CA314 Scrabble - Analysis

Gareth Hogan 20379616 *Detutu Adebisi 20330263*
Rosa Duessmann 20495376 *Peadar McHenry 20490894*
Eimear McKevitt 20500199 *Paul McKee 20414956*

Contents:

- Work Distribution
- Introduction
- Agile Method
- Refined Requirements Specification
- Scenarios
- Primary Class List
- CRC Cards
- Class Diagram
- Use Case Descriptions
- Structured Walk Through
- Team Minutes
- Conclusion

Work Distribution:

Paul - Refined Requirements Spec, CRC Cards
Peadar - Refined Requirements Spec, CRC Cards
Eimear - Class List, Class Diagram, Report
Rosa - User Scenarios, Use Case Descriptions, Team Minutes
Detutu - User Scenarios, Use Case Descriptions
Gareth - Class List, Class Diagram, CRC Cards, Report

Introduction:

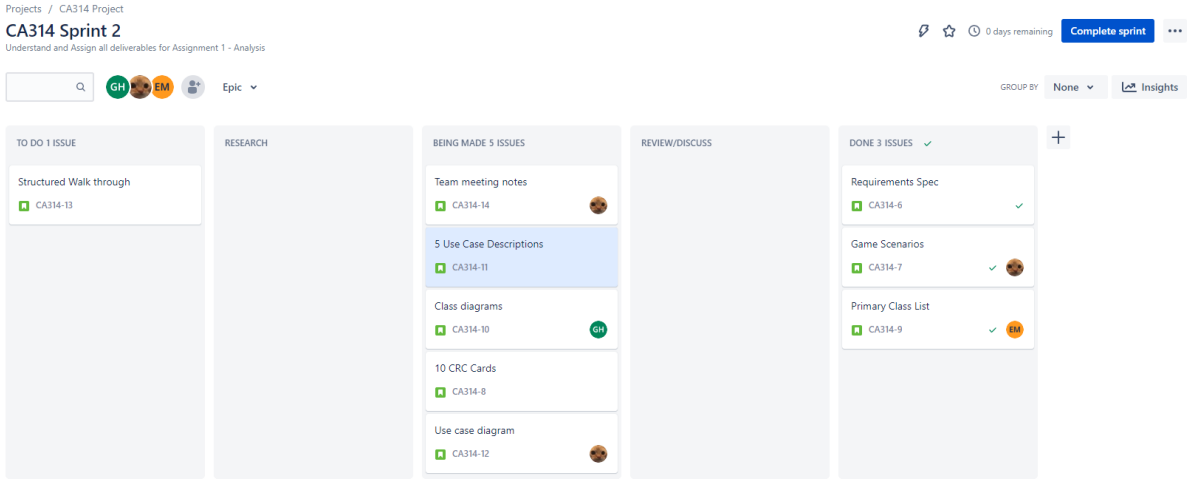
In this first stage of the project we were tasked with the initial analysis of our given game, Scrabble, and then the design and framework of how we are going to plan out and approach the development of our physical implementation.

In this submission you will find the basic overview and structure of our game, its classes, methods, structure and how we envision an eventual game being played using our design.

Agile Method:

During this project we are challenged to develop Scrabble as a game whilst also following an agile development process. In this first stretch of analysis and design we found that it worked quite well, we wanted to schedule frequent meetings to keep up with individuals and their progress, but we also made sure each meeting was brief and informal, no work was done during them.

We also trialled out using an agile tool to keep track of individual progress and workload while apart using Jira, which one of our team members had used over the summer in an internship. You can see an example of the board and tickets we kept in the image below. But ultimately we decided to deprioritise using it, it was a bit cumbersome to keep up to date and use for us. We are going to keep the board available and may update it as we move forward with the project.



Refined Requirements Specification:

Scrabble is a classic board game about forming words from a random set of 7 letters that each player has, the more complex the word and the rarer the letters used the more points they get. The basic premise of scrabble can be summarised from the rules of the board game version, described below how we know them.

- Scrabble is a board and tile game which can have 2-4 players.
- Players compete by spelling words with lettered tiles on the board which is made up of 225 squares.
- Each letter of the alphabet holds a specific amount of points and you get these points when you form a word using that letter.
- At the start of the game each player draws seven letter tiles each from the pool of 100 tiles.
- Tiles are replenished from the pool every time a tile is used meaning the players always hold seven tiles.
- Tiles in the pool and those of other players are kept secret so that a player can see only those tiles on the board and his own.
- To start the game the person with the letter closest to the letter ‘A’ has their turn first.

