CSC7083 Peer Assessment: Save Our Planet

This Assessment Document is intended to provide you and your assessor with an overview of each group member's involvement delivery of the CSC7083 Project.

Each group should complete one Assessment Document and its content must be agreed by all group members. The completed form should be included at the start of your group's PDF report. ***Don't forget to fill in the Group Number.***
There are three main parts to the Assessment Document – the Evaluation, the Declaration and the Personal Statements (**spaces for each team member's personal statement are provided on the reverse of this sheet**). All parts must be completed – otherwise your group's report will not be marked. Arrange a group meeting to discuss the evaluation and personal statements, and see the note below!

Evaluation	Group Number: 7			
Name	Contribution to team-working and motivation[1]	Contribution to documented analysis, design and testing [1,2]	Contribution to working system code[1,2]	**Peer Score** (Range 85 – 115)
Rebecca Gregory	*5*	5	4	115
Gerard Gargan	4	5	5	115
Patrick Duggan	5	4	5	115
Christopher Gillen	4	5	4	115
Joshua McCann	5	4	3	110

[1]Values for contribution: 1 = Minimal Contribution; 2 = Reasonable Contribution; 3 = Good Contribution; 4 = Very Good Contribution; 5 = Excellent Contribution
[2]This value should consider contributions in the round – direct contributions to required deliverables, and contributions that have made the deliverables possible.

Declaration

"I declare that I have read the Queen's University regulations on plagiarism, and that any contribution I have made to the attached submission is my own original work, except for any elements that I have clearly attributed to third parties. I understand that this submission will be subject to an electronic test for plagiarism and will also be subject to the University's regulations concerning late submission if it is received after the deadline."

Name	Date	Confirmation *(use the words shown in the example below!)*
Rebecca Gregory	*20/04/2024*	*I agree to the terms of the declaration*
Gerard Gargan	*20/04/2024*	*I agree to the terms of the declaration*
Patrick Duggan	*20/04/2024*	*I agree to the terms of the declaration*
Christopher Gillen	*20/04/2024*	*I agree to the terms of the declaration*
Joshua McCann	*20/04/2024*	*I agree to the terms of the declaration*

A note on the Evaluation:
Complete all the columns in the Evaluation Table. The Contribution columns are intended to help team members quantify each other's input to the project, before they award agreed **Peer Scores**. There will not necessarily be a precise correlation between the Peer Score and the Contribution values. However, high Contribution values, as an indicator of the importance of the team member's work to the success of the project, should normally result in a high Peer Score for a team member. Likewise a low Peer Score would be the expected outcome if Contribution values are low. Students who have made a high-value Contribution in all three contribution categories (e.g. 5,5,5) should expect to receive a higher Peer Score than students who have made a lower-value Contribution in one or more categories (e.g. 5,5,3).

If, having reviewed the Contribution values, the team agrees that Team Member 1 made a minimal contribution overall, a Peer Score of 85 would be appropriate for Team Member 1. If Team Member 1's contribution was excellent (critical to the success of the project in all areas of engagement), consider a peer score of 115. If Team Member 1 made a generally good contribution, doing what was expected of them, they could expect to receive a Peer Score of 100. It may be that a team member (for whatever reason) has disengaged from the project entirely, and in such circumstances a Peer Mark of 0 may be acceptable. ***Please inform the module Lecturer if a team member has left your group or has ceased to play an active role in the group.*** Each team member's overall score

for the project will be calculated according to the following formula, where S_i is Team Member *i*'s overall score, P_i is the Peer Score received by Team Member *i*, N is the number of members in the team, and M is the raw mark awarded to the report by the assessor.

$$S_i = \frac{P_i}{\frac{1}{N}\sum_{j=1}^{N} P_j} \times M$$

Any Peer Score within the range 85 – 115 will normally be accepted by the module Lecturer. **However, students are expected to award a range of marks within a team: it is very unusual in a project for everyone to display exactly the same level of ability and**

commitment, and the Peer Scores should reflect this. Be fair: be prepared to recognise someone who has adopted a leading role in the project, and acknowledge the fact that some contributions will be weaker than others. Uniform marks, or marks outside the range 85 – 115, may require that the Team discuss its decision with the module Lecturer, in order to agree a fair distribution of marks. Throughout the project, team members should use appropriately named folders in GitLab to help them co-ordinate their work and maintain a record of their contributions. Where team members cannot agree a distribution, or the distribution is unreasonable, the module Lecturer's judgement will be final.

Personal statement of (enter name):	*Rebecca Gregory*
The following were my most significant contributions to the project (100 words or less):	
Our team worked very collaboratively together, and we all contributed to all areas through weekly in-person meetings and additional Teams meetings. My most significant contributions include: 1) Team management/motivation – booking of meeting rooms Graduate Centre, setting up Whatsapp group, Teams meetings, minute taking, and sprint planning/work allocation. 2) User stories and use case tables – development, iterative improvements, converting into industry standard templates 3) Involvement in four of the sprints – code commits, testing, troubleshooting 4) User acceptance testing - completion of tests, identifying missing tests, creation of user acceptance testing tables 5) Group report writing/amalgamation of group contributions and evidence	

Personal statement of (enter name):	*Gerard Gargan*
The following were my most significant contributions to the project (100 words or less):	
I led the development of the system design and UML diagrams. I also completed two of the sequence diagrams. I was heavily involved in the programming/implementation for several use cases across three sprints. In addition, I helped the other team members with Git, completing demos/tutorials and helping them to get set up and learn the commands. I organised the plan for the group video, assigned each person to their section and edited the video. I also took meeting minutes on several occasions. I wrote the section of the report for the system design.	

