**Table of Contents**

[1. Requirements Elicitation 1](#_Toc131611325)

[1.1. Game guide 1](#_Toc131611326)

[1.2. Use Case Diagram 2](#_Toc131611327)

[1.3. Use Case Descriptions 2](#_Toc131611328)

[2. Requirement Analysis 11](#_Toc131611329)

[2.1. Sequence Diagram – Register Players 11](#_Toc131611330)

[2.2. Sequence Diagram – Move 12](#_Toc131611331)

[2.3. Sequence Diagram – Landing On Area 13](#_Toc131611332)

[2.4. Sequence Diagram – Landing On Chance 14](#_Toc131611333)

[2.5. Sequence Diagram – Start A Development 16](#_Toc131611334)

[2.6. Sequence Diagram – Sell 17](#_Toc131611335)

[2.7. Sequence Diagram – End Game 18](#_Toc131611336)

[3. System Design 19](#_Toc131611337)

[Appendix I 22](#_Toc131611338)

[Appendix II 31](#_Toc131611339)

[Appendix III 46](#_Toc131611340)

[Appendix IV 47](#_Toc131611341)

# Requirements Elicitation

## Game guide

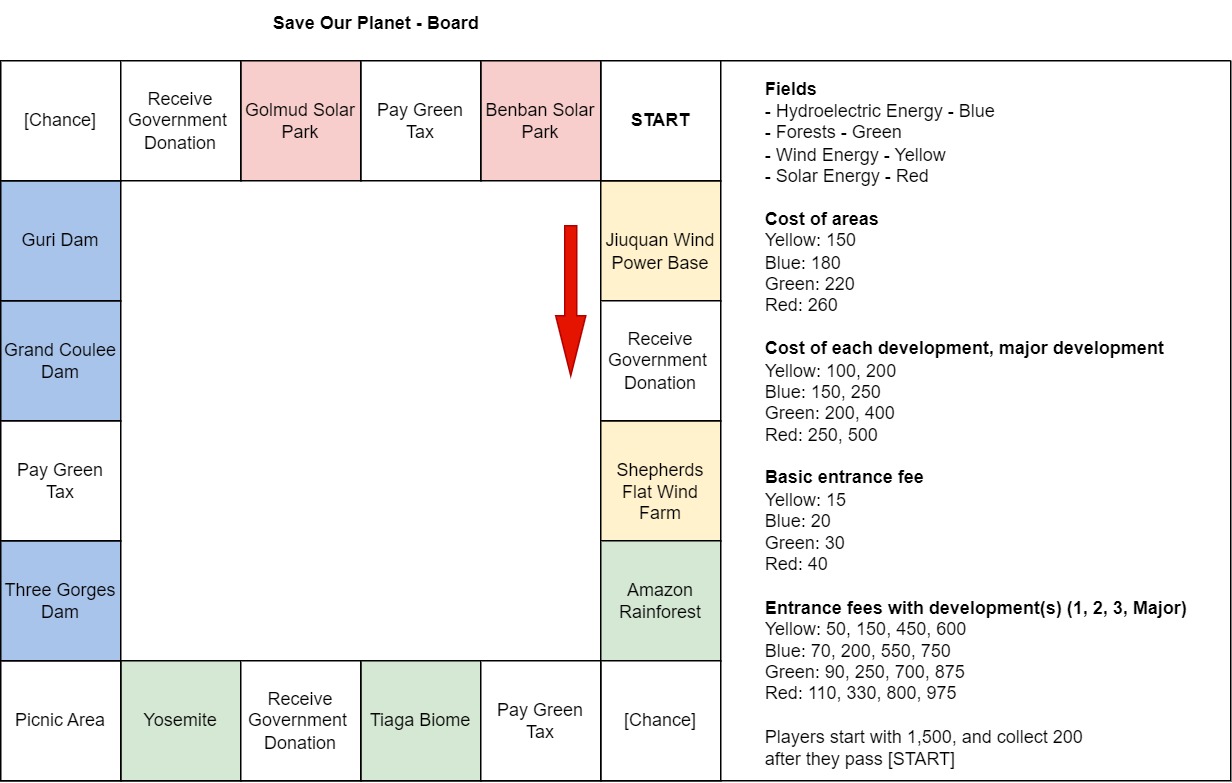


Figure 1 - Game Guide

## Use Case Diagram

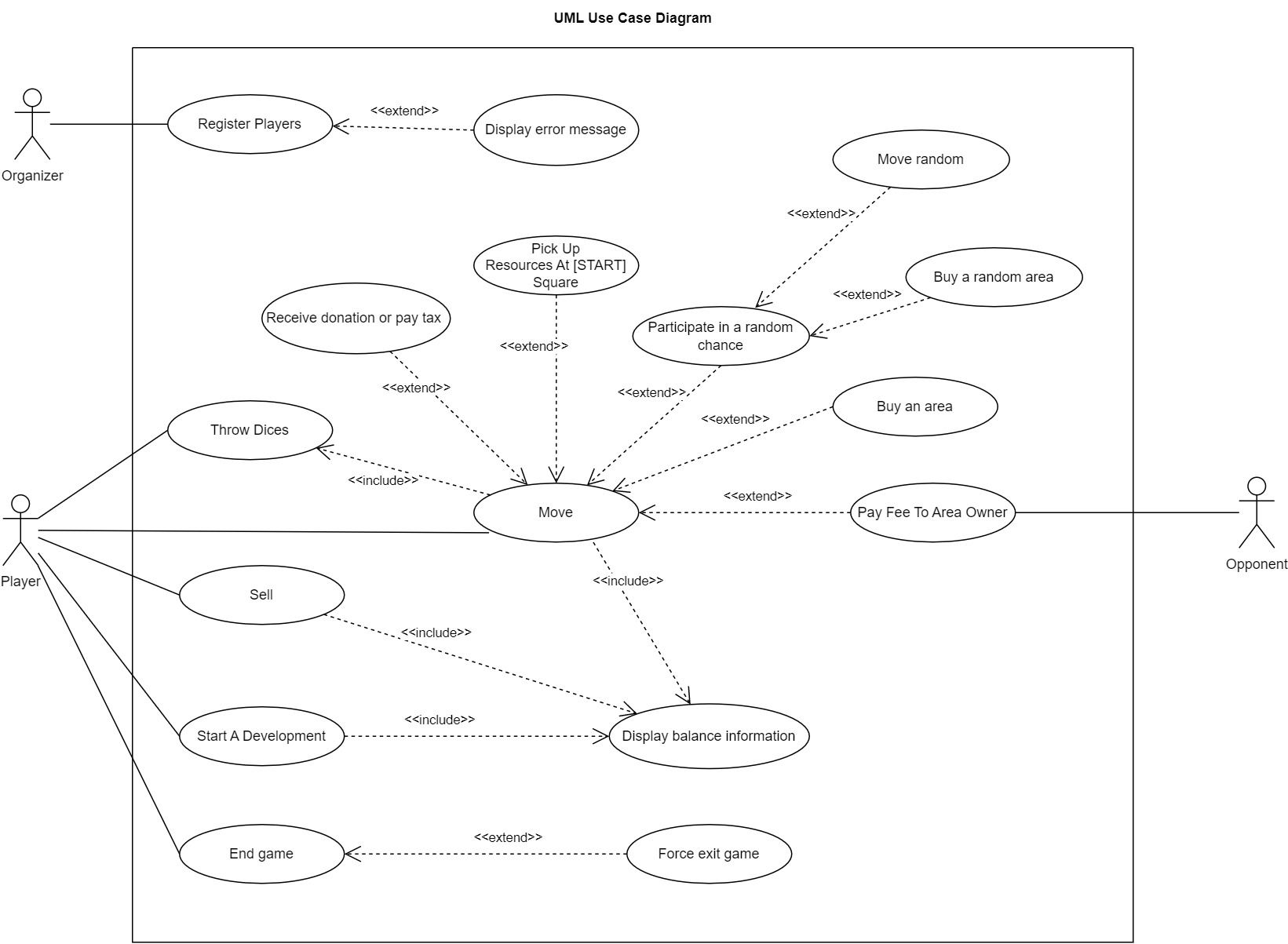
****

Figure 2 - Use Case Diagram

## Use Case Descriptions

|  |  |
| --- | --- |
| Name | [UC1] Register Players |
| Objective | To register players to the gameplay. |
| Actors | Organizer |
| Precondition | - |
| Main Flow | 1. A player enters the total number of players successfully [AF1]. The valid number of players shall be between 2 and 4. 2. Player enters their name in turns [AF2]. The name being entered shall not be identical to any name previously entered. |
| Alternative Flows [AF1] | 1. At [1], player enters the total number of players outside of valid range. 2. System displays error message and prompts user to enter a valid number again. 3. If user enters valid number, processing continues at [2].   If user enters invalid number again, processing continues at [AF1] [2]. |
| Alternative Flows [AF2] | 1. At [2], player enters a name which is identical to a name previously entered by another player. 2. System displays error message and prompt user to enter a valid name again. 3. If user enters a valid name, processing continues at [3].   If user enters an invalid name, processing continues at [AF2] [2]. |
| Post-conditions | Players are registered for the upcoming gameplay. |

|  |  |
| --- | --- |
| Name | [UC2] Move |
| Objective | To move player to a new position on the board |
| Actors | Player |
| Precondition | The player’s turn has started. |
| Main Flow | 1. Players will play the game in turns. When a player’s turn starts, system will display the player’s name and current balance. 2. Player throws dices [UC3]. 3. The player moves a number of steps according to the player’s dice roll results. System evaluates player’s new position and displays the player’s obligations and/or opportunities according to whether the player passes [START] square [UC4], and the type of square the player lands on.   When the player lands on a new square, he or she might: Buy an Area [UC5], Pay Fees to owner [UC6], Receive Donation or Pay Tax [UC7], or Participate in a Random Chance [UC8]. |
| Alternative Flows | - |
| Post-conditions | The player is in a new position. |

|  |  |
| --- | --- |
| Name | [UC3] Throw Dices |
| Objective | To determine a player's number of moves in his or her turn. |
| Actors | Player |
| Precondition | The player’s turn has started. |
| Main Flow | 1. System asks if the player is ready to throw the dices. In this game, player will roll two virtual dices, each with six faces with values ranging from 1 to 6. 2. When the player enters his confirmation, system will roll the virtual dices. 3. System displays the result of the throw. |
| Alternative Flows | - |
| Post-conditions | Player gets the number of moves to make in this turn. |

|  |  |
| --- | --- |
| Name | [UC4] Pick Up Resources at [START] Square |
| Objective | To pick up some resources when player passes [START] square |
| Actors | Player |
| Precondition | 1. The player’s turn has started. 2. The player has thrown dices. [UC3] 3. The player lands on [START] square or moves past [START] square. |
| Main Flow | 1. The player gains [200]. 2. System displays the reason for the change in resources and player’s new balance. |
| Alternative Flows | - |
| Post-conditions | The player’s balance increases by [200]. |

|  |  |
| --- | --- |
| Name | [UC5] Buy an Area |
| Objective | To own an area |
| Actors | Player |
| Precondition | 1. The player’s turn has started. 2. The player has thrown dices. [UC3] 3. The player lands on an available area that no one owns. |
| Main Flow | 1. System informs the player about the opportunity to take charge of this area and the area details. 2. The player might confirm to buy the area by entering his or her decision [AF1]. System then confirms and displays the acquisition and the updated resources information. System displays the reason for the change in resources and player’s new balance.   If the player denies this opportunity, the player’s movement ends. |
| Alternative Flows | [AF1] If the player does not have enough money to buy the area, system will inform the user. The player’s movement ends. |
| Post-conditions | The player acquires an area. |

|  |  |
| --- | --- |
| Name | [UC6] Pay Fee to Area Owner |
| Objective | To pay fees to the owner of the area. |
| Actors | Player, Opponent |
| Precondition | 1. The player’s turn has started. 2. The player has thrown dices. [UC3] 3. The player lands on an area which is owned by another player. |
| Main Flow | 1. System informs the player about the obligation to pay some fees as specified in the game rule to the owner of the area and deducts this amount from the player’s balance [AF1]. 2. System displays the reason for the change in resources and player’s new balance. |
| Alternative Flows | [AF1] If the player does not have enough money to pay the fee, the system will ask if they wish to sell developments or areas [UC10]. If they can’t or won’t, they lose the game. The player’s areas then become available for other players to buy. |
| Post-conditions | 1. The player’s balance decreases by the specified amount. 2. The opponent’s balance increases by the specified amount. |