- The first player must place a word with at least one tile on the centre square of the board
- A player places down two or more tiles to form a word on the board, they can use tiles already placed on the board in their own word.
- If the word is in the dictionary then the player is awarded the score of the word
- To calculate the score of a word you add up all the points each letter is worth, accounting for any special squares letters are placed on
    - Double or Triple letter squares multiply the score of the letter that is placed on them
    - Double or Triple word squares multiply the score of the whole word formed
- Players continue playing in order
- If they cannot form a word they can spend their turn to swap all their tiles out for new ones in the bag
- The game ends when a player has run out of tiles or nobody can make another word, the winner is then decided by the player with more points.
- To calculate a player's score at the end of the game add up the points from all of their word scores, then subtract the sum of their unplayed letters.
- If one player uses all their letters, add the sum of everyone's unplayed letters to their score.

User Scenarios:

We framed our user scenarios as user stories, small statements of all the things an actor might do while playing the game. These small informal snippets of how the game will be interacted with helped us to flesh out the classes and methods for our initial design. We will probably think of more scenarios as development continues but that is the nature of iterative and agile development.

- As a user, I can choose to play against another person on the same machine, or against the PC.
- As a user, I can view the scoreboard of games played on this machine.
- As a player, I require 100 letter tiles to play Scrabble.
- As a player, I can draw 7 tiles at the beginning of the game.
- As a player, if I chose the letter closest to "A" I can play the first word in the game and place it on the star in the centre of the board.
- As a player, the words I play must be at least 2 letters.
- As a player, the words I play will be automatically checked against any active dictionaries.
- As a player, after my turn I can draw the number of tiles I just played.
- As a player, I can build on other player's words.
- As a player, I can exchange the tiles that I don't want for new ones after my turn.
- As a player, I can create unique dictionaries to play with.
- As a player, I can add or remove words from these custom dictionaries.

- As a player, I can import custom dictionaries from CSV files.
- As a player, I can choose to include or exclude certain custom dictionaries at the start of a game.
- As a player, I can choose to end a game at any point, provided the other player agrees (if there is one).
- As a player, I can enter my name, to be entered on a scoreboard with my score.

Primary Class List:

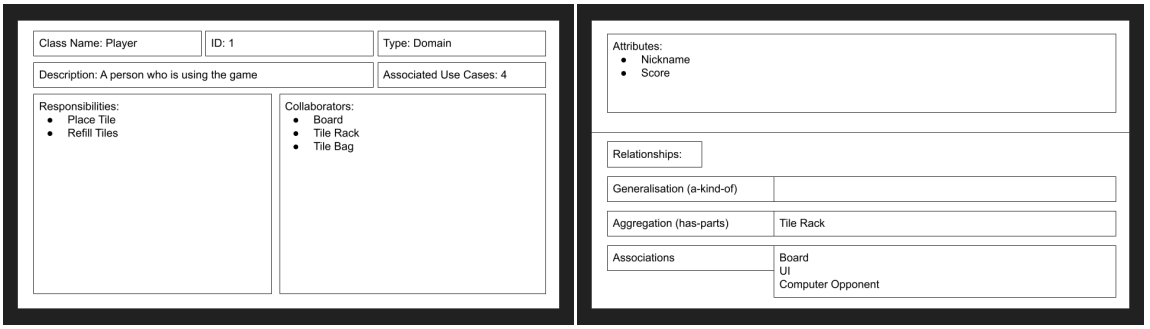
Here is the primary classes we have envisioned using in our implementation, you can see how they interact and behave in the CRC Cards and Class Diagram

1. Player
2. UI/Menu
3. Board
4. Tile Bag
5. Tile Rack
6. Tile
7. Rule Book
8. Dictionary
9. Computer Opponent

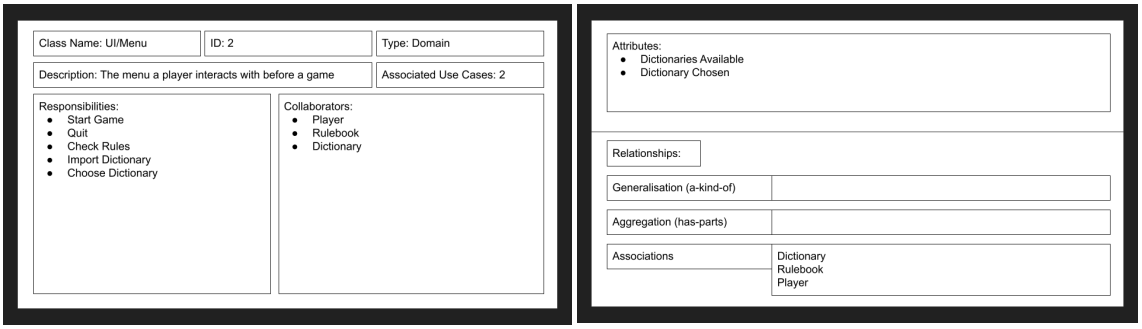
CRC Cards:

