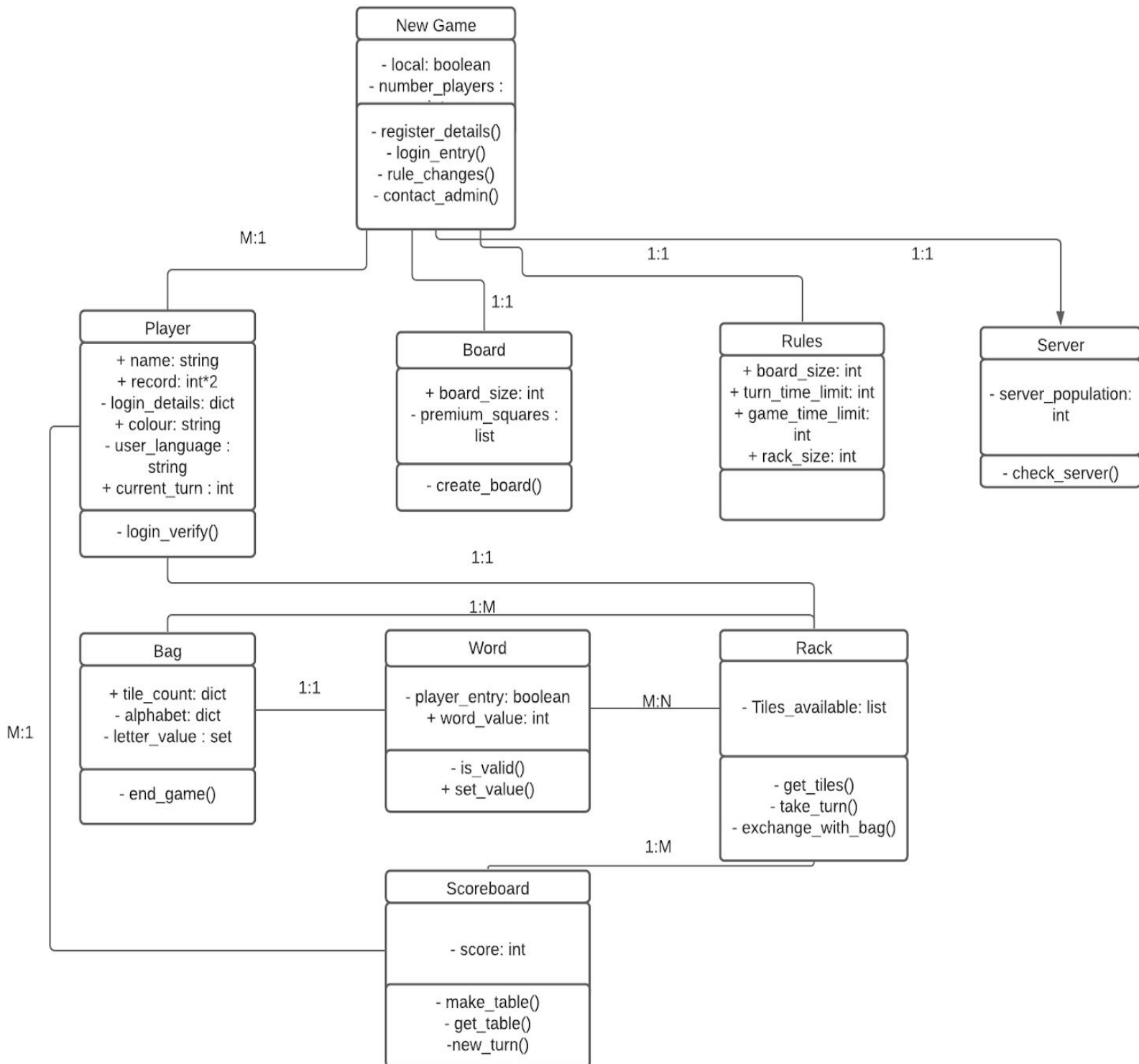
Class Name: Bag	ID: 5	Type
Description	Associated Use Cases	
The Bag class will keep track of how many tiles are left in the bag and the value of each of those tiles. When the bag runs out of tiles the game will end	Exchange Tile Distribute Random Letters to Users	
Attributes	Methods	
Tile_count - integer Alphabet - dictionary Letter_value - set	end_game()	
Responsibilities	Collaborators	
Tiles remaining	Rack	
Holds value of every letter	Word	