|  |  |
| --- | --- |
| Name | [UC7] Receive Donation or Donate |
| Objective | To let the player gains/losses a certain amount of resource. |
| Actors | Player |
| Precondition | 1. The player’s turn has started. 2. The player has thrown dices. [UC3] 3. The player lands on ‘Receive Donation’ or ‘Pay Tax’ square. |
| Main Flow | 1. If the player lands on ‘Receive Donation’ square, he or she will receive an amount of resource as specified by the square. If the player lands on ‘Pay Tax’ square, he or she will pay an amount of resource as specified by the square (200) [AF1]. System will inform the player of the event, based on the square that he or she lands on. 2. System displays the reason for the change in resources and player’s new balance. |
| Alternative Flows | [AF1] If the player does not have enough money to pay, the system will ask if they wish to sell developments or areas [UC10]. If they can’t or won’t, they lose the game. The player’s areas then become available for other players to buy. |
| Post-conditions | The player’s balance decreases/increases by the specified amount. |

|  |  |
| --- | --- |
| Name | [UC8] Participate in a random chance |
| Objective | To participate in a random opportunity. |
| Actors | Player |
| Precondition | 1. The player’s turn has started. 2. The player has thrown dices. [UC3] 3. The player lands on [Chance] square. |
| Main Flow | 1. System informs player of the opportunity. 2. System draws a random action from its [Chance] pool, which might be either: Move Forward (S1), Move Backward (S2), Purchase an Area (S3) |
| Sub Flows | (S1) If the player receives ‘Move Forward’ chance, system will randomly move the player forward by 1, 2 or 3 squares. The player has to move to the new position and receives any obligation/opportunity presented in the new position.  (S2) If the player receives ‘Move Backward’ chance, system will randomly move the player backward by 1, 2 or 3 squares. The player has to move to the new position and receives any obligation/opportunity presented in the new position.  (S3) If the player receives ‘Purchase an Area’ chance, system will inform the player of the opportunity to buy any available area on the board, and ask for player’s decision to proceed with the opportunity.  If the player accepts the chance, system will display the information of all available areas on the board. The player then enters the area that they want to acquire. If the selected area is within the player’s budget, system then acknowledges the acquisition and displays the updated resources information. System displays the reason for the change in resources and player’s new balance.  If the player rejects the chance, the use case ends. |
| Post-conditions | The player participates in a random opportunity. |

|  |  |
| --- | --- |
| Name | [UC9] Start a Development |
| Objective | To develop an area within an acquired field. |
| Actors | Player |
| Precondition | 1. The player’s turn has started. 2. The player has finished his or her Move (UC2). 3. The player is in charge of at least one field. |
| Main Flow | 1. The player is in charge of a field when he or she acquires all areas within a field. By then, the player can start a development in any areas within the field. An area can have up to 3 developments, and the next development will be a major development. The player cannot further develop an area after a major development has been made. He or she can only implement one development per turn. 2. System displays information of all areas qualified for development and/or major development. 3. Player enters their chosen area for development. [AF1] 4. System records the development of the chosen area and player’s new balance. 5. System displays the reason for the change in resources and player’s new balance. |
| Alternative Flows | [AF1] Player can reject the opportunity to develop an area in this turn by entering his or her decision. |
| Post-conditions | The player completes a development in an area that he or she owns. |

|  |  |
| --- | --- |
| Name | [UC10] Sell |
| Objective | To sell an area or development to the system. |
| Actors | Player |
| Precondition | 1. The player’s turn has started. 2. The player is obligated to pay money that they don’t have. (UC6, UC7) |
| Main Flow | 1. The system will inform the player that they have insufficient resource to pay for their obligation. 2. If the player has no assets/properties left to sell, he or she is disqualified from the game. Else, the system displays all properties that the player is able to sell, and if the player is in charge of any fields. The system then asks the player if they wish to sell some developments or areas. 3. If the player accepts the opportunity, the player enters which area and the number of developments they wish to sell to the system [AF1].   Before the player attempts to sell an area within a field they are in charge of, he or she must sell off all developments within the field first. To facilitate this game rule, after receiving input from the player, two different alternatives of system behaviors are proposed:   * If there is no development within the field, system would try to sell off the selected area. * If there is any development within the field, system would try to sell the developments within the area.  1. The system displays the changes in balance after the transaction. 2. The system will check if the player has enough money to complete their obligation. If yes, then the system will complete the obligation. If not, the system will go back to 2. If the player has no property to sell, they are disqualified from the game. |
| Alternative Flows | [AF1] Player can reject the opportunity to sell and they are then disqualified from the game. |
| Post-conditions | The player sells a development, or an area, to make resources available to pay any outstanding obligations. |

|  |  |
| --- | --- |
| Name | [UC11] End Game |
| Objective | To end the game |
| Actors | Player |
| Precondition | - |
| Main Flow | 1. A player will be out of the game if he or she has no resource or properties left [UC10]. When one player remains, he or she is the winner. 2. System displays information of resources each player holds. |
| Alternative Flows | If a player no longer wants to play, the player can stop the game in their turn and the game ends. System displays information of resources each player holds. |
| Post-conditions | Game exits. |

# Requirement Analysis

## Sequence Diagram – Register Players

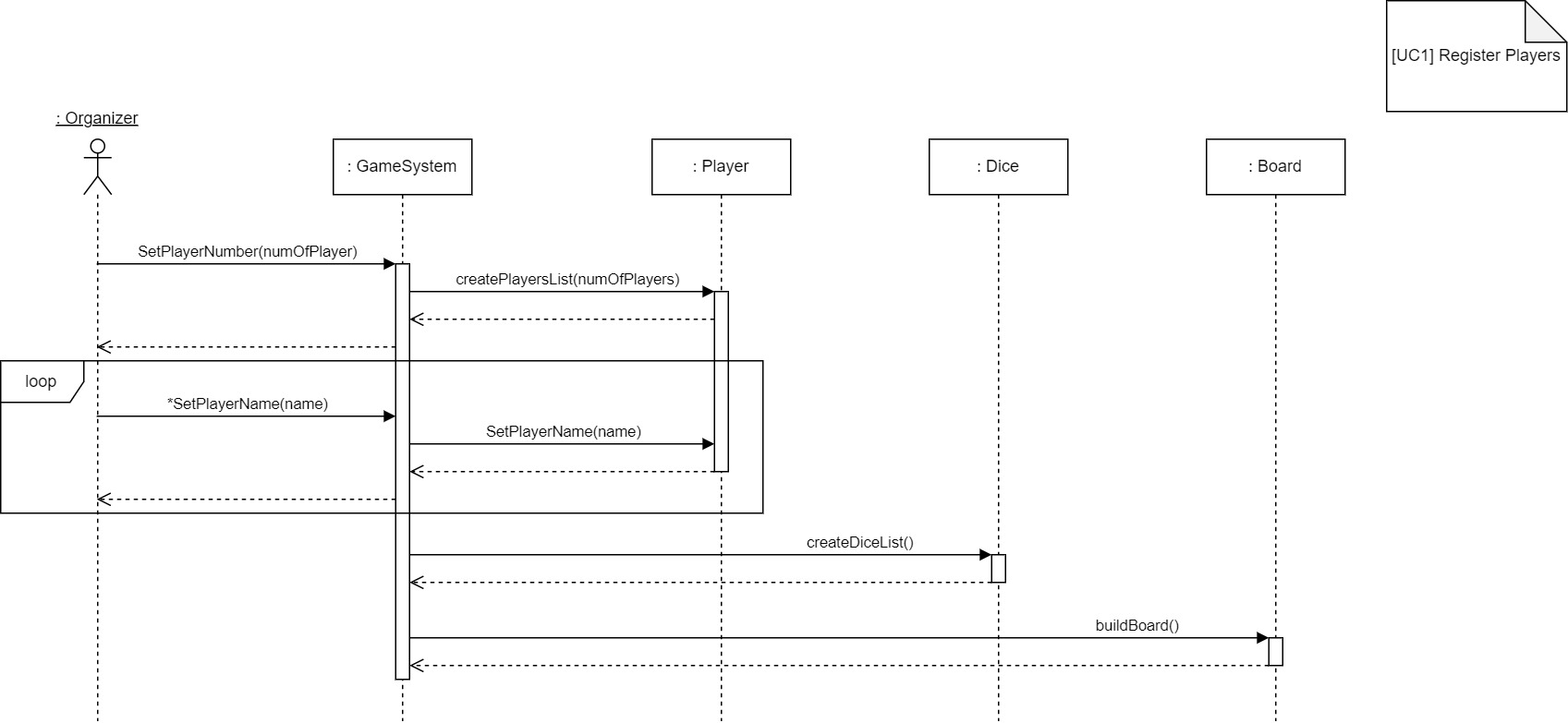


Figure 3 - Sequence Diagram: Register Players

Figure 3 describes system’s internal processes when registering players into the game system. First, the organizer will enter the total number of players, which will then be used to create a list of player objects (:Player). The system will then execute a loop to prompt the organizer to enter the player’s name in turn, and call each player object to update the player’s name. Before the game starts, the system will also prepare the game environment including creating a list of dice objects (:Dice) and call for the board object (:Board) to be built.

## Sequence Diagram – Move

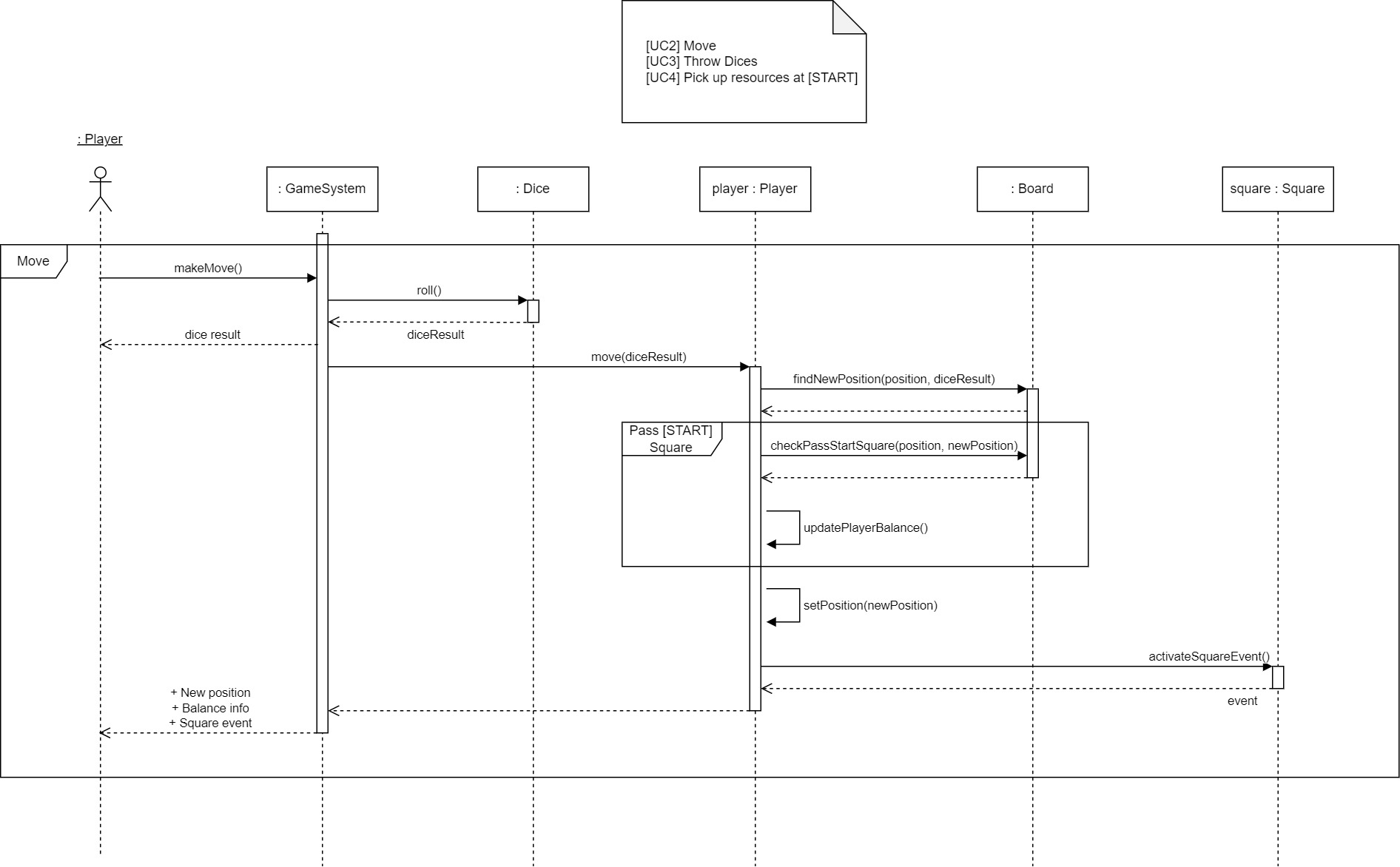


Figure 4 - Sequence Diagram: Move

Figure 4 describes the behavior of the system when the player confirms to make a move to land a new position on the board. Rolling dice is the first action to determine the player’s number of moves, in which the game system will ask each of the two dice objects to return a virtual face value. The system will then trigger the movement of the current player object by first calling the board object to find the new position based on the result of the dice rolls, followed by updating the player’s new position. It is also reasonable to include the check if the player passes START square so he or she will receive a bonus resource (corresponding to UC4) during the move, since this behavior concerns evaluating the player’s movement as well as their new position relative to the START square. Finally, the system will trigger any event associated with the new position, and since this behavior always accompanies the behavior of a player landing on a new space, it is included in the player’s move call.