Personal statement of (enter name):	*Patrick Duggan*
The following were my most significant contributions to the project (100 words or less):	
We all worked incredibly well as a team, but some of my most significant contributions involved the programming of the game and the commits to Gitlab. I also took charge of the user acceptance testing because I had been heavily involved in the requirements engineering process and was comfortable with the requirements. I took a large chunk of code to show for the video as I was comfortable with it after the user acceptance testing.	

Personal statement of (enter name):	*Christopher Gillen*

The following were my most significant contributions to the project (100 words or less):

- Code for JUnit testing and gathering evidence of successful tests
- Exhaustive User acceptance testing and gathering evidence of successful tests
- During testing, I identified bug in system behaviour for user-story 19 which was fixed.
- Input into the report's testing section.
- In design/UML session, suggesting the Square class was made abstract. This allowed us to take advantage of inheritance/polymorphism etc. to more easily implement the system requirements for the game board.

Personal statement of (enter name):	*Joshua McCann*

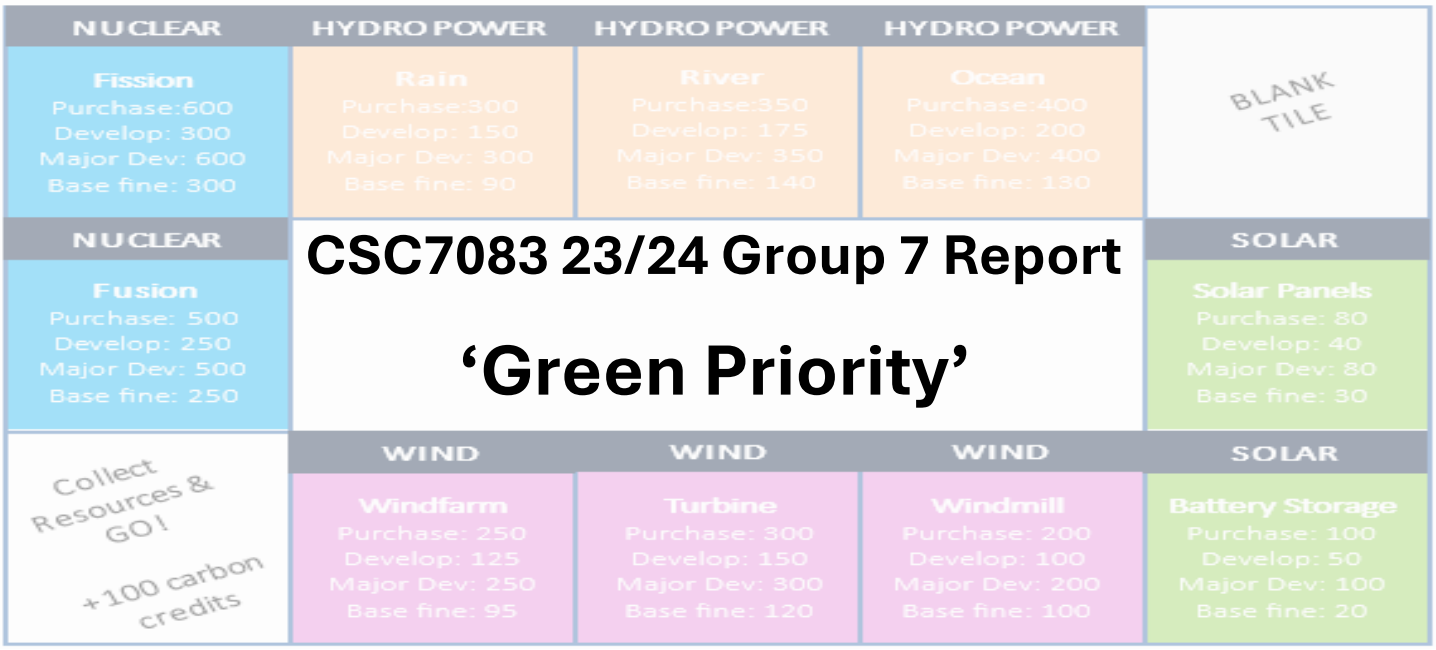
The following were my most significant contributions to the project (100 words or less):

Overall, our team worked collaboratively throughout the project, and everyone provided input to multiple areas. The below are areas I had the most significant input:

I started the Jira group and invited all team members and module Lecturer Janak Adhikari to join. I transferred the user stories into the backlog and added them to their respective epics.

I led the work relating to sequence diagrams. I worked with other team members to ensure we followed the same standard and level of detail throughout all 10 use cases.

I was minute taker for 7 of the team meetings, adhering to the template structure and capturing the main topics covered during the meetings.



GROUP 7 TEAM MEMBERS

Patrick Duggan	40125560
Christopher Gillen	12934038
Gerard Gargan	40061139
Rebecca Gregory	13673076
Joshua McCann	40059274

GROUP 7 VIDEO DELIVERABLE LINK

https://youtu.be/J78uX1P9Md0

Table of Contents **Page Number**

Introduction

Software engineering, i.e. the disciplined systematic approach of design, development, testing and maintenance of software – is of vital importance in this age of digital transformation and rapid advancement. When done correctly, with appropriate methodologies and processes, benefits include: user centric design, adherence to project timelines and requirements, adaptability and flexibility, and quality assurance of the completed product.