Here is the CRC Cards for our nine primary classes, if it is easier for you to read here is a link to the slideshow we made these on: [CRC CARDS]

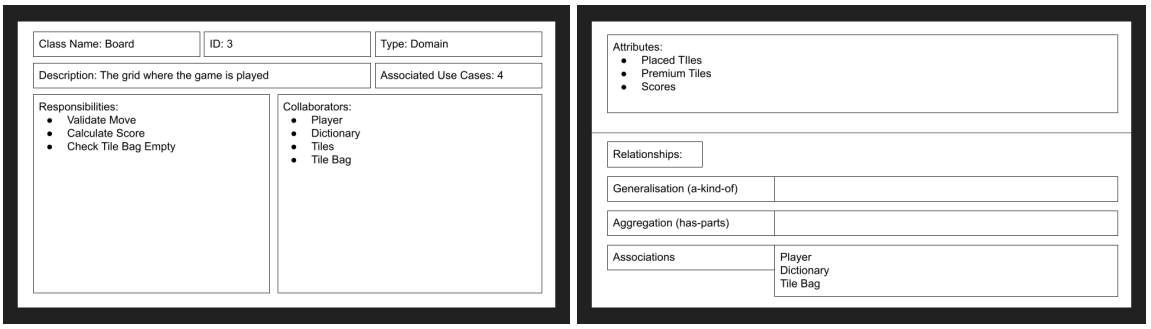
Player



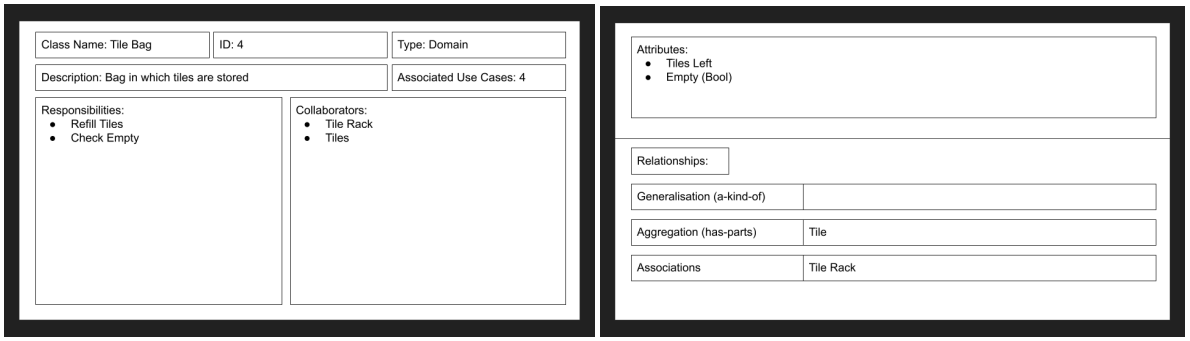
UI



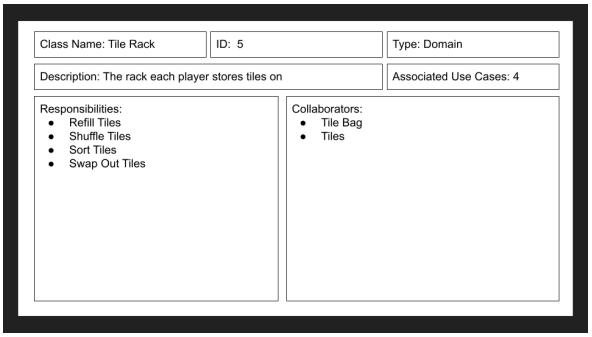
Board



Tile Bag



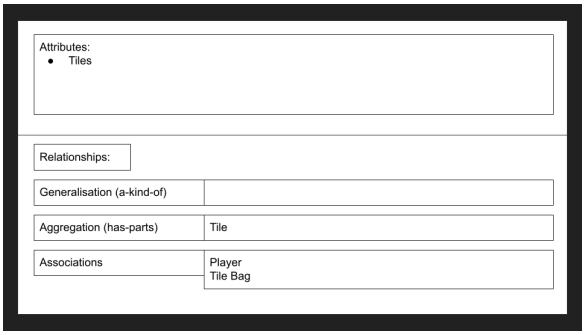
Tile Rack



Class Name: Tile Rack	ID: 5	Type: Domain
Description: The rack each player stores tiles on		Associated Use Cases: 4