## Sequence Diagram – Landing On Area

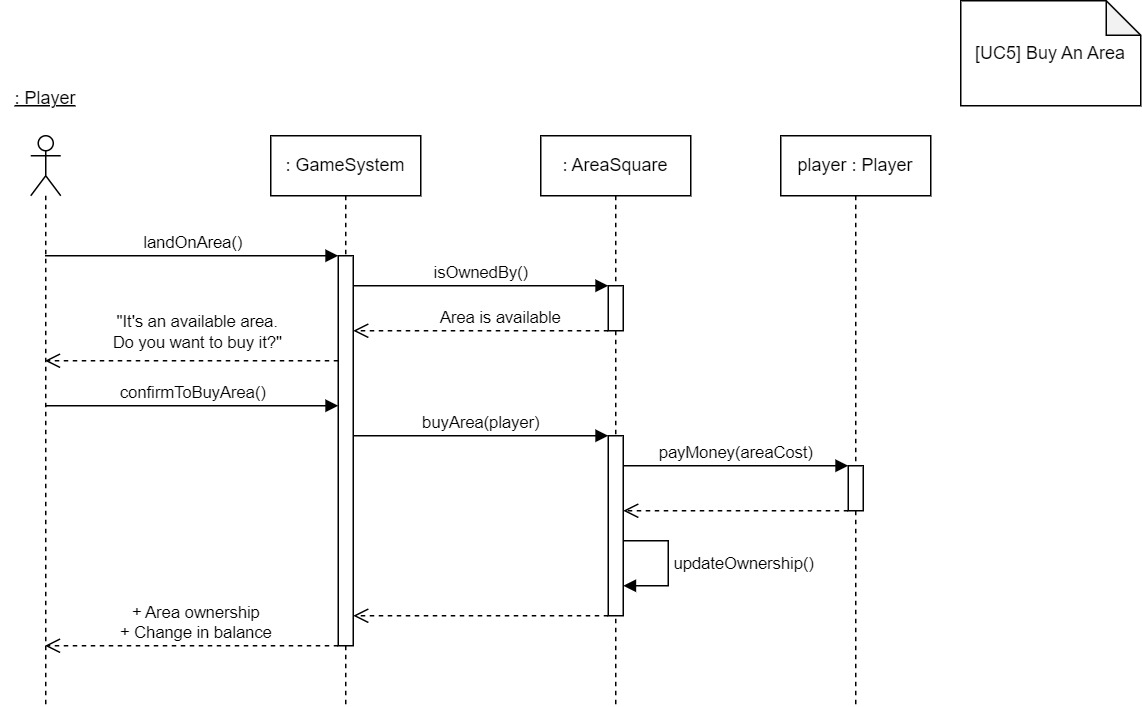


Figure 5 - Sequence Diagram: Buy Area

Figure 5 specifies the system behaviors after the player makes a move (as described in Figure 4) and lands on an area. System will first check with the area object (:AreaSquare)  to see if it was owned by any player, and the diagram illustrates the case when the area is not owned by any player and available for purchase. The player can choose to skip, which would result in simply ending the opportunity, or accept to buy this area. Once the player confirms his intention to buy the area, the game system will send a request to the area object to process the transaction, which involves the player object to pay money for the cost of purchase, and the area object to update the ownership. To end the buying process, the game system informs the player of the acquisition and the change in his or her balance.

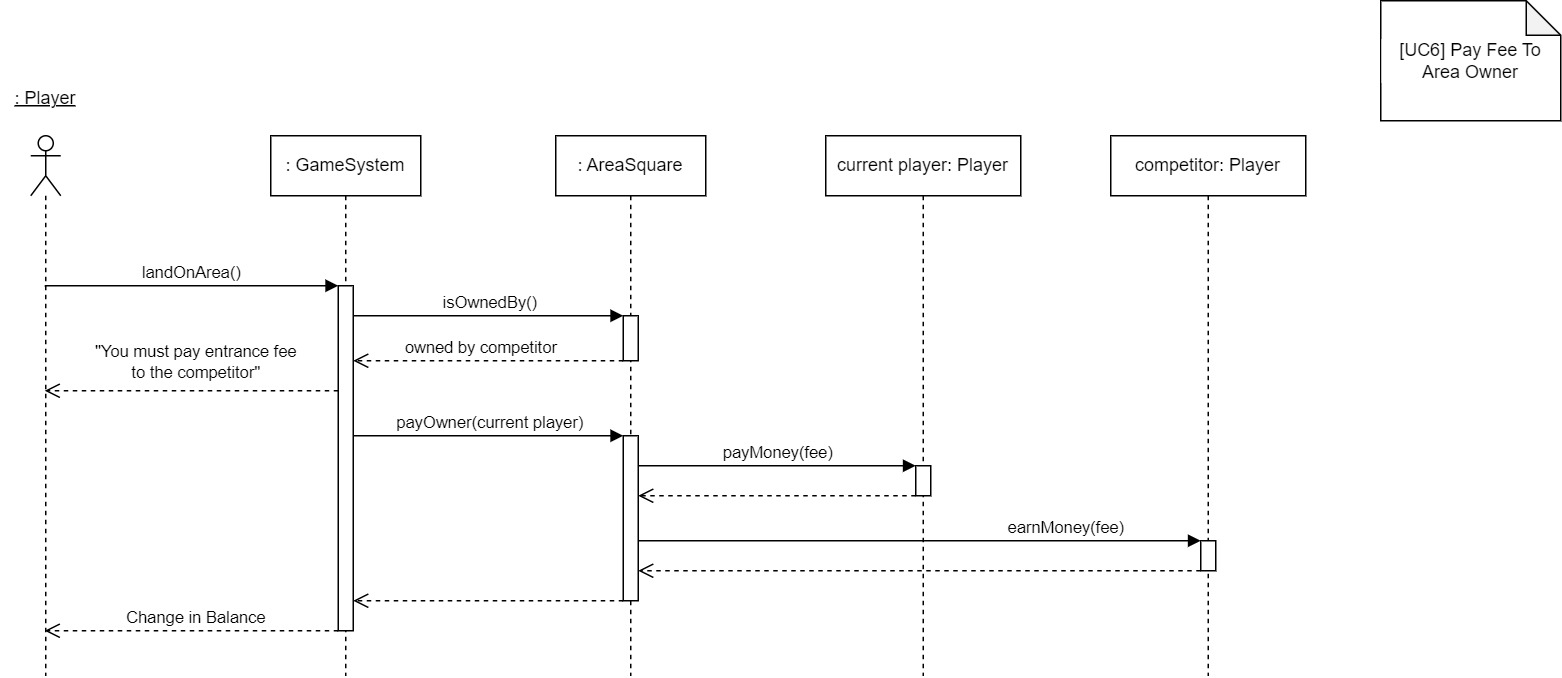


Figure 6 - Sequence Diagram: Pay Fee To Area Owner

Figure 6 depicts another case when the player lands on an area. If the area is owned by one of his or her competitor, the player will have to pay the entrance fee specified by the area policy to the owner. The game system will automatically send a request to the area object, in which it will retrieve its applied entrance fee and send a request to the current player object to pay the money, and another request to the competitor object to receive the money.

## Sequence Diagram – Landing On Chance

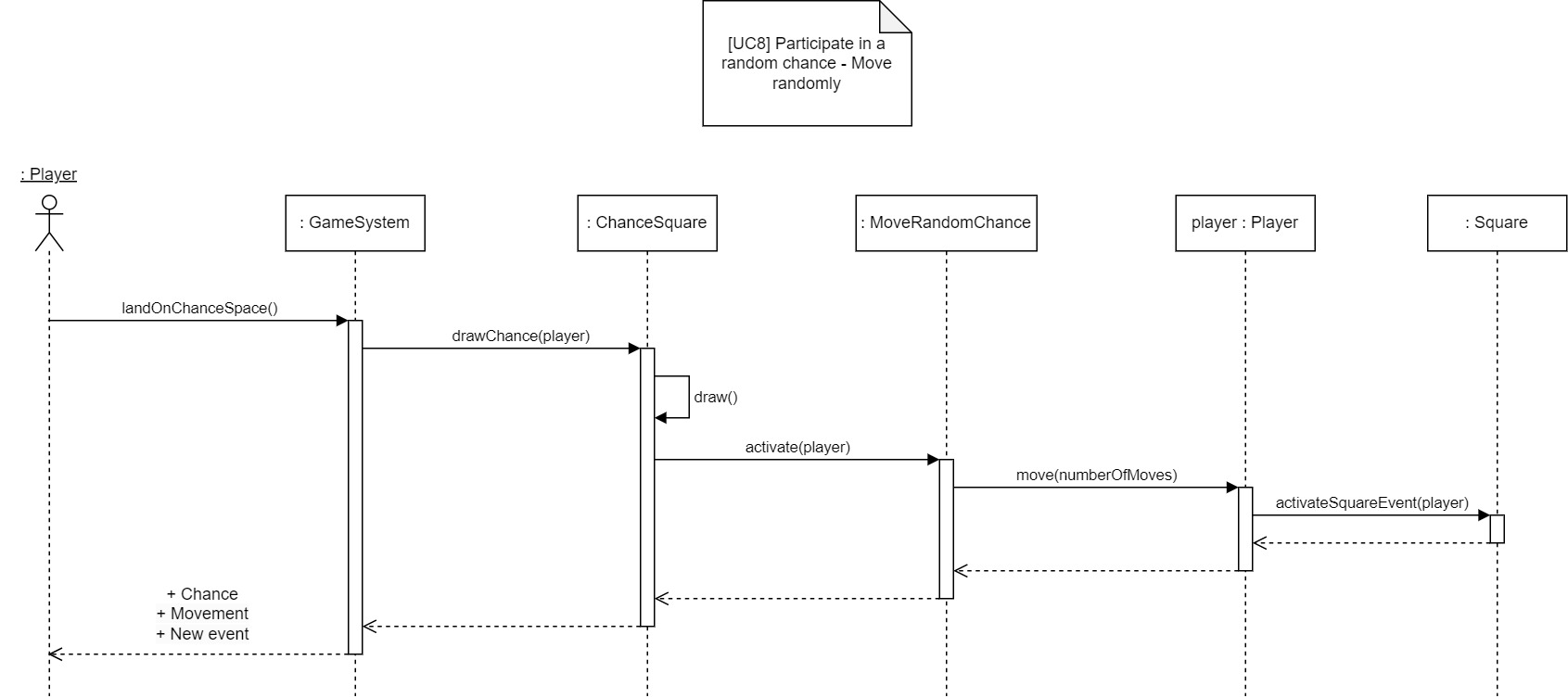


Figure 7 - Sequence Diagram: Move Random Chance

When the player lands on a Chance square, the game system will draw a random chance from its “chance pool” and trigger any event associated with the chance. Figure 7 describes the case of drawing a “Move Random” chance, in which the player will be moved forward or backward a random number of moves. After receiving a request to draw a chance from the game system, the chance square object (:ChanceSquare) will execute the draw, and call activate on the Move Random chance object (:MoveRandomChance). The chance object will then generate a random number of steps, and ask the player object to move. Further details of a single move have been illustrated in Figure 4, and instead of passing in the dice roll results to method *move()*, in this case we pass in the number of steps that are generated previously when activating the Move Random chance object.