Class Name: Word	ID: 6	Type
Description	Associated Use Cases	
The Word class will be used to verify a players entry onto the board is a real word. It will also then calculate the value of the word that has been played	Enter Word	
Attributes	Methods	
Player_entry - Boolean Word_value - int	is_valid() set_value()	
Responsibilities	Collaborators	
Checks if the word is valid		
Calculates the value of the word	Bag	

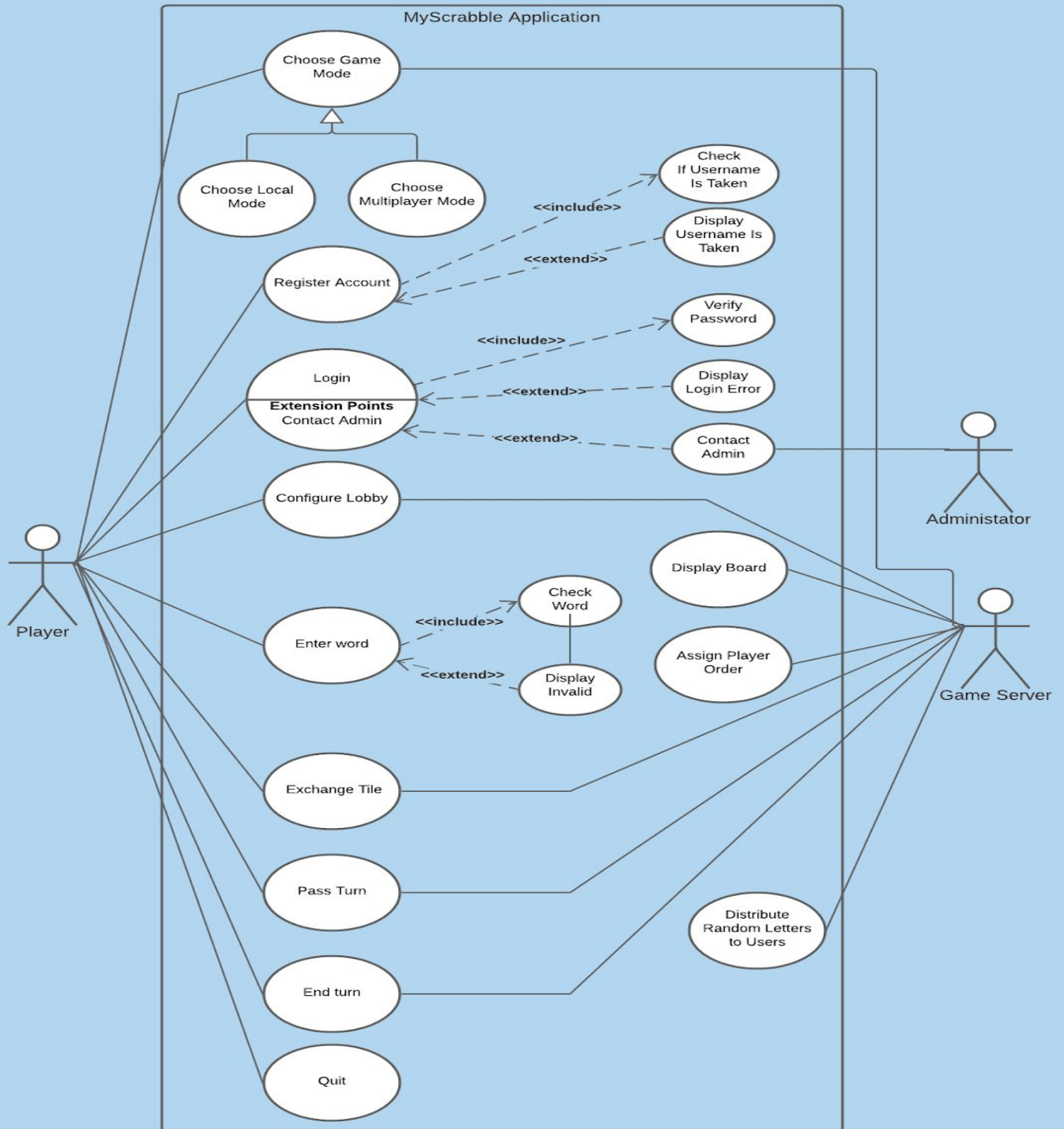
Class Name: Rack	ID: 7	Type
Description	Associated Use Cases	
The Word class will be used to verify a players entry onto the board is a real word. It will also then calculate the value of the word that has been played	Enter Word Exchange Tile Pass Turn End Turn	
Attributes	Methods	
Tiles_available - integer	get_tiles() take_turn() exchange_with_bag()	
Responsibilities	Collaborators	
Presents letters to player	Bag	
Allows the player to play word	Board, Word	