The specification of this assignment requested a virtual board game, to be played via the console mode of the IDE – with an environmentally friendly theme, based on caring for the planet.

To facilitate this ask, we implemented Agile methodology to guide and plan our system design and development, and promote collaborative working.

We entitled our project 'Green Priority' – to reflect the environmental theme request, but also to acknowledge our agile methodology and user story prioritisation.

This report will detail our software engineering processes including requirements engineering, system design, agile methodology, working system and associated testing components (JUnit and user acceptance testing). Furthermore, we will discuss some suggested improvements – as although our game met the specification within the requested timelines, we are aware iterative and continued improvement is part of the software engineering process and were keen to consider what future sprints could include.

Requirements Engineering

Requirements and User Stories

Requirements engineering is the process of establishing the services that a client requires from a system, and the constraints under which it operates and is developed.

A requirement itself is a feature that a system must have, or a constraint that it must satisfy, to be accepted by the client. In our case, the assignment specification indicated the constraints and features of the system.

Initially, we reviewed the assignment specification and as a group extracted a list of requirements, adherent to the quality attributes of: unambiguous, testable, clear, correct, understandable, feasible, independent, atomic, consistent, traceable and implementation free. These are available for review in Appendix 1.

After developing the requirements, we generated a list of user stories to expand on the assignment specification requirements. A user story is a brief, informal and clear statement that describes a specific feature from the perspective of the end user, focusing on user experience. An expanded list of user stories is available for review in Appendix 2.

User stories are an important tool in Agile software development, as they are the 'who and why' of user requirements - giving general guidance and description, but no implementation details. They tend to be more 'user friendly' and understandable than a list of requirements, providing a common language of understanding between the provider and client. Although useful to improve engagement, it can however be difficult to determine when user stories are complete enough to cover all essential requirements. To ensure our user stories were of high quality, we confirmed they were independent, negotiable, valuable, estimable, small and testable, as summarised in Table 1:

User Story ID	*Description*
US01	As a player, I want to be able to play a single game with 2-4 human players, so that I can enjoy the game with friends or family.
US02	As a player, I want to be able to enter a unique name consisting of alphanumeric characters at the beginning of the game, so that I can personalise my gaming experience.
US03	As a player, I want to see the rules of the game displayed after players have been allocated, so that I can understand how to play the game properly.
US04	As a player, I want my turn order to be determined by the order in which I entered my name, so that the gameplay remains fair and consistent.
US05	As a player, I want to roll two virtual dice during my turn, each with three sides, to determine the number of spaces I move on the game board.
US06	As a player, I want my position on the game board to be updated after rolling the dice, so that I can see how many spaces I've moved.
US07	As a player, I want to start the game with 1000 credits, so that I have resources to use during gameplay.
US08	As a player, I want to be presented with a list of obligations and opportunities based on the square I've landed on, so that I can make informed decisions during my turn.
US09	As a player, I want to be able to choose one action to complete per turn, so that I can strategically manage my resources and progress in the game.
US10	As a player, if I choose not to purchase a square, it should be offered to other players, ensuring that all squares have the potential to be owned.
US11	As a player, I want to be notified about changes to my resources after completing an action, along with the reason for the change, so that I can track my progress and make informed decisions.
US12	As a player, I expect the game board to consist of 12 squares, each with a specific designation, providing a structured environment for gameplay.
US13	As a player, if I land on or pass the start square, I should gain 100 credits, rewarding me for progressing in the game.
US14	As a player, I expect one of the two field areas to cost the least amount of resources, providing strategic opportunities for investment.
US15	As a player, I expect one of the two field areas to cost the most amount of resources, adding depth to the strategic decision-making process.
US16	As a player, if I own or control the whole field, I should be able to develop an area within that field, creating a development and potentially increasing its value.
US17	As a player, if I own or control the whole field and have established three developments in an area, I should be able to complete a major development, enhancing its value further.
US18	As a player, if I land on another player-controlled area, I should pay credits based on the value of resources in that area, ensuring fair and balanced gameplay.
US19	As a player, if I run out of credits, I should leave the game, and my areas should revert to undeveloped spaces, providing opportunities for other players.
US20	As a player, if I choose to stop playing, the game should end, concluding the game for all players.
US21	As a player, I expect a game statistics summary to be produced at the end of the game, including the amount of credits each player holds, providing insights into our performance.

Table 1: User Story Summary

After creating the user stories, we met in person to discuss their importance in relation to delivering a functional game that met the specification, and allocate their priority in a user story map (see Figure 1 – larger version for viewing available in Appendix 3). Our user story map gave a visual representation of the overarching themes, epics and associated user story priorities, providing the basis of our sprint allocation.

Figure 1: User Story Map

As a team, we set up multiple meetings with module lecturer Dr Janak Adhikari, simulating client/stakeholder meetings, to discuss and resolve any queries, and for 'show and tell' sessions – an important agile practice.

Understanding the importance of utilising agile methodology in our project, we proceeded to implement Jira project management software, to assist in sprint planning, work allocation and collaborative teamworking.

We initially created a project called CSC7083-Team7, and started by inserting all our user stories, with priorities and epics allocated as per our user story map. In planning sprints, user stories were assigned to team members to implement in line with system modelling plan. The backlog indicating the epics and user stories can be viewed in the Jira software project.