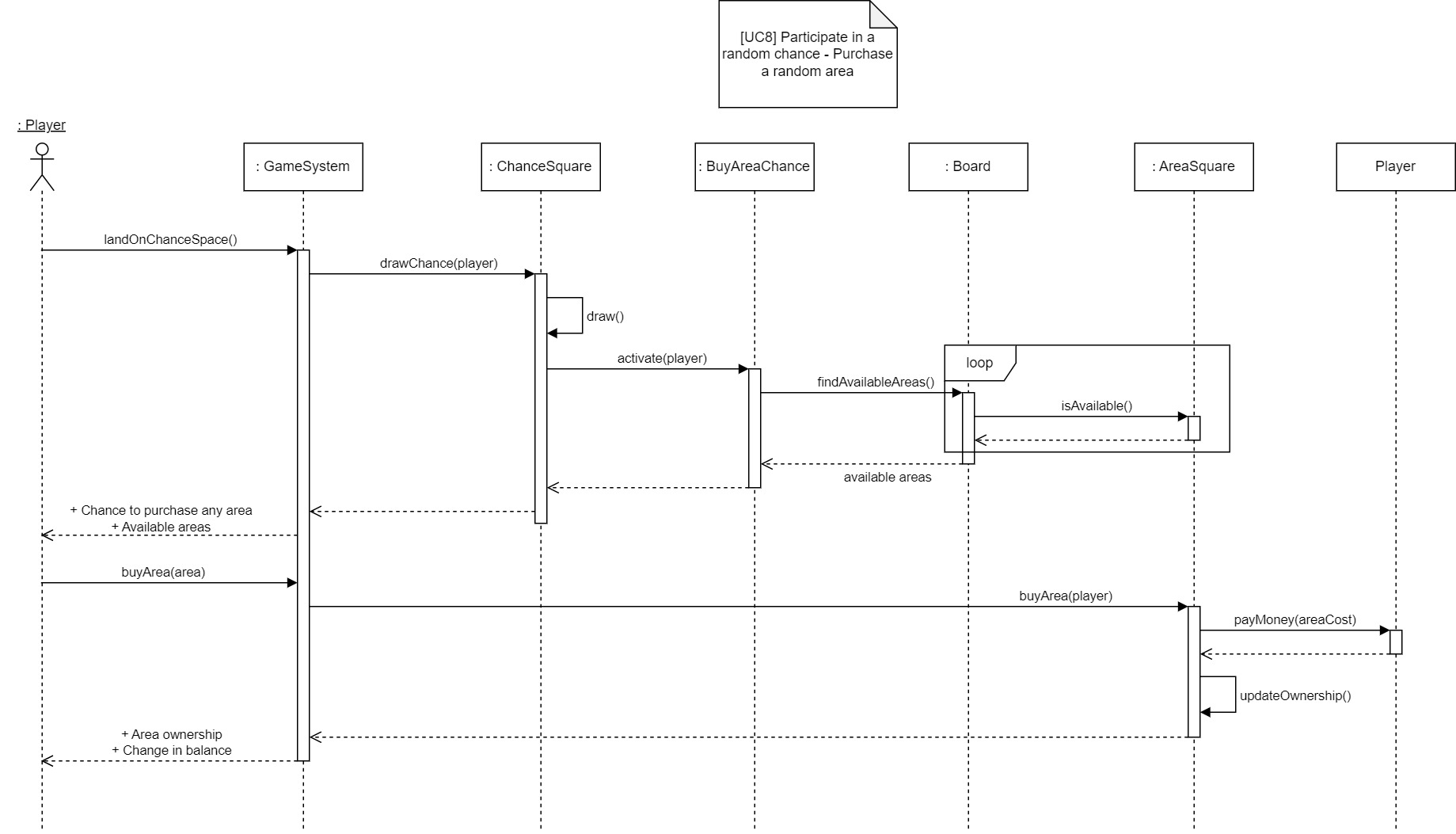


Figure 8 - Sequence Diagram: Purchase Any Area

Figure 8 illustrates another result of the draw, “Purchase Area” chance, which allows people to select any available area on the board to purchase. The game system will call the Buy Area chance object (:BuyAreaChance) to activate its event. Once activated, it will send a request to board object to find all current available areas on the board, which involves the board object going through the list of area objects and ask each area object if it is available for purchase. The player can then see the list of available areas returned from the game system, make his or her own evaluation and select the area that they want to purchase. The game system will then forward the request *buyArea()* to the respective area object that the player has selected, through which it will process the transaction to ask the player object to pay money for the cost of area, and update the area ownership.

## Sequence Diagram – Start A Development

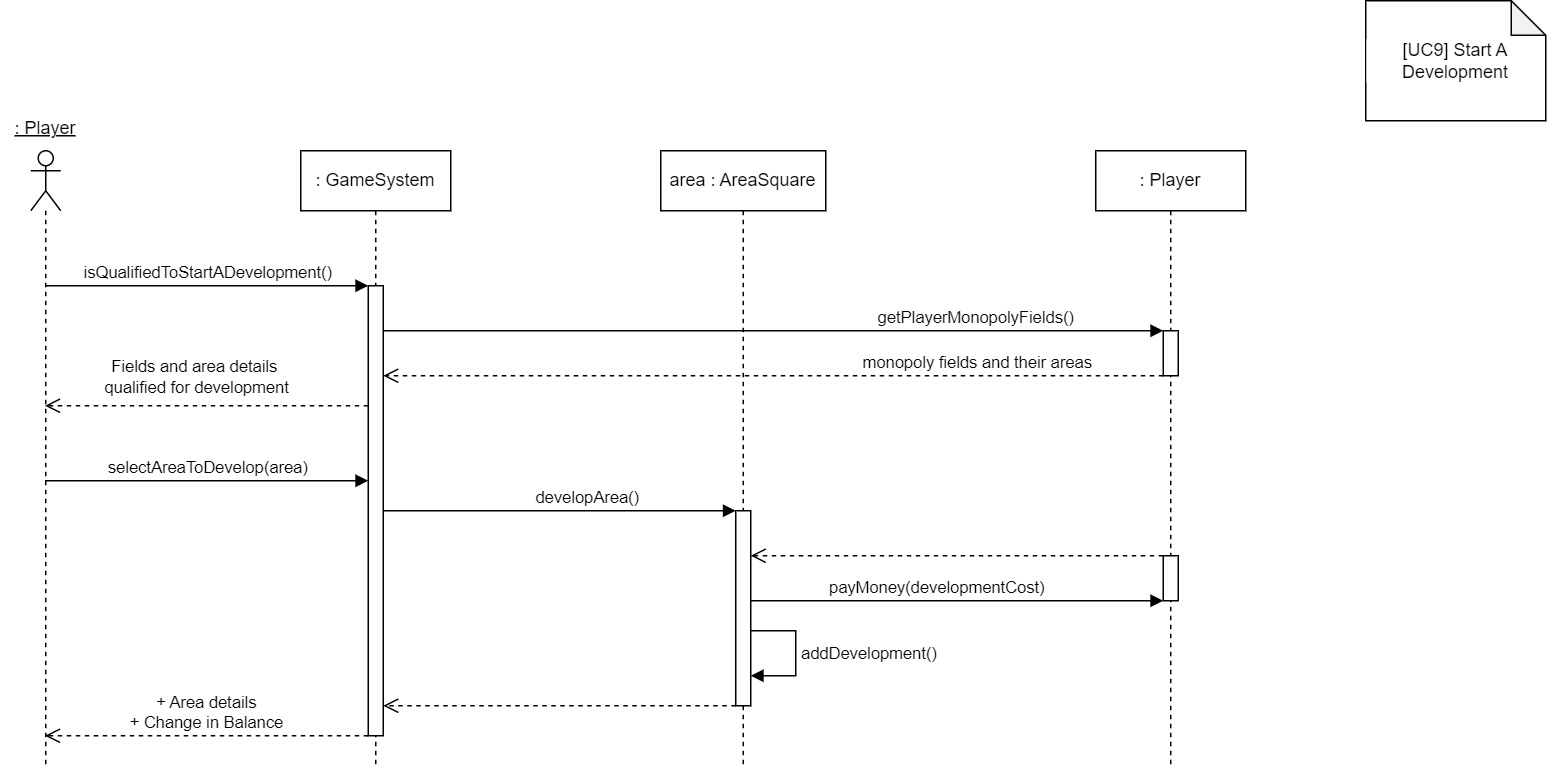


Figure 9 - Sequence Diagram: Start A Development

In a single turn, the player can choose to start a development once he or she finishes the move. As demonstrated in Figure 9, the game system will first check if the player is qualified to promote any development, given that the player must own the whole field (monopoly of the field), or be in charge of every area within a field, in order to trigger the development event. The system will call the player object to request for a list of fields the player is currently in charge of, which requires the player object to keep track of its monopoly fields. After extracting the list of monopolies, the game system will then show the current player this list of fields and their respective areas that are qualified for development. The diagram portrays the typical case when the player accepts the opportunity and enters their choice of area. The system will send a request to develop to the area object, which then triggers the Player object to pay money for the development cost, and the area object to update its development status.

## Sequence Diagram – Sell

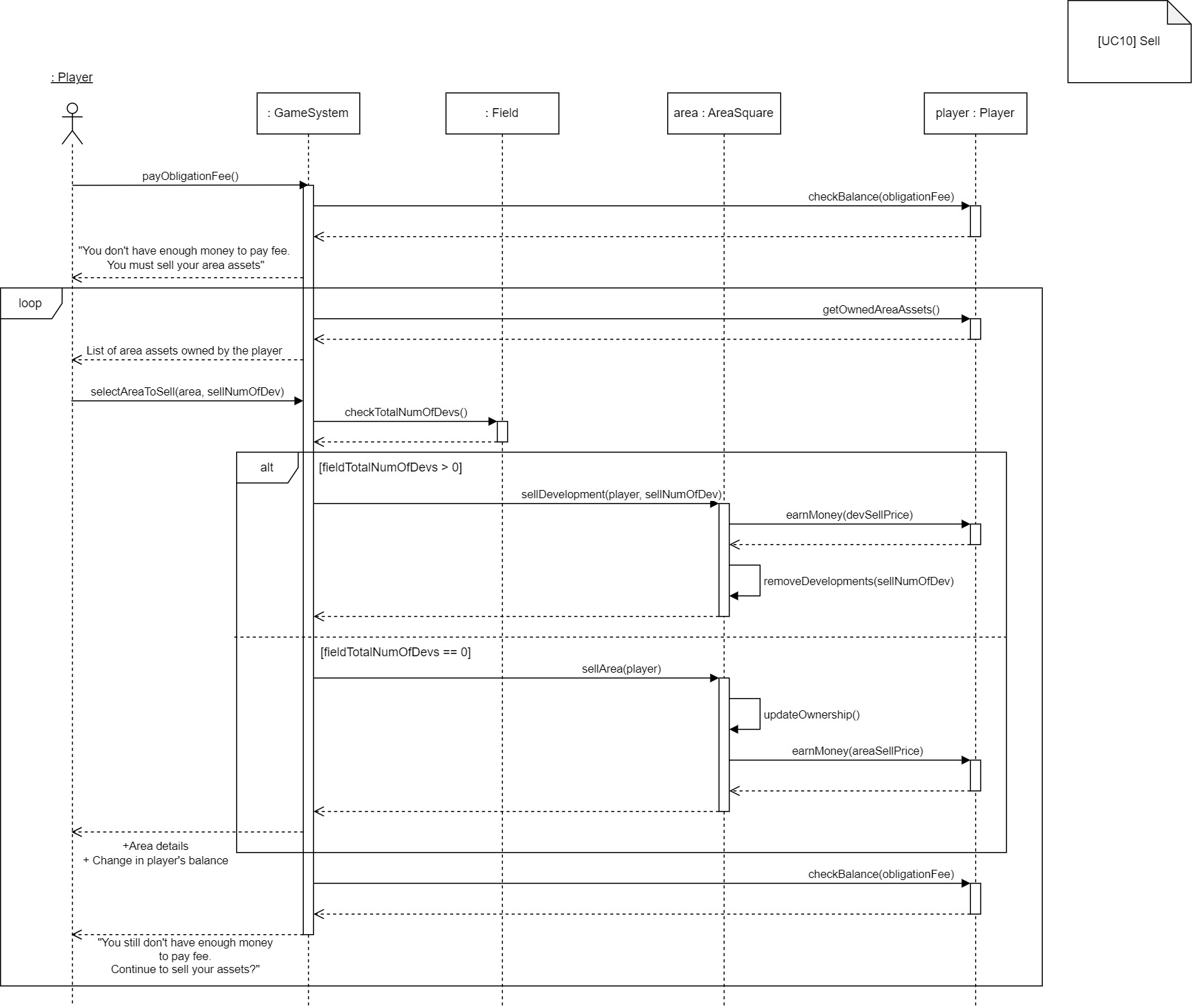


Figure 10 - Sell

When the player is in a situation in which he or she must pay an obligation fee, for example landing on a Pay Tax square, or landing on an area owned by his or her competitor, the player must keep selling his or her assets, specifically any area or development in the area, until acquiring enough money to pay off the obligation. As illustrated in Figure 10, the game system will execute a loop to keep prompting the user to sell their assets as long as their balance is not enough. The system will first retrieve the list of areas owned by the player object, which should be tracked by the player object for every buying/selling area transaction. It will display the retrieved list of areas for the player to select. The player will then enter his or her choice of area and the number of developments to sell. Considering the rule in selling assets stating that all developments within the field should be sold before the area could be sold, for every selling attempt the system will need to check if there is any development in the field that the selling-attempted area belongs to. If there is any development found, the system will always try to sell the development, otherwise it will sell the selected area. The diagram specifies two alternatives, which correspond to the game rule above. After retrieving the total number of developments from the field object, it will make its evaluation. If the total is greater than 0, it will send a *sellDevelopment()* request to the area object, in which the player object will earn an amount of selling developments price, and remove developments from the area object. In the game implementation, the system will also need to check the development status of the area before actually process the selling (it cannot sell the selected number of developments if there are not enough developments to sell), which is not portrayed in the diagram for simplification purpose. If the total is 0, the system will send a *sellArea()* request to the area object, which involves updating the area object’s ownership, and calling the player object to earn an amount of selling the area price. Finally, the game system will check if the player has had enough balance to pay off the obligation, and the diagram illustrates the case when the player still hasn’t met the requirement and the loop will continue.