Responsibilities:
- Refill Tiles
- Shuffle Tiles
- Sort Tiles
- Swap Out Tiles

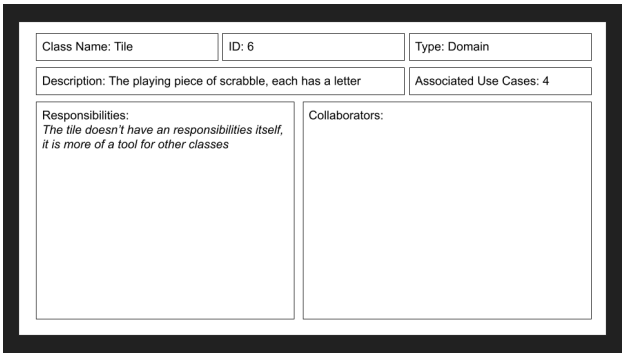
Collaborators:
- Tile Bag
- Tiles



Attributes:
- Tiles

Relationships:	
Generalisation (a-kind-of)	
Aggregation (has-parts)	Tile
Associations	Player Tile Bag

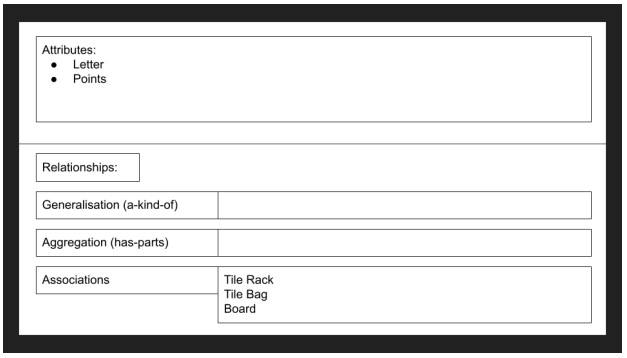
Tile



Class Name: Tile	ID: 6	Type: Domain
Description: The playing piece of scrabble, each has a letter		Associated Use Cases: 4

Responsibilities:
The tile doesn't have an responsibilities itself, it is more of a tool for other classes

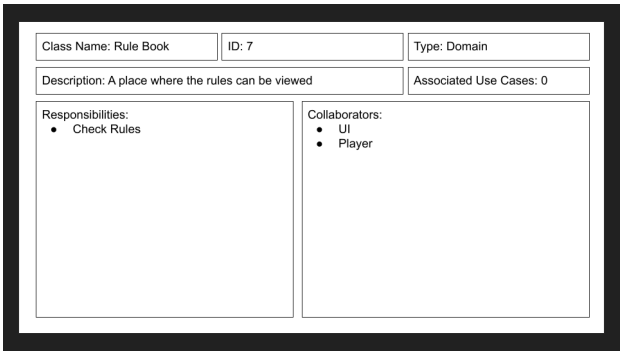
Collaborators:



Attributes:
- Letter
- Points

Relationships:	
Generalisation (a-kind-of)	
Aggregation (has-parts)	
Associations	Tile Rack Tile Bag Board

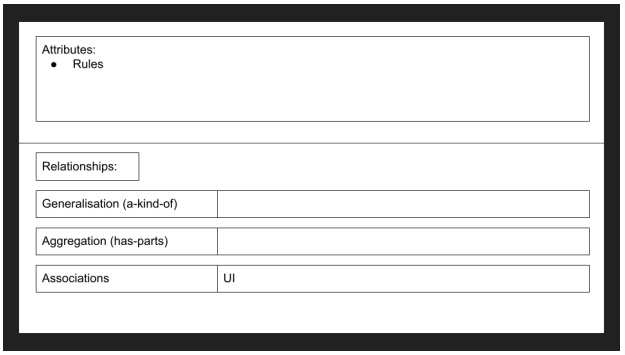
Rule Book



Class Name: Rule Book	ID: 7	Type: Domain
Description: A place where the rules can be viewed		Associated Use Cases: 0