Class Name: Scoreboard	ID: 8	Type
Description	Associated Use Cases	
The Scoreboard will track the player's score through-out the game and display them.		
Attributes	Methods	
Score - integer	make_table() get_value() new_turn()	
Responsibilities	Collaborators	
Keeps track of every player's score	Player	
Displays everyone's score	Word	

Class Diagram



Use Case Diagram



Use Cases

USE CASE 1	Local Multiplayer Game	
Goal in context	Multiple players play the game on a local machine	
Scope	Core and System	
Preconditions	All players are sitting to play on the same machine and have a MyScrabble instance opened	
Success End Condition	Minimum two players make a move in MyScrabble match	
Failed End Condition	Two players are not able to input a move in a Scrabble game	
Primary	Player	
Secondary	Game System	
Trigger	The player presses the local play option	
DESCRIPTION	Step	Action
	1	User presses the local play option on the home screen
	2	The system presents a screen to input four names
	3	Player inputs the player names and presses ok

	4	System logs the information provided
	5	The system calculates players based on the number of names given
	6	The system displays board and highlights the name of the current player
	7	Player inputs a word
	8	System logs the word
	9	The system alters the game instance based on the previous move and highlights the next current player
	10	Second player inputs a word
EXTENSION	Step	Branching Action
	3a	Player inputs no names and presses ok
	5a	The system finds the no. of names to be zero
	5b	The system reports a lack of sufficient data
	5c	The system presents a screen to input four names (Step 2)
	7a	A player is given an option and selects pass: Next player is selected

	7b	<p>A player is given option and selects exchange a tile:</p> <p>7b.1)The player selects tiles to replace</p> <p>7b.2) System replaces selected tiles with random letters</p>
	8a	The system detects the logged move to be invalid
	8b	The system finds an invalid move and highlights the same player
VARIATION		Branching Action
	1	Users select play local option on a different variety of device or browser

USE CASE 2	Playing MyScrabble Online - Login	
Goal in Context	User logs into the game remotely online.	
Scope	Functional requirements of a system	
Scope & Level	System, Core	
Success End Condition	The user plays a MyScrabble game.	
Failed End Condition	User struggles to login and leaves the login page.	
Primary, Secondary Actors	Player. Online server system.	
Trigger	The player selects to play remotely on the home page.	
DESCRIPTION	Step	Action
	1	The login page is displayed
	2	User types in details on the login screen
	3	The server checks user login details
	4	The server checks IP Address
	5	Player connects to a server with the same language users, based off IP address proximity.

	6	The player starts playing MyScrabble
EXTENSIONS	Step	Branching Action
	3a	Invalid username or password: Display invalid login details
	4a	No same language server available: Wait 10 seconds and try to connect again
VARIATIONS		Branching Action
	1	The player accesses game on the different web browser
	2	Players access game with IP addresses from different countries.



USE CASE 3	Playing MyScrabble Online - Contacting Administrator	
Goal in Context	User contacts the administrator	
Scope & Level	System, Core	
Preconditions	User must have a PC with an internet browser and an internet connection.	
Success End Condition	User contacts the administrator	
Failed End Condition	User struggles to contact the administrator and leaves the site.	
Primary, Secondary Actors	Player. Administrator.	
Trigger	The player selects play remotely on the home page and is on the login page for MyScrabble.	
DESCRIPTION	Step	Action
	1	The player selects the contact administrator button
	2	The default email account is opened with administrator email in the email address placeholder.
	3	The player sends an email.
	4	The user gets an acknowledgement email.
EXTENSIONS	Step	Branching Action



VARIATIONS		Branching Action
	1	The player accesses game on the different web browser
	2	Player default email providers

USE CASE 4	MyScrabble Online Registration
Goal in Context	The player registers to play a game of MyScrabble online
Scope & Level	System, Core
Preconditions	The player must have a MyScrabble instance open to the home page, on a working computer
Success End Condition	Player successfully registers online for MyScrabble
Failed End Condition	Player are unable to register due to an error
Primary, Secondary Actors	Player. Online server system.
Trigger	The player selects play remotely on the home page