## Sequence Diagram – End Game

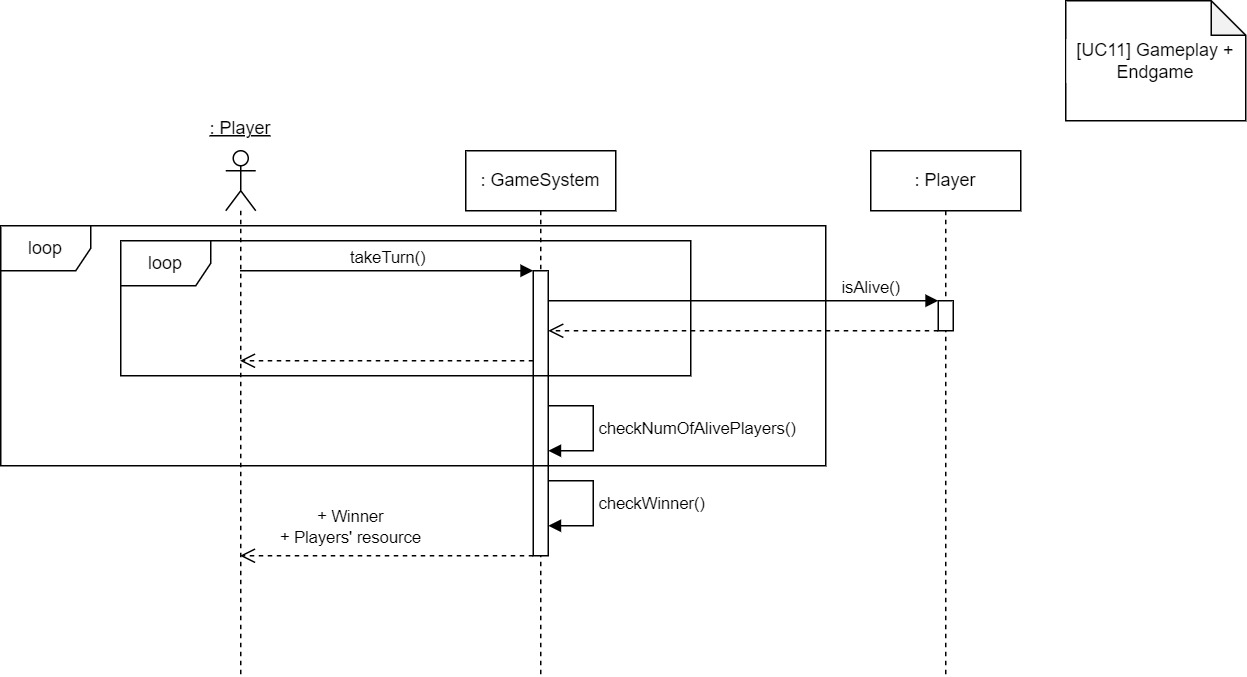


Figure 11 - Sequence Diagram: End Game

Figure 11 illustrates the general gameplay as well as the end game logic. Player will take turns, and in each turn the game system will check with the current player object if player is alive, then count to the number of alive players, which would then be reset after all players have taken their turn. Once the number of alive players decreases to one, the loop will terminate and the game will end, after which the system will announce the winner as well as the players’ resource.

# System Design

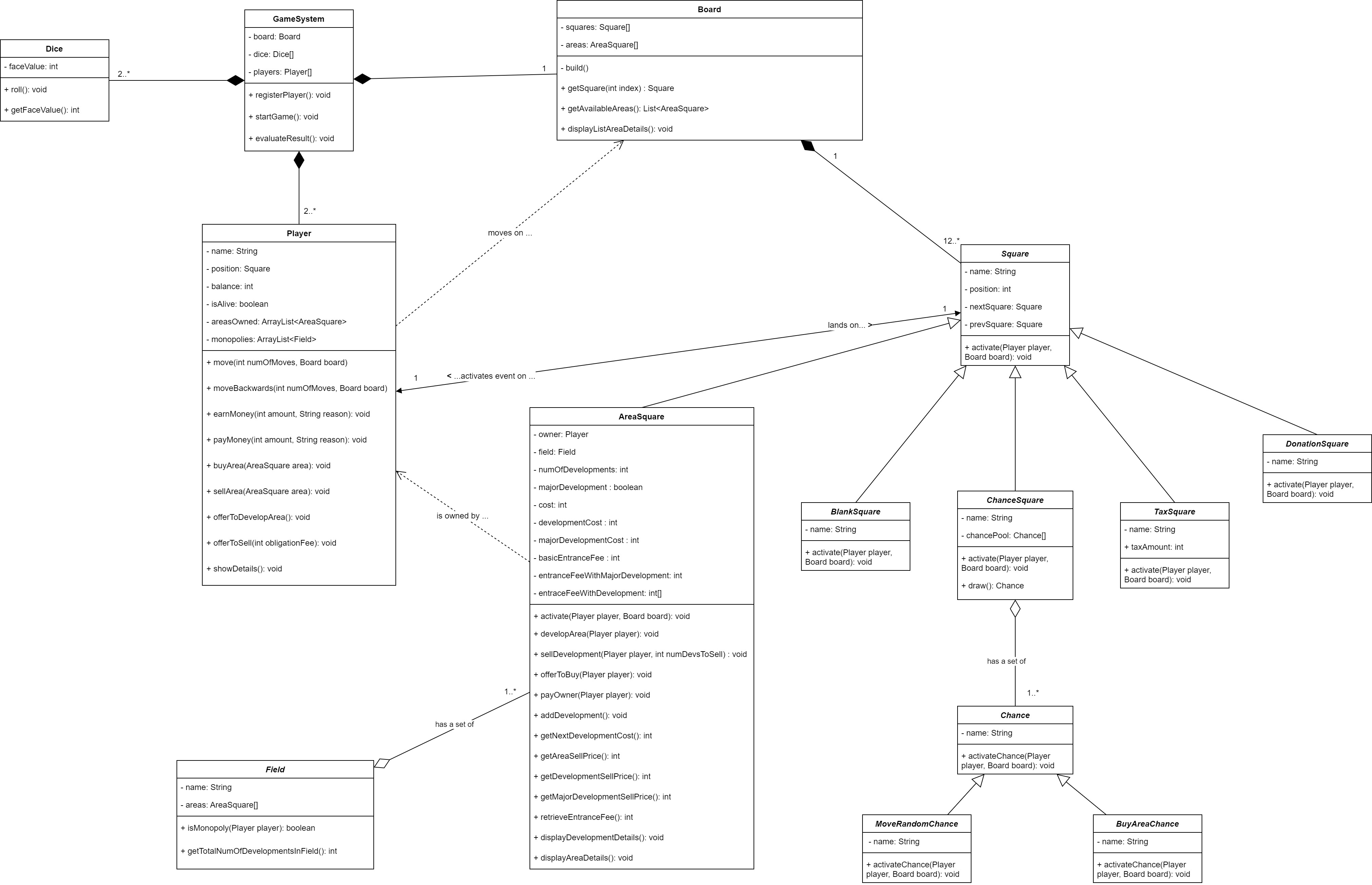


Figure 12 - Class Diagram

A Class Diagram, as illustrated in Figure 12, has been constructed to describe the encapsulation of individual classes and the relationships among them.

We can summarize the main ideas behind the class encapsulation, including:  Board encapsulates the list of squares and records the location of individual square on the board; Chance and Square would have behavior of activating any event associated with it; Player encapsulates the player’s name, player’s current position on the board, current balance, its status if the player is still alive and behaviors such as moving forward/backwards on the board, earning and paying money;  Dice encapsulates the behavior of rolling and a face value as attribute to represent a result of a roll; AreaSquare encapsulates its owner, costs and prices for its buying/selling transactions, and the field it belongs to; Field encapsulates a list of areas; GameSystem that manages the operation of the game encapsulates a board instance, a list of dices and a list of players registered to the gameplay.

For this game system, five types of relationship are identified: association, dependency, aggregation, composition and inheritance.

*Association*: There is a one-to-one association relationship between Player and Square, in which when a player lands on a particular square, it will activate its event on that particular player.

*Dependency*: Two dependency relationships are found in this game. First, the relationship between Player and Board can be explained by the fact in this game setting that a player moves on a board. Without the board object, the Player’s encapsulated *move()* method cannot be applied. Second, the dependency relationship between AreaSquare and Player can be realized based on the game rule that an area could be owned by a particular player. An area object uses a concrete instance of Player to capture its owner, which will then be used to evaluate which event (buy area, pay fee to owner, etc.) the area will activate when a player lands on its space.

*Aggregation*: An aggregation relationship is discovered between Field and AreaSquare; ChanceSquare and Chance. Field aggregates AreaSquare since one or more areas establish a field, or we can describe it as areas are ‘*part-of*’ a field. ChanceSquare aggregates Chance, and it represents a ‘*has-a*’ relationship. Further explanation for why we do not identify the relationship between these classes as Composition are presented in Composition part.

*Composition*:  The relationship between Board and Square is realized to be Composition, in which a board object is composed of multiple square objects. Similarly, the Composition relationship between GameSystem and other classes including Dice, Player and Board is also discovered.

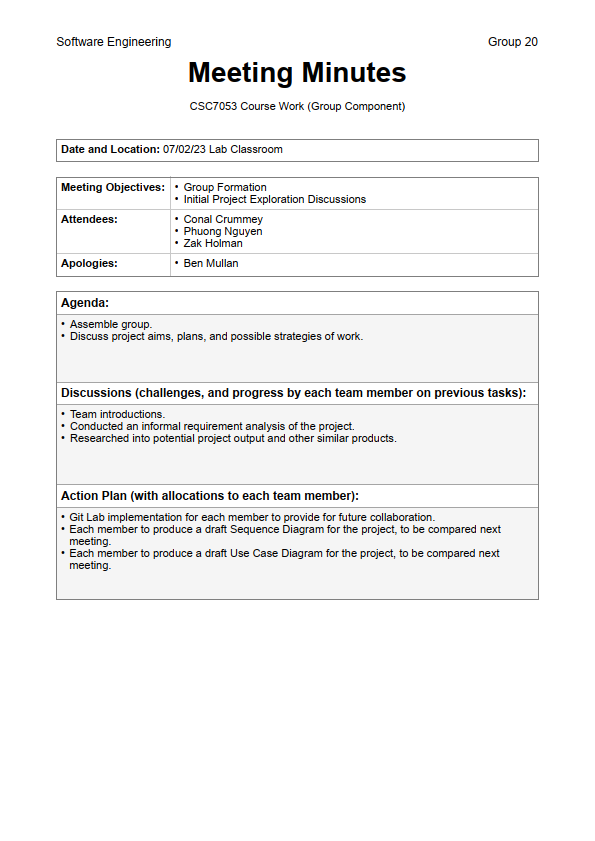
It is notable that we identify the Composition relationship between these classes base on a characteristic of Composition, in which the lifetime of the parts is dependent on the lifetime of their host. In this case, there is no meaning to keep individual Square instances if they are not integrated to build a board object. Without GameSystem, Dice, Player and Board couldn’t also exist. This is not the case for the Aggregation relationships we describe above, since both AreaSquare and Chance can exist independently without their hosts Field and ChanceSquare respectively.