Responsibilities:
- Check Rules

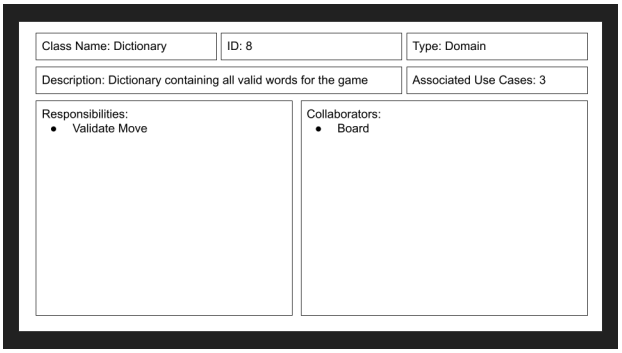
Collaborators:
- UI
- Player



Attributes:
- Rules

Relationships:	
Generalisation (a-kind-of)	
Aggregation (has-parts)	
Associations	UI

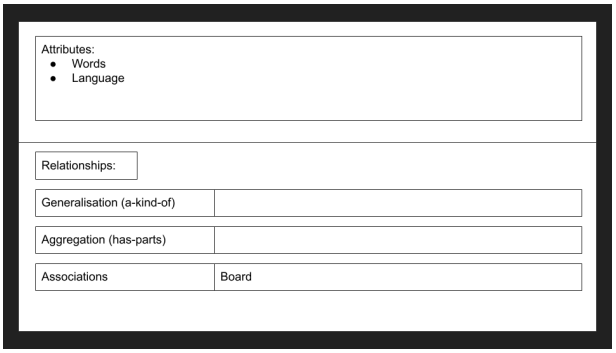
Dictionary



Class Name: Dictionary	ID: 8	Type: Domain
Description: Dictionary containing all valid words for the game		Associated Use Cases: 3

Responsibilities:
- Validate Move

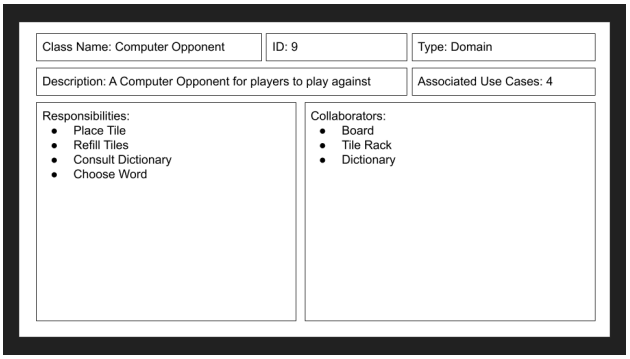
Collaborators:
- Board



Attributes:
- Words
- Language

Relationships:	
Generalisation (a-kind-of)	
Aggregation (has-parts)	
Associations	Board

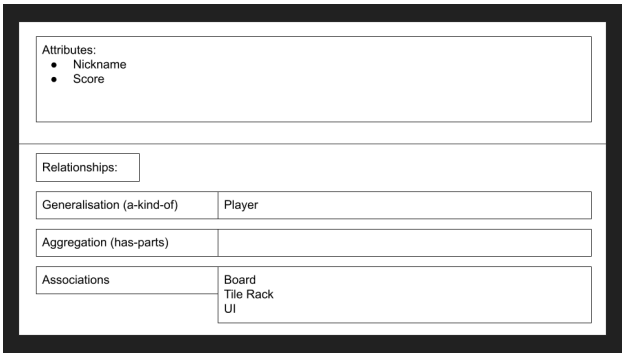
Computer Opponent



Class Name: Computer Opponent	ID: 9	Type: Domain
Description: A Computer Opponent for players to play against		Associated Use Cases: 4