Overall, we concluded the requirements engineering phase of the project with a diagram of the game board we intended to implement through the console game (see Figure 2 – larger version available in Appendix 4).

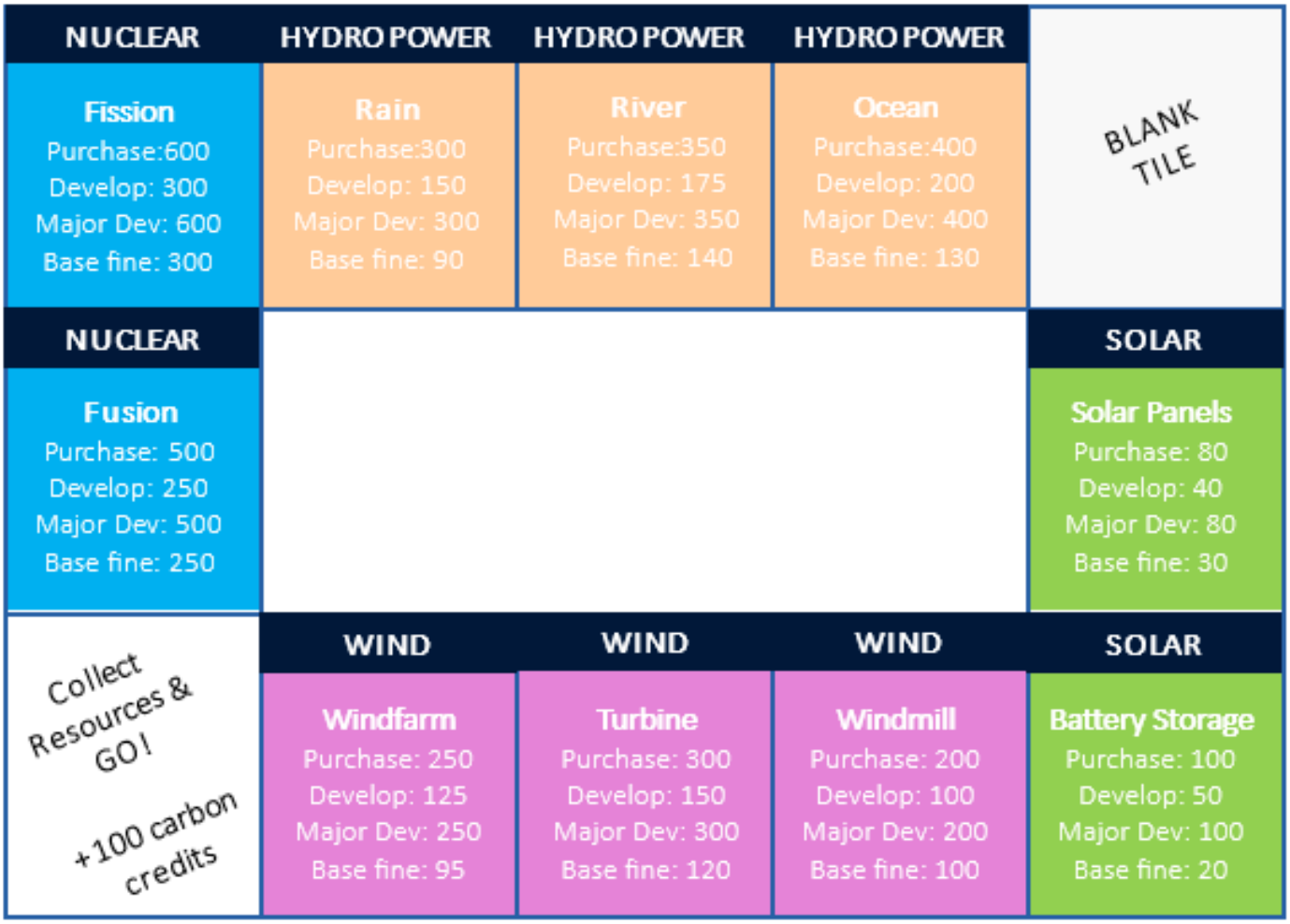


Figure 2: Virtual Game Board

Use Cases

Use cases are a method to show interactions between users and a system within Unified Modelling Language (UML). UML is a set of different diagram types that may be used to model software systems currently on version 2.5 released December 2017[i].

A use case details the behaviour of a system or part of system including the sequence and variation of actions that result in an observable output to a user. They are used to capture the intended behaviour of the system being developed without having to specify how it will be implemented. Use cases consist of a structured text and summarised with a diagram. The use cases cover all the possible interactions described within the requirements.

A single use case can cover multiple user stories. Typically, the use case covers a high-level functional aspect of the system such as registration or taking a turn.

Use cases were developed based on the user stories being grouped into different events that made logical sense covering a high-level functionality.

As indicated by our Use Case Diagram below (Figure 3 – large scale version available Appendix 5), the relationships between use cases help organise behaviours and interactions to reduce repetition. An include relationship between use cases means that the base use case has to incorporate the behaviour of another use case that it points towards, whereas an extend relationship between use cases means that the base use case optionally incorporates the behaviour of another use case that it points towards. The system can exist with or without the extended relationships.

Please see Appendix 6 for our use case tables, which more extensively describes the objectives, actor(s), pre-conditions, main flow, alternative flow and post conditions for each use case.