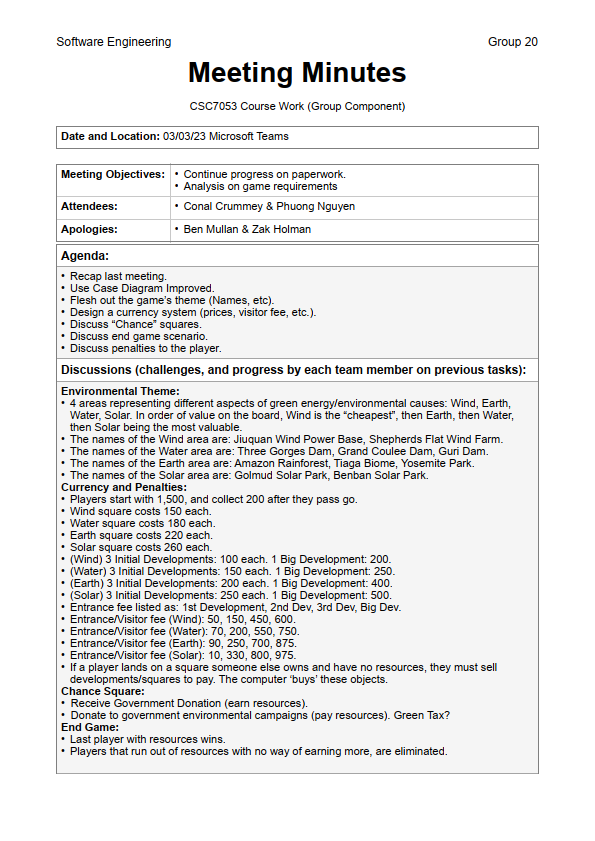
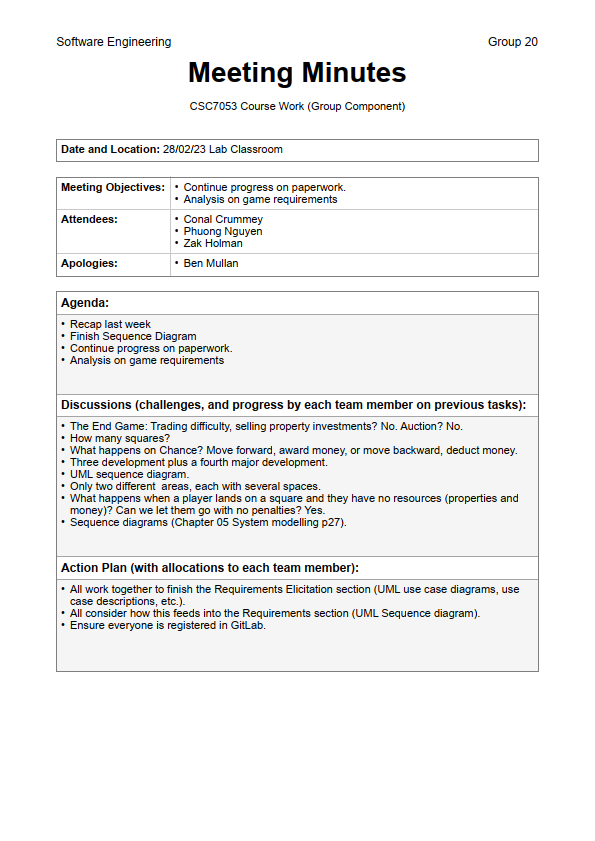
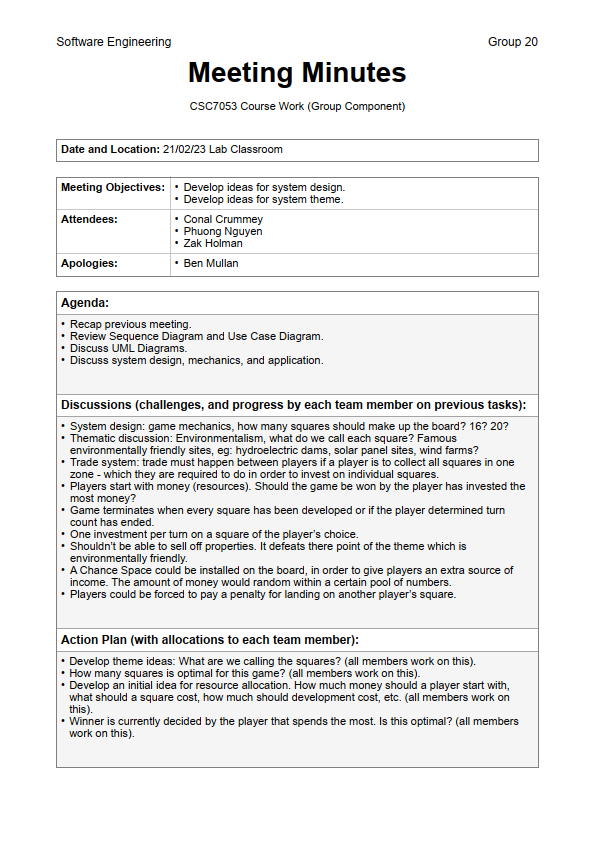
*Inheritance*: We identify five types of Square objects: BlankSquare, ChanceSquare, TaxSquare, DonationSquare and AreaSquare, and design each type as a class inheriting from a general Square class. Each type of Square will have its own implementation of *activate()* to kickstart its event. In a similar manner, we design each chance as a class inheriting from Chance class with an intention to expand and maintain the game’s pool of chances in the future, and let individual chance capture its own attributes to be used for its event.

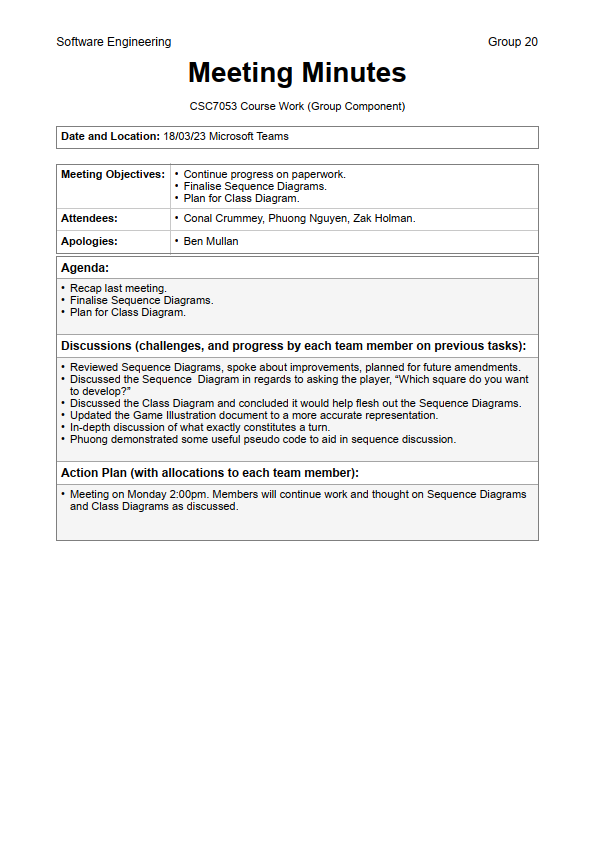
# Appendix I

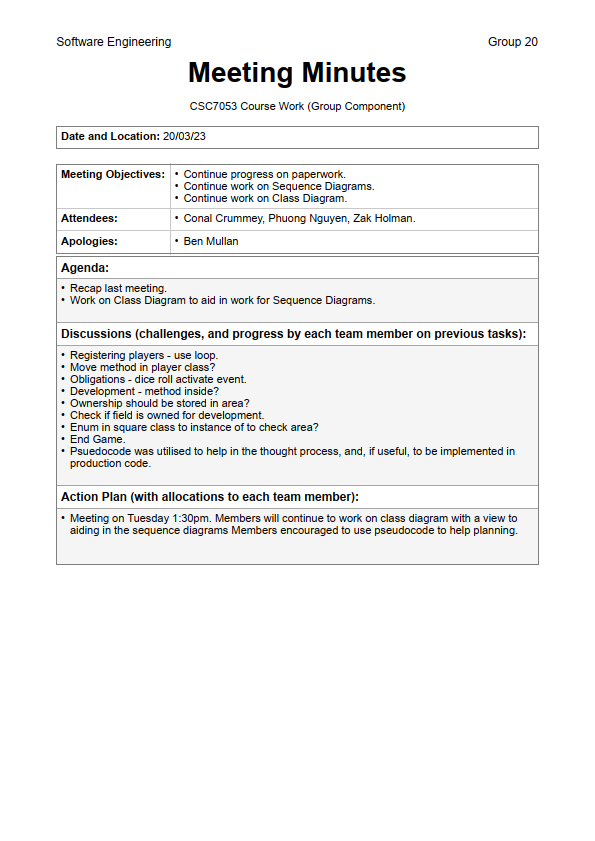
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Use Case Ref** | **Description of Test** | **Test Initialization** | **Test Inputs** | **Test Procedure** | **Expected Results** | **Passed?** |
| 01 | UC01 | Test register valid number of players | Start of the game | 2,3,4 | Enter the number of players | Successful entry. System then prompts to ask for players’ names | Yes - the game registered the correct number of players. |
| 02 | UC01 | Test register invalid number of players | Start of the game | 1, 5 | Enter the players data when prompted | Message displayed prompting the user to enter a valid number of players | Yes - the system displays error message and dives the user another opportunity to enter number of players |
| 03 | UC01 | Test register player names valid | Number of players entered (3 players) | “Zak”, ”Phuong”, ”Conal” | Enter the players name when prompted | Players are given a unique name before the game starts | Yes - Each and every player was given a unique name by the user |
| 04 | UC01 | Test register player names invalid | Number of players entered | “Zak”, “Zak” | Enter the players name when prompted | When two players are given the same name, a message is displayed asking the user to give each player a unique name | Yes - The message was displayed and the user given another opportunity to enter a different name |
| 05 | UC02 | Testing player movement | Players have been registered | Enter key | Player hits enter when prompted | Player’s position is updated. Opportunities are presented based on what square the player lands on | Yes - the player’s position was updated and the correct opportunities were presented. |
| 06 | UC03 | Testing the Dice roll | Players have been registered | Enter key | Player hits enter when prompted | For each dice, a random number is generated between 1-6. The sum of these dice throws returned | Yes - the dice generated, and returned, the correct numbers. |
| 07 | UC04 | Testing the move method’s START functionality | Player position = 18 | Enter key | Player rolls dice and lands on/ passes the START square | The player gains 200 currency, a message is displayed and the move method is not interrupted. | Yes - the player gained the right amount after passing Start, and the relevant info was displayed. |
| 08 | UC05 | Test buy area, player accepts offer | Player with 150 currency lands on “Jiuquan Wind Power Base” area | “1” | Player accepts the offer to buy the area when prompted | The player is given an offer to buy the area they land on with their currency, if the player accepts the offer, the player loses the currency  needed to buy the area and gains ownership of the area. | Yes - the player was able to purchase an area and the game correctly attributed the player’s ownership to the square. |
| 09 | UC05 | Test buy area, player rejects offer | Player with 150 currency lands on “Jiuquan Wind Power Base” area | “2” | Player accepts the offer to buy the area when prompted | The player is given an offer to buy the area they land on with their currency, if the player rejects this offer a message is displayed confirming their decision | Yes - The player was able to reject the offer to buy the area and the correct message was displayed |
| 10 | UC05 | Test buy area, players has insufficient currency | Player with 100 currency lands on “Jiuquan Wind Power Base” area | No user input | No user input | A message is displayed informing the player that they do not have enough currency to purchase the area | Yes - The correct message is displayed |
| 11 | UC06 | Testing the payOwner  method, current player is the owner | Player who owns “Jiuquan Wind Power Base” lands on “Jiuquan Wind Power Base” area | No user input | No user input | A message is displayed to tell the player that they already own the area | Yes - The correct message is displayed |
| 12 | UC06 | Testing the payOwner  method, The player has enough currency to pay the fee | Player1 with 100 currency lands on “Jiuquan Wind Power Base” area owned by Player2 | No user input | No user input | A message is displayed telling the player that they have to pay a fee. The current player loses the currency needed to pay the fee. This currency is then transferred to the owner of the player | Yes - The correct message is displayed an the correct amount of currency is transferred between the players. |
| 13 | UC06 | Testing the payOwner  method, The player does not have enough currency to pay the fee | Player1 with 5 currency lands on “Jiuquan Wind Power Base” area owned by Player2 | No user input | No user input | A message is displayed telling the player that they do not have enough currency to pay the fee. The player is then prompted to sell some of their areas or developments [UC10], if the player cannot gain enough currency to pay the fee, they are knocked out of the game | Yes - The correct message is displayed, the player is prompted to sell and if they cannot sell enough they are knocked out of the game |
| 14 | UC07 | Testing the Receive Donation square. | Player lands on the Receive Government Donation square. | In the game, roll dice to land on the Receive Government Donation square. | Roll the dice to land the player on the Receive Government Donation square. | The player’s balance is increased by 200. | Yes - the player landed on the square and had their balance increased by 200. |
| 15 | UC07 | Testing the Green Tax square. | Player lands on the Green Tax square. | In the game, roll dice to land on the Green Tax square. | Roll the dice to land the player on the Green Tax square. | The player’s balance is decreased by 200. | Yes - the player landed on the square and had their balance decreased by 200. |
| 16 | UC08 | Testing Random Chance Event (Move Backward) | Player lands on the Random Chance Square. | In the game, roll dice to land on the Random Chance square. | Roll the dice to land the player on the Random Chance square. | The player is moved backward between 1-3 squares. | Yes - the player was moved backward 1-3 squares successfully. |
| 17 | UC08 | Testing Random Chance Event (Move Forward) | Player lands on the Random Chance Square. | In the game, roll dice to land on the Random Chance square. | Roll the dice to land the player on the Random Chance square. | The player is moved forward between 1-3 squares. | Yes - the player was moved forward 1-3 squares successfully. |
| 18 | UC08 | Testing Random Chance Event (Purchase an Area) | Player lands on the Random Chance Square. | In the game, roll dice to land on the Random Chance square. | Roll the dice to land the player on the Random Chance square. | The player is offered the chance to purchase an area anywhere on the board. | Yes - player was prompted to purchase an area of their choice. The player successfully bought said square. |
| 19 | UC08 | Testing Random Chance Event (Purchase an Area) - every area is already purchased. | Player lands on the Random Chance Square. | In the game, roll dice to land on the Random Chance square. | Roll the dice to land the player on the Random Chance square. | The player is offered the chance to purchase an area anywhere on the board, but all areas are purchased so the game moves to the next turn. | Yes - player was informed that all areas were already purchased, so the turn ended. |
| 20 | UC08 | Testing Random Chance Event (Purchase an Area with insufficient funds) | Player lands on the Random Chance Square. | In the game, roll dice to land on the Random Chance square. | Roll the dice to land the player on the Random Chance square. | The player is offered the chance to purchase an area anywhere on the board. They have no resources, the game informs the player and asks them to try again. | Yes - the game offered the chance, player had insufficient funds, the game asked the player to try again. |
| 21 | UC09 | Testing the player developing an area (first development). | The game gives the player the option to buy a development. | During the player’s turn, opt to buy the first development. | Opt to buy the first development. | The area now has a development. The game displays that info. | Yes - player bought development. Price was subtracted from resources. |
| 22 | UC09 | Testing the player developing an area (second development). | The game gives the player the option to buy a development. | During the player’s turn, opt to buy the second development. | Opt to buy the second development. | The area now has two developments. The game displays that info. | Yes - player bought second development. Price was subtracted from resources. |
| 23 | UC09 | Testing the player developing an area (third development). | The game gives the player the option to buy a development. | During the player’s turn, opt to buy the third development. | Opt to buy the third development. | The area now has three developments. The game displays that info. | Yes - player bought third development. Price was subtracted from resources. |
| 24 | UC09 | Testing the player developing an area (major development). | The game gives the player the option to buy a development. | During the player’s turn, opt to buy the fourth development. | Opt to buy the major development. | The area now has four developments. The game displays that info. | Yes - player bought fourth (major) development. Price was subtracted from resources. |
| 25 | UC09 | Testing the player developing an area when the max level of developments has already been achieved. | The game gives the player the option to buy a development. | During the player’s turn, opt to buy an illegal fifth development. | Opt to buy the illegal fifth development. | The game stops you from buying a development on that square. It asks you to try again. | Yes - game stopped the illegal fifth development and asked the player to try again. |
| 26 | UC09 | Testing the player’s ability to decline the opportunity to start a Development on an area. | The game gives the player the option to buy a development. | During the player’s turn, opt to not buy the development. | Opt to not buy the major development. | The game recognises the player’s option and moves on to the next turn. | Yes - player opted to not buy a development and the game moved on. |
| 27 | UC09 | Testing the player’s ability to attempt to buy a development with insufficient funds. | The game gives the player the option to buy a development. | During the player’s turn, opt to buy the development. | Opt to buy the major development. | The game recognises the player’s option but informs the player about their insufficient funds and asks the player to try again. | Yes - player opted to buy, but was informed of the insufficient funds. Was asked to try again. |
| 28 | UC10 | Testing the player’s ability to sell a development. | The player lands on an opposing player’s area, or tax area, and is prompted to sell. | The player chooses which development to sell. | The player opts to sell a specific development. | The development is sold, the value of which is added to the player’s balance. The balance then pays the outstanding penalty. | Yes - The development was sold from several areas and the penalty was paid and the game advanced. |
| 29 | UC10 | Testing the player’s ability to sell an area. | The player lands on an opposing player’s area, or tax area, and is prompted to sell. | The player chooses which area to sell. | The player opts to sell a specific area. | The property is sold, the value of which is added to the player’s balance. The balance then pays the outstanding penalty. | Yes - the player is prompted to sell and successfully sells an area to cover the penalty cost. |
| 30 | UC10 | Testing the player’s ability to reject the opportunity to sell a development or property, thus disqualifying them. | The player lands on an opposing player’s area, or tax area, and is prompted to sell. | The player refuses the option to sell a property or development. | The player opts against selling anything. | The player is disqualified. | Yes - the player was disqualified. |
| 31 | UC11 | Testing the End Game in the event that there is a winning player. | Start the game and play it to completion. | Be the last player to have resources. All others have zero. | Play the game to completion until there is a winner. | Game ends. Winner is displayed with game stats. | Yes - the game ended and the results were displayed along with the winner. |
| 32 | UC11 | Testing the End Game in the event that the game is voluntarily ended. | Start the game and play a round. | Manually opt to end the game on the player’s turn. | Play the game and manually end it on the player’s turn. | Game ends. Stats (area info and resources) are displayed. The game ends in a draw. | Yes - the game ended and the results were displayed. |