Responsibilities:
- Place Tile
- Refill Tiles
- Consult Dictionary
- Choose Word

Collaborators:
- Board
- Tile Rack
- Dictionary



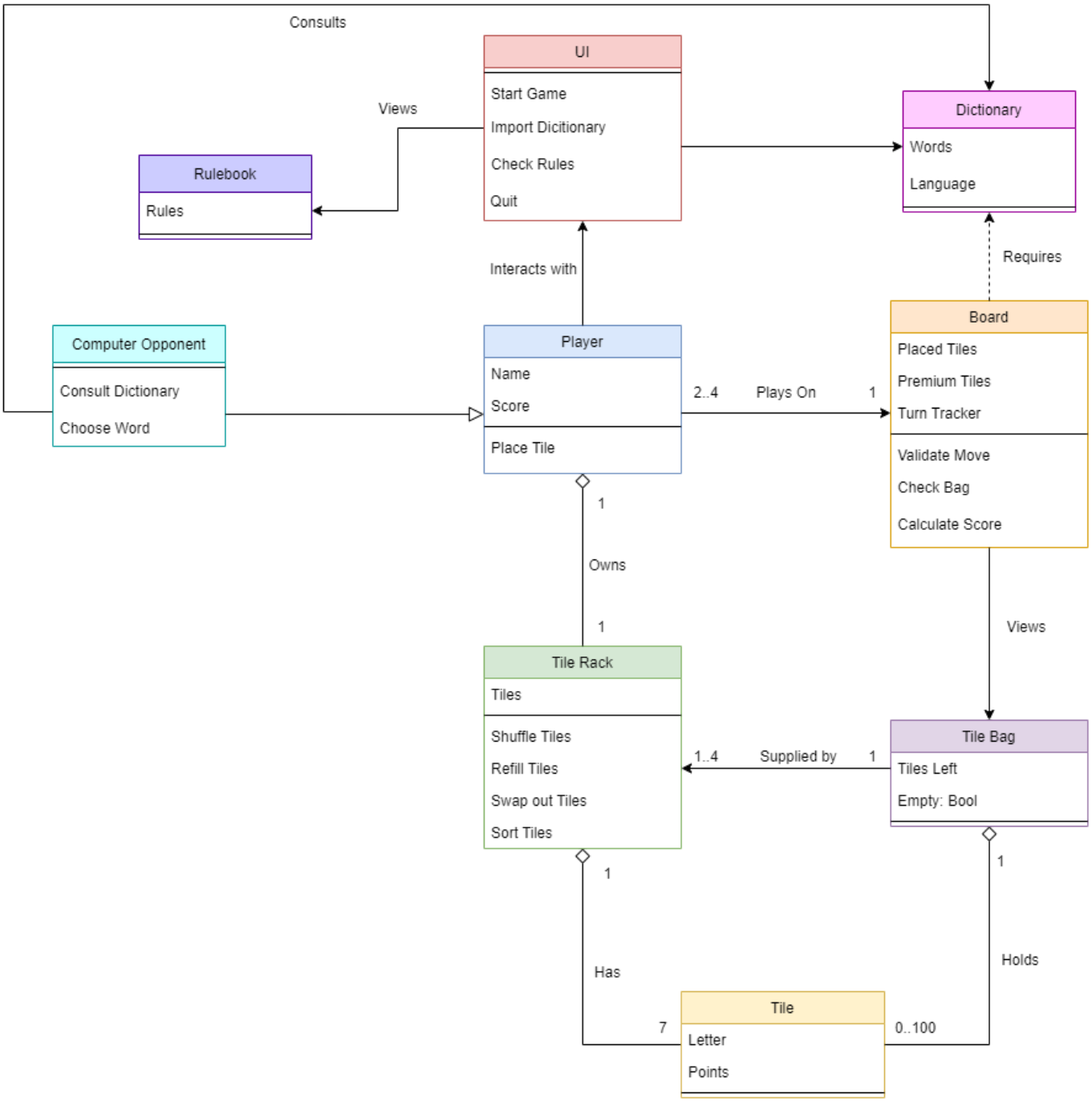
Attributes:
- Nickname
- Score

Relationships:	
Generalisation (a-kind-of)	Player
Aggregation (has-parts)	
Associations	Board Tile Rack UI

Class Diagram:

Here is our class diagram which shows how all the classes work together, we found making this diagram together very useful for visualising how the implementation of our game will work, how classes interact with each other and their internal attributes and methods.

This diagram will continue to grow and develop throughout the project, we hope to add to it and keep updated it as we add classes, methods and relations we couldn't envision yet in this early design.



Use Case Descriptions:

Here are descriptions of 5 use cases in our project, modelled after Cockburn's template. And after these descriptions you can see a mini use case diagram showing the actor (Player) and how they can interact with the system through use cases.

Use Case Description for First Play:

USE CASE 1	Chosen a letter closer to "A" than any other player	
Goal in Context	Get to play the first word in the game	
Scope & Level	System, Core.	
Preconditions	Word must be at least 2 letters, Must have drawn 7 tiles already, Letter chosen must be the closest to "A" out of all players	
Success End Condition	Word is valid, Eligible for multiple plays because premium square was incorporated into their play	
Failed End Condition	Word doesn't pass dictionary test	
Primary, **Secondary Actors**	Scrabble Player. Competitors. A bot	
Trigger	Game has just begun	
DESCRIPTION	**Step**	**Action**
	1	Player draws letter closest to "A" out of all players

	2	Player creates a word on the virtual board, places on premium square
	3	Word is automatically checked against active dictionary
	4	Word validity is checked
	5	Word is deemed as valid
	6	Player gets another turn since premium square was incorporated into their play
	7	Game continues
EXTENSIONS	**Step**	**Branching Action**
	4a	Word is deemed as invalid
VARIATIONS		**Branching Action**
	1	Premium square is not incorporated into first play