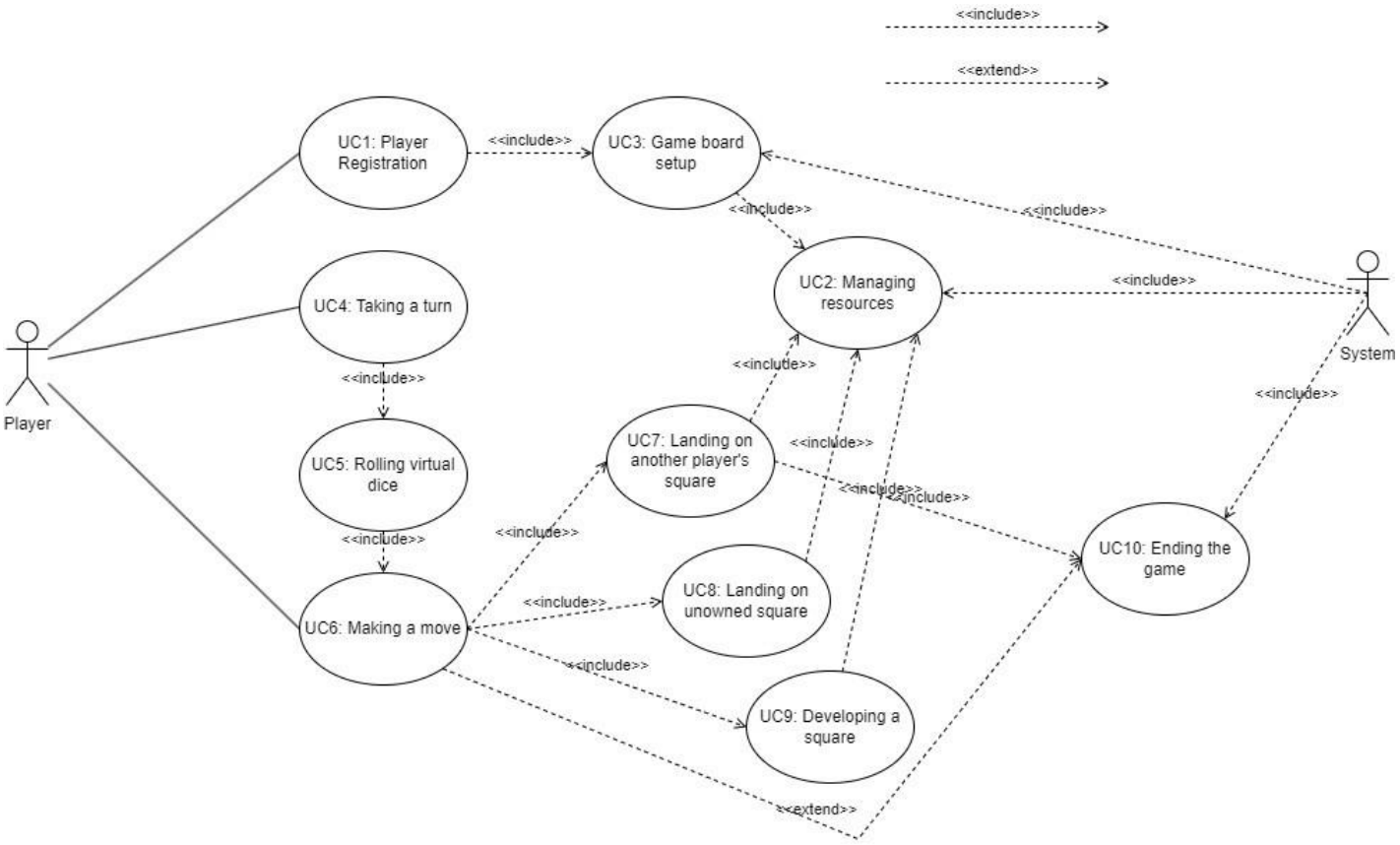


Figure 3: Use Case Diagram

System Design

We adopted an object-oriented approach in designing the system, which enabled us to model the game effectively.

Domain Diagram:

We developed a domain class diagram (Appendix 7) to better understand and visualise the important conceptual aspects of our system structure and form the basis of our more detailed UML class diagram.

UML Class Diagram:

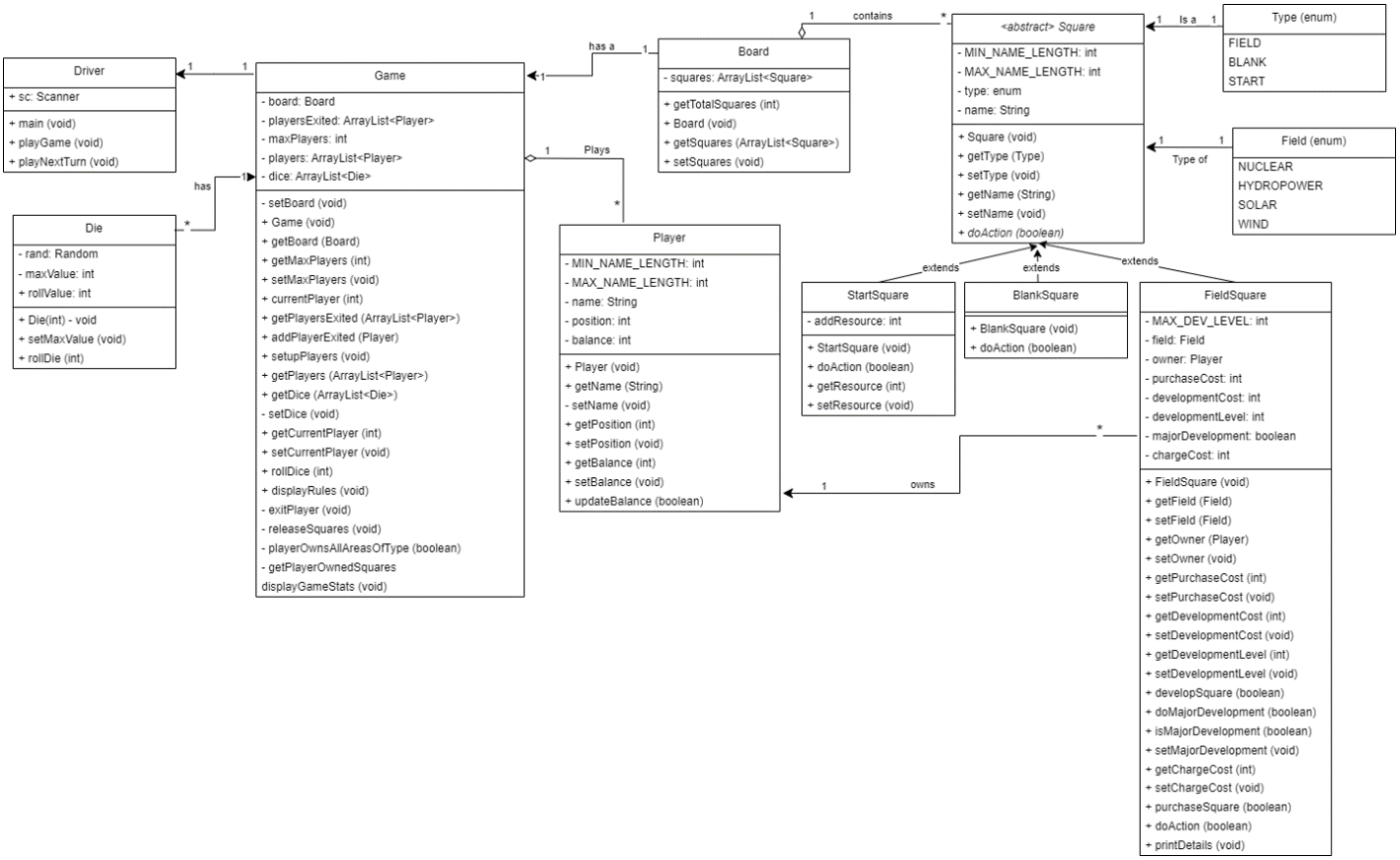


Figure 4: UML Class Diagram

Please see Appendix 8 for a larger scale version of our UML Class diagram.

Key classes such as **Game**, **Player**, **Board**, **Die**, and **Square** (abstract) were incorporated into the design. The driver class is where the program starts and where the game, squares, and die are instantiated. The game rules are displayed and the ***setupPlayers()*** method is invoked. The game will ask the user to input the number of players and will check this against the maximum and minimum number of players. Providing this is valid, it will prompt each user to enter their name, checking for duplicate names and min/max length criteria and re-prompting where invalid input has been entered. This class also houses the main logic for playing the game and managing turns.

The **Square** class was deliberately designed as abstract, serving as a template for three child classes: **StartSquare**, **BlankSquare**, and **FieldSquare**. This design choice facilitated the use of polymorphism, allowing us to define distinct implementation details and behaviors for each type of square.

By storing all squares in an ArrayList of type **Square** in the board class, we were able to take advantage of polymorphism and dynamic method dispatch. When a player lands on a square,

the ***doAction()*** method is invoked. Thanks to polymorphism, the specific implementation of this method for the corresponding child class is executed at runtime, enhancing the game's extensibility. For example, if a player lands on a **BlankSquare**, it will do nothing other than display a message. If they land on a **FieldSquare**, the player may be prompted to purchase the field, or pay a fine depending on the games current state. To add new squares of any type of the board, they can easily be instantiated and added to the ArrayList of Squares in the **Board** class in the order that they should appear on the board.

We incorporated several features into the game to ensure easy extensibility. Adjusting the maximum number of players is as simple as modifying the **maxPlayers** variable within the **Game** class. Similarly, the starting balance of players and their initial position on the board can be customised via parameters in the **Player** constructor.

To designate a **FieldSquare** to a specific field, we introduced a property named Field in the **FieldSquare** class, implemented as an Enum. Adding or modifying available fields is straightforward, as it involves simply updating the Enum class.

In our design, we established that a **FieldSquare**'s development level starts at Level One and progresses to Level Four, with the possibility of a major development upon reaching Level Four. Each development increment incurs an increase in the fine's cost.

We determined that the game concludes when all but one player exhaust their carbon credits. To handle this scenario, we implemented a method within the **Game** class that removes a player from active play upon depletion of their carbon credits. The player's properties are returned to the board at development level one, and their data is transferred to a separate ArrayList named **PlayersExited**, facilitating end-of-game statistics tracking.

Our design approach followed an agile methodology. Initially, the game supported only two dice with a maximum roll number of six. However, we enhanced the game's flexibility by enabling support for any number of dice. Additionally, the maximum roll number can now be adjusted via constructor parameters when instantiating a new **Die** object. These dice can then be added to the **Game**'s ArrayList, and the ***rollDice()*** method efficiently handles rolling all dice and returning the sum of their results.