DESCRIPTION	Step	Action
	1	The player selects the remote play option on the home screen
	2	The system presents the user with Login Page
	3	User click the register option on the login page
	4	The system presents the user with Registration Page
	5	The user inputs the fields of username, email and password
	6	The system validates information and checks if the username is available
	7	The system shows a registration confirmation
EXTENSIONS	Step	Branching Action
	3a	3a.1) User inputs login values and submits the login form 3a.2) System raises a login error 3a.3) Player is sent back to Step 2
	5a	User updates one of the fields incorrectly
	5b	User leaves a field blank
	6a	The system tells the player the username is taken



	6b	The system tells the player the password is invalid
	6c	The system tells the player the email account already has an associated account
	7b	The system shows registration error
VARIATIONS		Branching Action
	1	The user selects the option on a mobile device, tablet, smart device using a different browser

USE CASE 5	Playing MyScrabble Online
Goal in Context	The user plays the game remotely online.
Scope & Level	System, Core
Preconditions	User must have a PC with an internet browser, an internet connection and valid login details.
Success End Condition	Player successfully complete a turn of online MyScrabble
Failed End Condition	Player are unable to play a game of MyScrabble due to an error
Primary, Secondary Actors	Player. Online server system.
Trigger	The player selects play remotely on the home page and successfully login into their account



DESCRIPTION	Step	Action
	1	Player's turn is assigned randomly
	2	The empty board is displayed to all players
	3	14 Random letters are distributed to each player
	4	Player enters word
	5	Word is checked by the system
	6	Word is displayed on the board by the system
	7	The turn is completed and the system moves to the next player
EXTENSIONS	Step	Branching Action
	1a	A player selects a tile from the bag to determine what turn they go.
	5a	Word is valid
	5b	Word is invalid: 6b.1) Display the invalid word 6b.2) Return to step 5



	4a	A player is given an option and selects pass: Next player is selected
	4b	A player is given option and selects exchange a tile: 4b.1)The player selects tiles to replace 4b.2) Random tiles replace selected tiles 4b.3) Proceed to step 5
VARIATIONS		Branching Action
	1	The player accesses game on a different web browser


A Structured Walkthrough

Playing MyScrabble Online

Once the player accesses MyScrabble through a web browser, the player will be presented with a choice to play online or locally on the home page. If you choose to play online, you will be requested to log in with your unique credentials by `Login_entry()` or if the user is new to MyScrabble they can register with `Register_details()`. This login system enables us to create a profile for each player allows us to easily assign names to users, identify their native language via IP address location, and track statistics and scores for each game. This information can be used to try to maximise the length of time they spend playing MyScrabble. Furthermore, if the player is having issues/queries logging in there will be a `Contact_admin()` section on the login page enabling players to email the administrator.

After the user successfully logs into their account they will be connected to a server based on the `users_language` in the `Player` class. If no server is currently available the program will use `check_server()` which waits for ten seconds before checking for a free server again of the same language preference. Once entered into a server each player's name is retrieved from the `Player` class in `name`, a random colour and turn number, such as the first turn, is given to each player. The `Rules` class will have a fixed `board_size` of 15 squares by 15 squares, `turn_time_limit` of forty seconds, and `rack_size` of seven tiles. Once all initialised the `Start_game()` will be called to begin the game.

Once the game begins, the tiles are taken in with their respective scores from a separate file, and then added to the bag which has a set number of tiles controlled by `tile_count`. Then each player's racks receive seven random letters. The bag's total tiles count is reduced and the first player can begin their turn. Once the player's turn begins a count-down timer starts and is displayed to all players. The time limit is retrieved from the `time_limit` attribute in the `Rules` class. The timer ensures there are no long waiting times for all the players. The player is first given a choice to play, pass or exchange a tile. If the player decides to pass the player's turn will end and then the next player's turn will begin. If the player decides to exchange tiles the user selects each tile they wish to exchange. The selected tiles are returned to the bag and replaced with random tiles from the bag. If the player selects play they input a word chosen from `Tiles_available` in the `Rack` class. The player types a letter they would like to place on the board from the letters in the rack, indicating the position on the board by using a grid system that has the alphabet on the y-axis and numbers on the x-axis. This is repeated until all letters in the word are accounted for. The word is then checked to see if it is in the dictionary using `is_valid()`. If the inputted word is invalid the user will be asked to input another combination of letters again. Else, if the word is valid the points are then added based on the positions of the letters on the board which may include point multipliers, using `set_value()` and then added to the player's total score variable.