Use Case Description for Player's Turn:

USE CASE 2	User plays a turn

Goal in Context	User creates a word out of the 7 tiles they have	
Scope & Level	System, Core.	
Preconditions	Word must be at least 2 letters	
Success End Condition	Word is valid	
Failed End Condition	Word doesn't pass dictionary test (invalid)	
Primary, **Secondary Actors**	Scrabble Player. Competitors. A bot	
Trigger	User's turn to play.	
DESCRIPTION	**Step**	**Action**
	1	Player gets a turn in the game
	2	Player creates a word on the virtual scrabble board
	3	Word is automatically checked against active dictionary
	4	Word is deemed as valid
	5	Next user gets their own turn
	6	Game continues

EXTENSIONS	**Step**	**Branching Action**
	4a	Word played is deemed invalid by active dictionary
	6a	Entire play is deemed unacceptable. Player removes their tiles and forfeits their turn

Use Case Description for Swapping Out Tiles

USE CASE 3	Swap out tiles in User's tile rack	
Goal in Context	Receive a new set of tiles from the tile bag into the tile rack	
Scope & Level	System, Core.	
Preconditions	Player chooses to swap tiles	
Success End Condition	Tiles in User's rack have been replaced with new ones from the tile bag	
Failed End Condition	Not enough tiles left in the bag to swap	
Primary, **Secondary Actors**	Currently active scrabble player	
Trigger	User wants to swap tiles in rack with new ones from the bag	
DESCRIPTION	**Step**	**Action**

	1	Player gets a turn in the game
	2	Player chooses to swap tiles
	3	The system checks if there are enough tiles in the tile bag + user's tiles to refill the user's tile rack
	4	The old tiles are moved from the tile rack to the tile bag
	5	7 new tiles are selected at random from the tile bag
	6	The selected tiles are moved from the tile bag into the player's tile rack
EXTENSIONS	**Step**	**Branching Action**
	3a	Not enough tiles in tile bag to refill the player's
	3b	User informed that the swap has failed
VARIATIONS		**Branching Action**
	N/A	N/A

Use Case Description for Customising Dictionary

USE CASE 4	Customise Dictionary
Goal in Context	Import desired CSV file containing desired dictionary
Scope & Level	System, Core.

Preconditions	CSV file must be valid, Other players agree on dictionary choice	
Success End Condition	CSV file is successfully imported	
Failed End Condition	CSV file is corrupt/ Import failed	
Primary, **Secondary Actors**	Scrabble Player. Competitors. A bot	
Trigger	Game hasn't started yet, Players deciding on what's considered a valid dictionary for the game	
DESCRIPTION	**Step**	**Action**
	1	Game is about to begin
	2	Player's deciding on valid dictionary choices
	3	Player permitted to import CSV file
	4	CSV file is valid
	5	Game commences
	7	Words are constantly being checked against active dictionary for validity

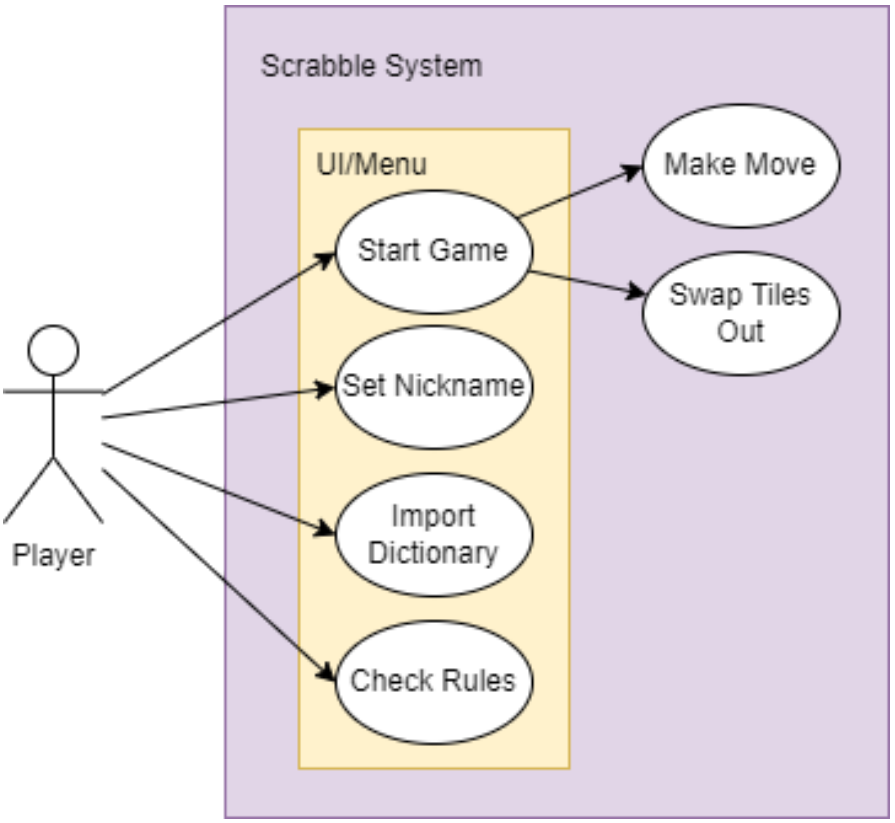
EXTENSIONS	Step	Branching Action
	3a	Player not permitted to use such dictionary as other player's disagree
	4a	CSV file is invalid/ corrupt
VARIATIONS		**Branching Action**
	1	CSV file may be invalid and user must import a different one