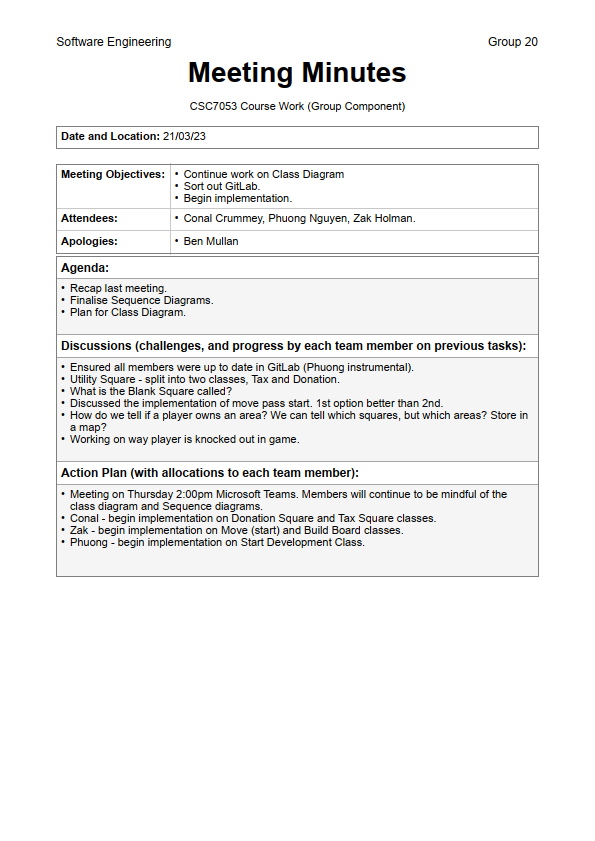
# Appendix II

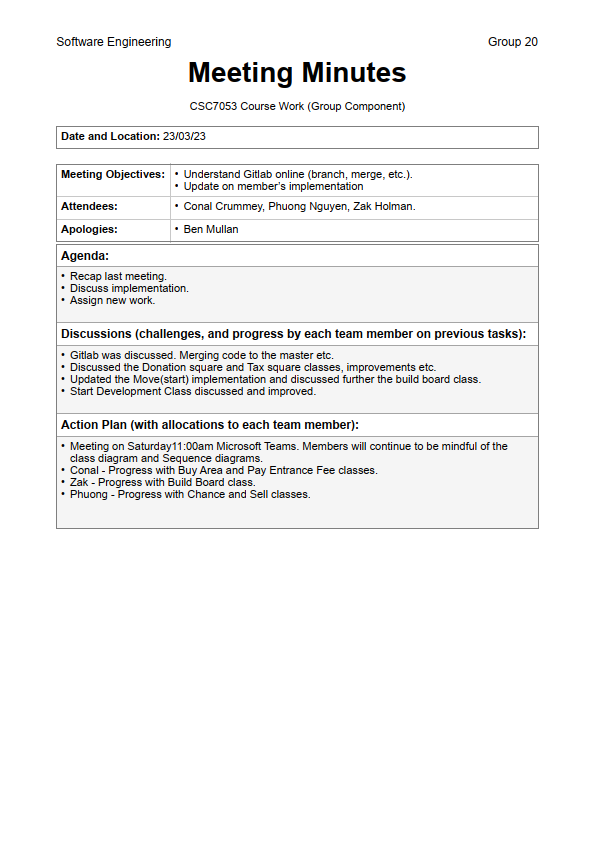


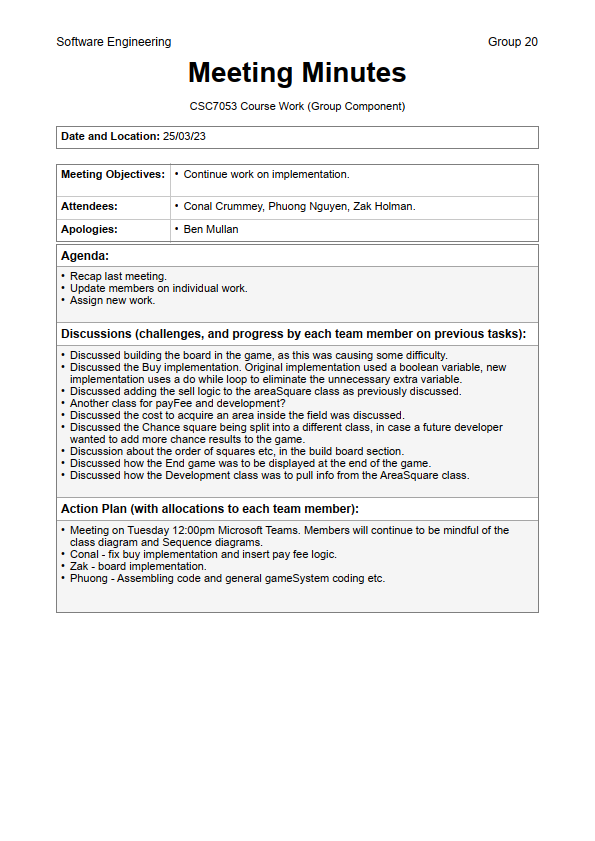


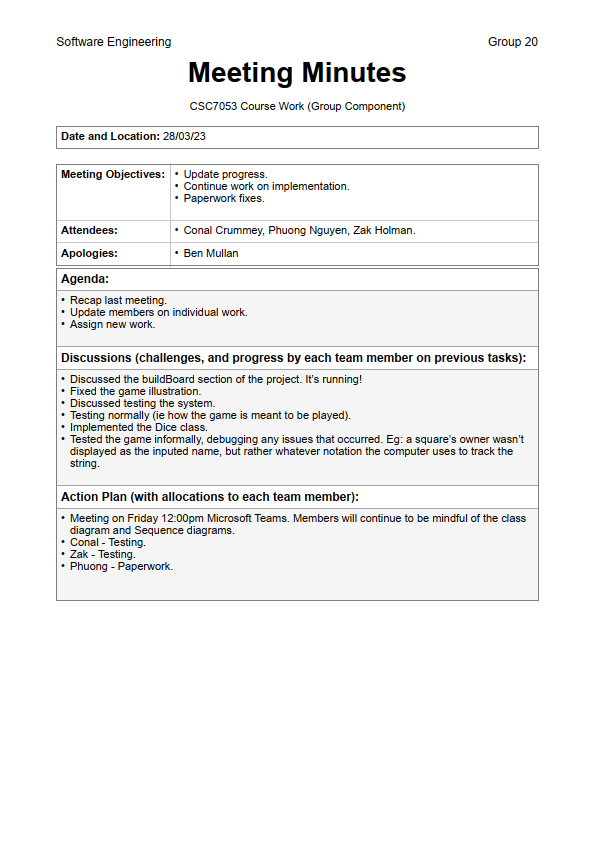


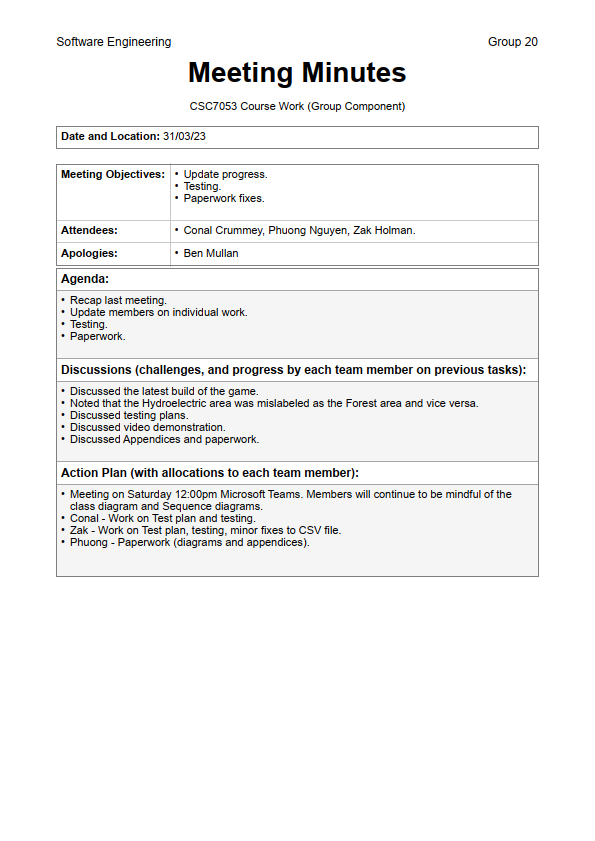


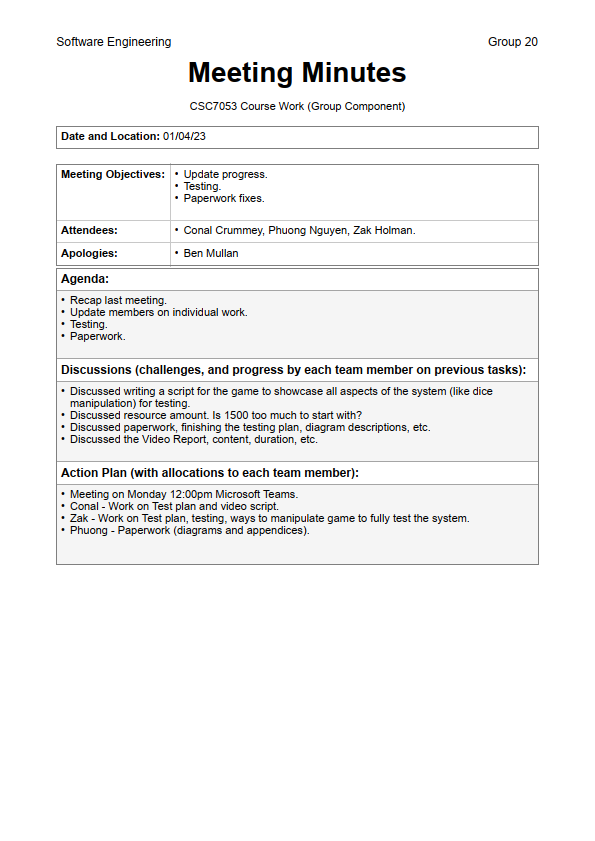


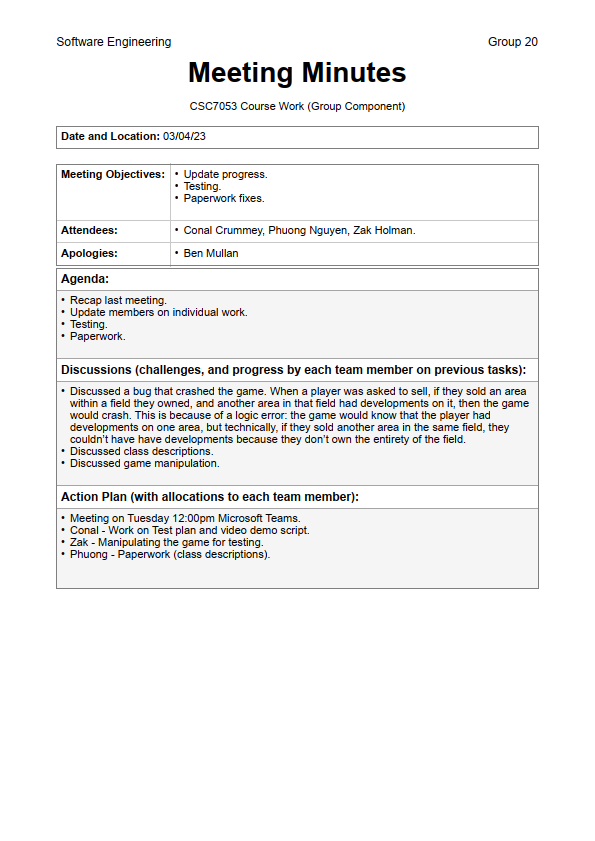