Throughout the game, the scores of each player are kept within the `Scoreboard` class and displayed by the `get_table()` function. The player's rack is then repopulated with randomly selected new tiles, which has been once again subtracted from the total number of tiles in the bag. Once this has been completed, player 1's turn is now over and it is moved across to the next player. The game continued between the players until either the time limit of fifty-two minutes has been reached, or the tile bag has been emptied where `Tile_count` is equal to zero. If the tile bag has become empty, one final turn will commence. Upon the completion of this turn, the game has been completed. The win or loss is recorded in the `record` found in the `Player` class. The players are then asked if they would like to play again and if there are any rules they would like to change. If they do, the game then repeats itself. If not, the game is over and the program is now terminated.

Playing MyScrabble Locally

When you choose offline, you are then prompted to type in the player names using `Player_name()` and also `Player_colour()`. Once that has been completed, you then have the ability to change various rules using the `Rules` class to create a unique experience that can be changed after every game. These rules include the ability to choose the game time length with `game_time_length()`, the size of the board using `Board_size` and the rack size each player has with `Rack_size`. When the rules have been decided and chosen, the game can begin which is controlled by `Start_game()`.

Once the game begins, the tiles are generated with their respective scores from the Alphabet dictionary, and then added to the bag which has a set number of tiles controlled by `Tile_count()`. Each player's racks receive seven random letters. The bag's total tiles count is reduced and the first player can begin their turn. Once the player's turn begins a count-down timer starts and is displayed to all players. The time limit is retrieved from the `time_limit` function in the `Rules` class. The timer ensures there are no long waiting times for all the players. The player is first given a choice to play, pass or exchange a tile. If the player decides to pass the player's turn will end and then the next player's turn will begin. If the player decides to exchange tiles the user selects each tile they wish to exchange. The selected tiles are returned to the bag and replaced with random tiles from the bag. If the player selects play they input a word chosen from `Tiles_available()` in the `Rack` class. The player types a letter they would like to place on the board from the letters in the rack, indicating the position on the board by using a grid system that has the alphabet on the y-axis and numbers on the x-axis. This is repeated until all letters in the word are accounted for. The word is then checked to see if it is in the dictionary using `is_valid()`. If the inputted word is invalid the user will be asked to input another combination of letters again. Else, if the word is valid the points are then added based on the positions of the letters on the board which may include point multipliers, using `Word_value` and then added to the player's total score variable.

Throughout the game, the scores of each player are kept within the `Scoreboard` class and displayed by the `Table()` function. The player's rack is then repopulated with randomly selected



new tiles, which has been once again subtracted from the total number of tiles in the bag. Once this has been completed, player 1's turn is now over and it is moved across to the next player. The game continued between the players until either the time limit of fifty-two minutes has been reached, or the tile bag has been emptied where **Tile_count** is equal to zero.

If the tile bag has become empty, one final turn will commence. Upon the completion of this turn, the game has been completed. The players' total scores are weighed against each other using **Total_per_person** and a winner can be announced. The players are then asked if they would like to play again and if there are any rules they would like to change. If they do, the game then repeats itself. If not, the game is over and the program is now terminated.

Group Minutes And Agenda

Meeting 1: Meeting new people

Date: 9/10/2020

- Discussed the actual game of scrabble.
- Familiarised ourselves with the rules.
- Read over the project specification.
- Decided on a language being python.
- Set the next meeting for the 12/10.

Meeting 2: Getting down to business

Date: 12/10/2020

- Discussed project specification in detail
- Started to assign roles and deadlines by use of a product backlog where no one was asked to do a task team members volunteered to do the task.
- Began to really bond as a team and understand our skills better.
- Set the next meeting for the 16/10.

Meeting 3: All coming together now

Date: 16/10/2020

- Reviewing the work we have done: We discussed our difficulties with aspects of the tasks that we had to complete as well as updated the team with current progress.
- We updated the team on how much time is needed to complete the task.
- Clearing up any issues that have arisen during our various tasks.
- Sorting out the minor details within our game.
- Set the next meeting for 19/10.



Meeting 4: Smooth Sailing

Date: 19/10/2020

- Everyone has finished their first draft of their tasks and we decided to refine the topics as a group to ensure that all the tasks were in unison.
- All finished task was taken off the product backlog.
- Applying the finishing touches on what has been completed thus far.
- Discussed the next stage of deliverables to keep it in our mind.

Meeting 5: Final Touches

Date: 21/10/2020

- We have completed our final draft.
- Checking on feedback and answering questions we had on each other's various tasks.
- Planning our next tasks as we are now a week ahead.
- Discussed a framework for our scrabble program.
- Set a date for our final deliverable 1 meeting.

Meeting 6: The Submission

Date: 28/10/2020

- Added all our work into one folder ready for submission
- Discussed the next deliverable.