Use Case Description for Ending the Game

USE CASE 5	End the game
Goal in Context	End the game and record highest human score to scoreboard
Scope & Level	System, Core.
Preconditions	No more words can be made, OR All human players agree to end the game in its current state
Success End Condition	Game is ended and User(s) are returned to the Main Menu UI, Highest human score is recorded on the scoreboard
Failed End Condition	User enters invalid name to associate with score on scoreboard
Primary, **Secondary Actors**	Scrabble Player with highest score Competitors. A bot

Trigger	User(s) want to end the game, OR No more words can be created	
DESCRIPTION	**Step**	**Action**
	1	Player gets a turn in the game
	2	Player cannot make any word AND the tile bag is empty
	3	Player with highest score is prompted to enter a name to associate with their score on the scoreboard
	4	Name is valid
	5	System resets scoreboard
	6	System resets player classes
	7	Users are returned to main menu UI
EXTENSIONS	**Step**	**Branching Action**
	4a	Name is invalid
	4b	User is prompted to re-enter a name
VARIATIONS		**Branching Action**
	2	User(s) trigger an early end to the game

Use Case Diagram:



Structured Walkthrough:

Here is a walkthrough of how our classes and methods work together as a player initialises and plays through a game, Classes are highlighted in Orange, and their methods are highlighted in Blue, some attributes are shown in Green
Each step illustrates how one or more classes are used for each move and how they use their methods to interact with each other

- A Player checks the rules in the Rulebook through the UI
- A Player interacts with the UI to Start Game
- All Players Refill their Tile Rack from the Tile Bag
- Player order is determined by who has a closer tile to 'A' alphabetically
- The first Player Places Tiles to form a word
- The Board Validates the word using the Dictionary
- The Board calculates score, factoring in any premium tiles
- The Player Refills their Tile Rack again
- It is now the next Player's turn, Board checks if Tile Bag Empty
- The Player Shuffles their Tiles on the Tile Rack
- The Player then Sorts their Tiles
- The Player finds a word and Places Tiles on the Board
- The Board Validates the word and Calculates score
- The Player Refills their Tile Rack

This cycle continues with players playing their tiles and gaining score until..

- Tile Bag is empty
- Players can keep playing until no more words can be made
- Game is ended
- Player with higher score wins

This walkthrough demonstrates the main flow of the game we have anticipated so far, there are more specific methods and attributes detailed in the class diagram that don't fit in the main flow of a normal game.

Conclusion:

After this early stage of early design and analysis we have set out a framework for how we are going to frame and implement the game, but we know that as we continue we will encounter things we did not anticipate during these first stages.

This is a vital part of the agile development style, as we iterate, implement and discuss problems as they occur our design and plan will grow with the project. Many of the things set out here only cover a basic view of the game, how we anticipate it working. We will continue to update designs found here, especially the class diagram, through the development cycle.

Team Minutes:

Sunday 25/09, 14.00-14.30

- Met the group, brainstormed each part of submission one, decided good times for future meetings and language for our implementation will be Python

Monday 26/09, 10.35-10.50

- Split out work to do for the next week (items 1, 2, 3), made a google drive to share documents

Wednesday 28/09, 10.35-10.50

- Shared feedback and checked in on the progress of the first three items for submission one

Friday 30/09, 12.00-12.20

- Shared out the next items (3a, 4, 5) for the first submission, talked about the schedule for submitting on time

Friday 07/10 14.20 - 15.30

- Last group session to finalise and bring together all elements into the report for submission.