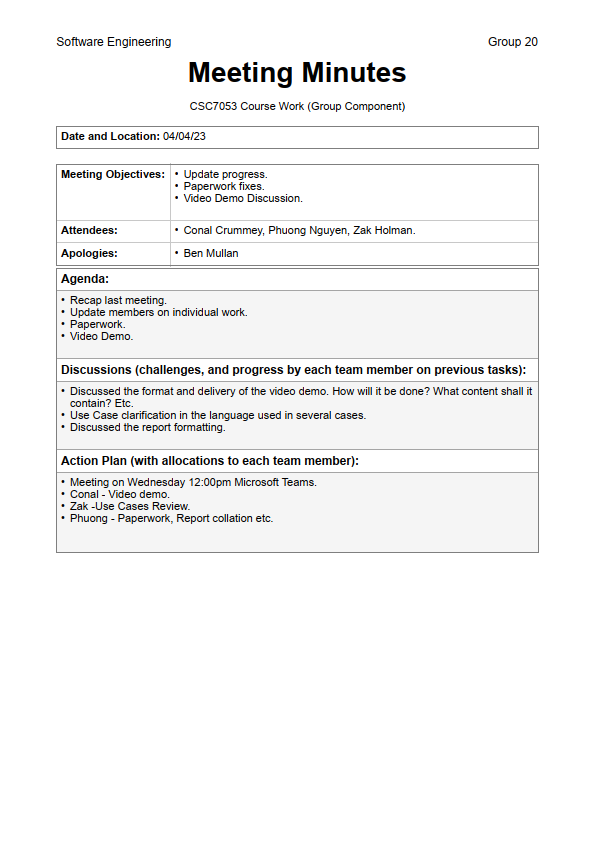


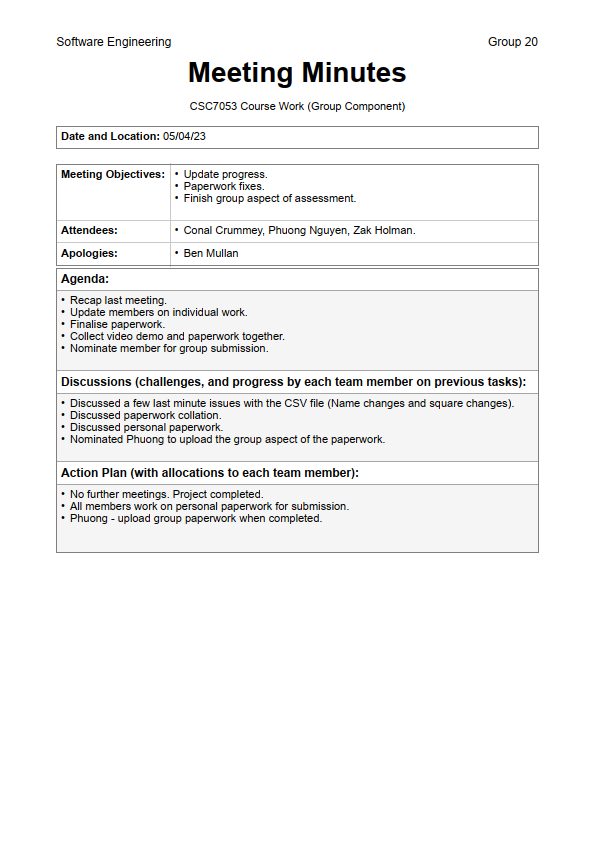












# Appendix III

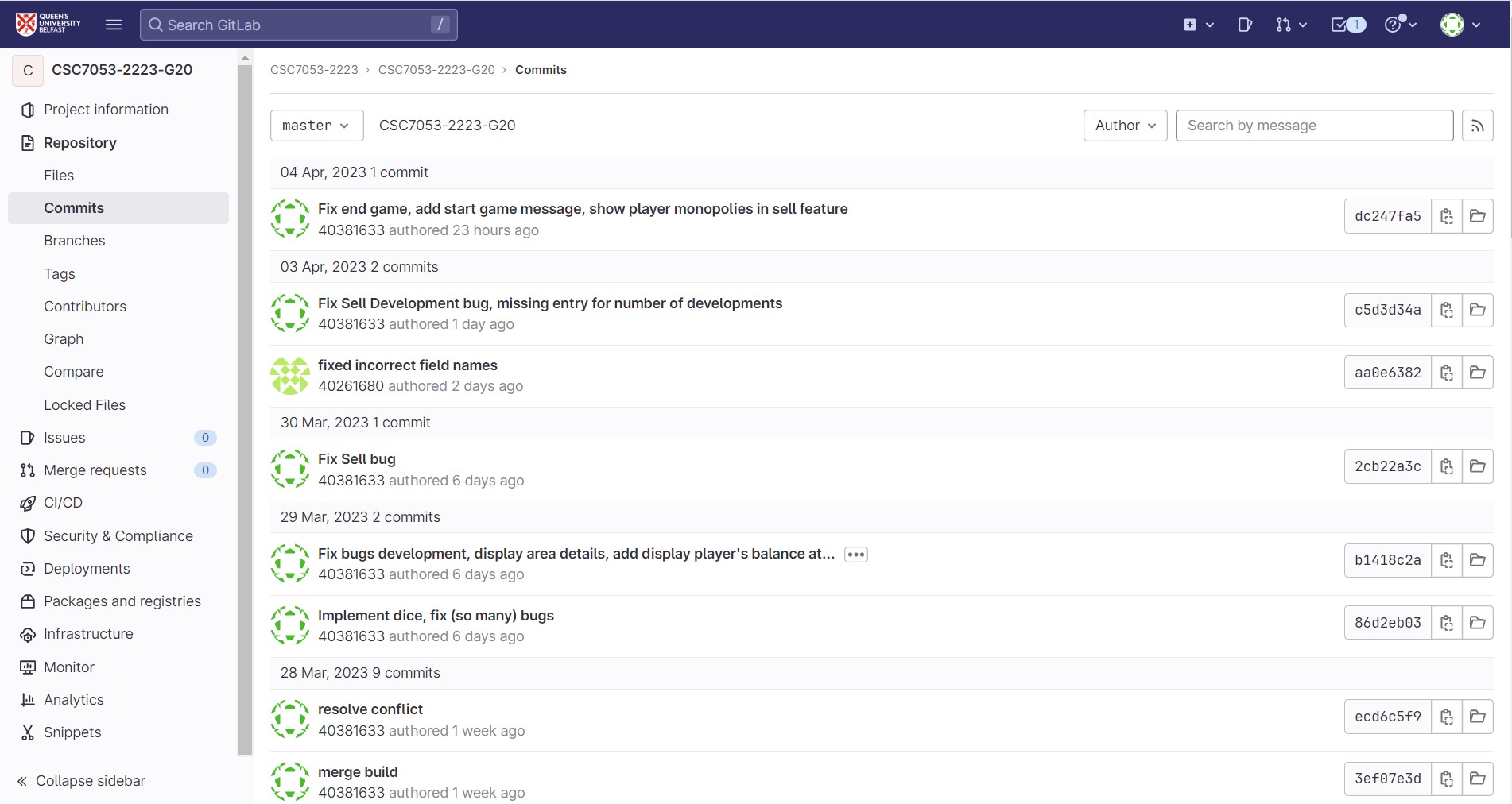


Figure 13 - GitLab Contribution

# Appendix IV

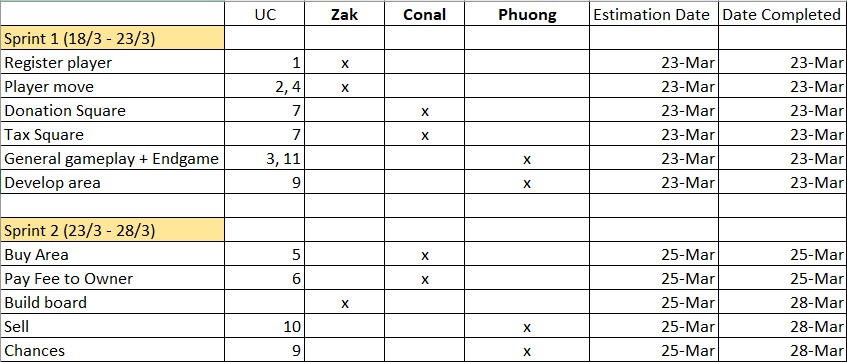


Figure 134 - Sprint Backlog Table