Sequence Diagrams:

We developed sequence diagrams for each of our use cases (Figures 5 – 14), following standardised UML notation, to provide an overview and better understanding of the interactions and behaviour of the objects in our proposed system.

This helped us visualise and analyse the system in more depth, validating that the behaviours and interactions would be as anticipated. This gave us confidence in planning our sprints that all team members understood the intended behaviours of the system.

Please see Appendix 9 for larger scale version of our sequence diagrams.

Figure 5: UC01 Sequence Diagram

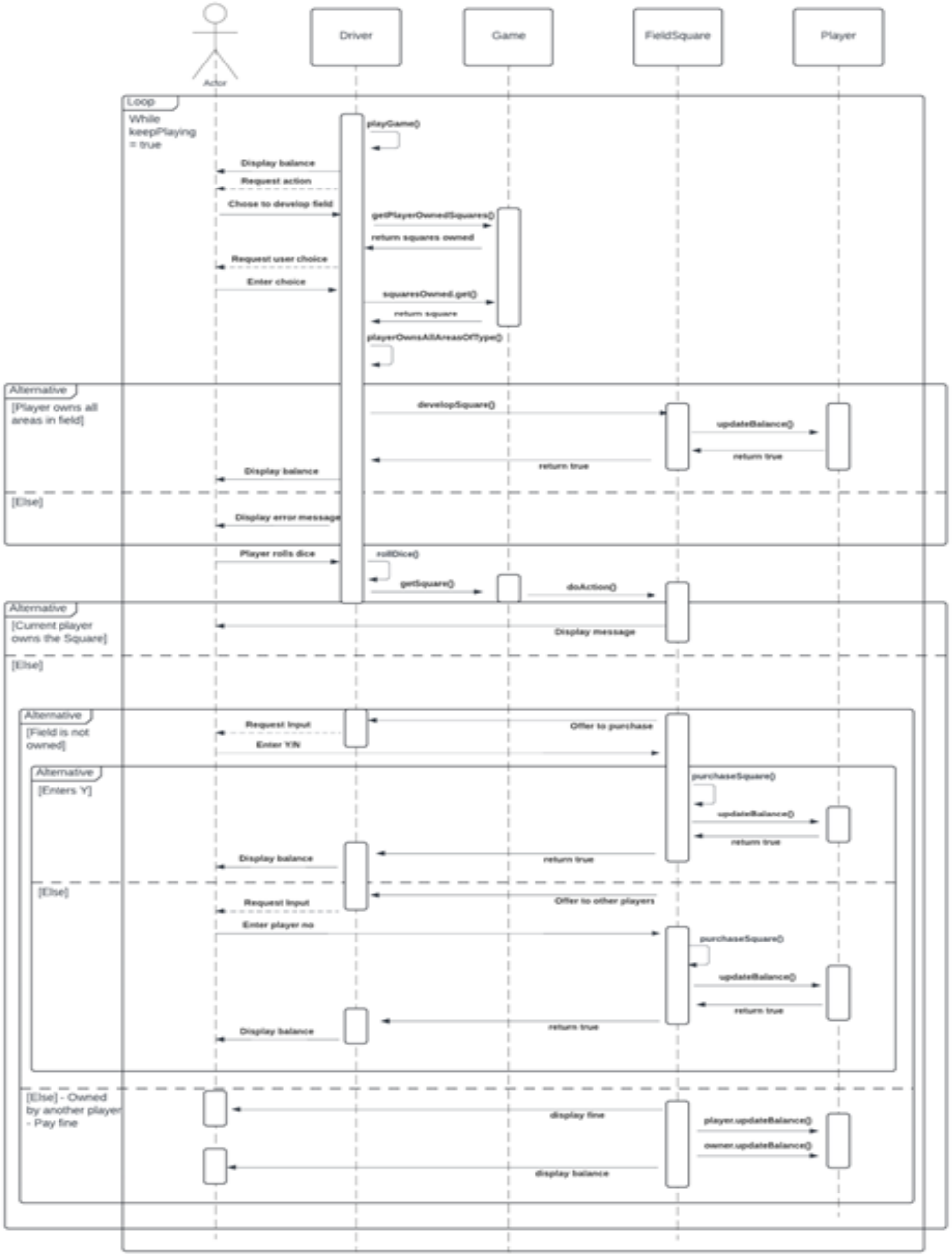


Figure 6: UC02 Sequence Diagram

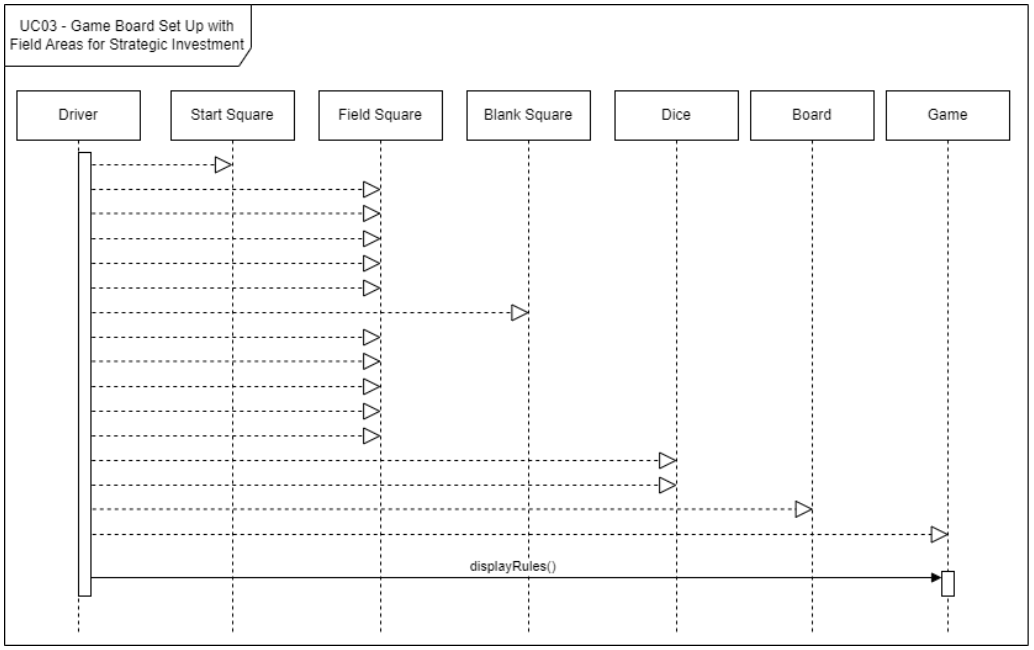


Figure 7: UC03 Sequence Diagram

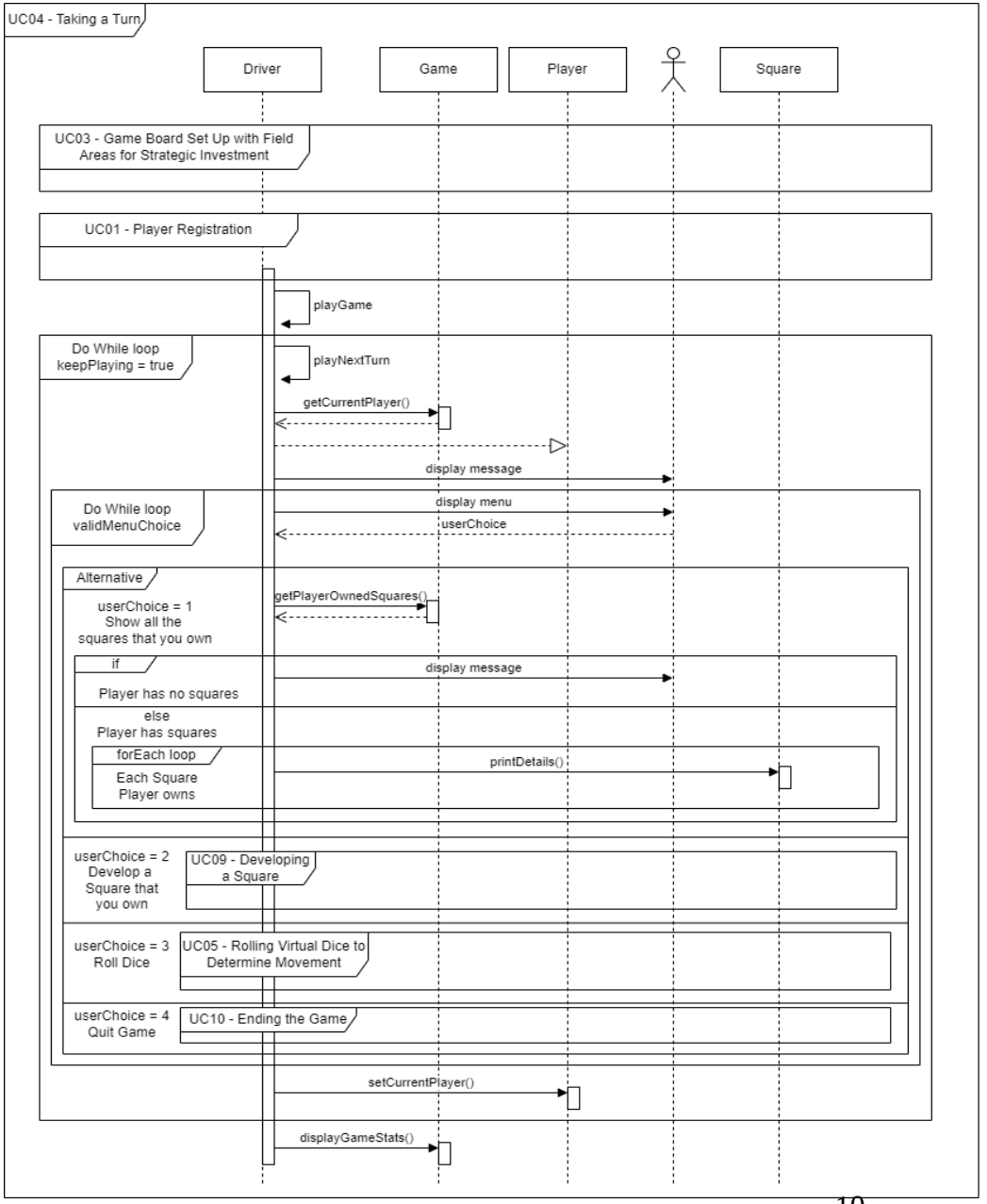


Figure 8: UC04 Sequence Diagram

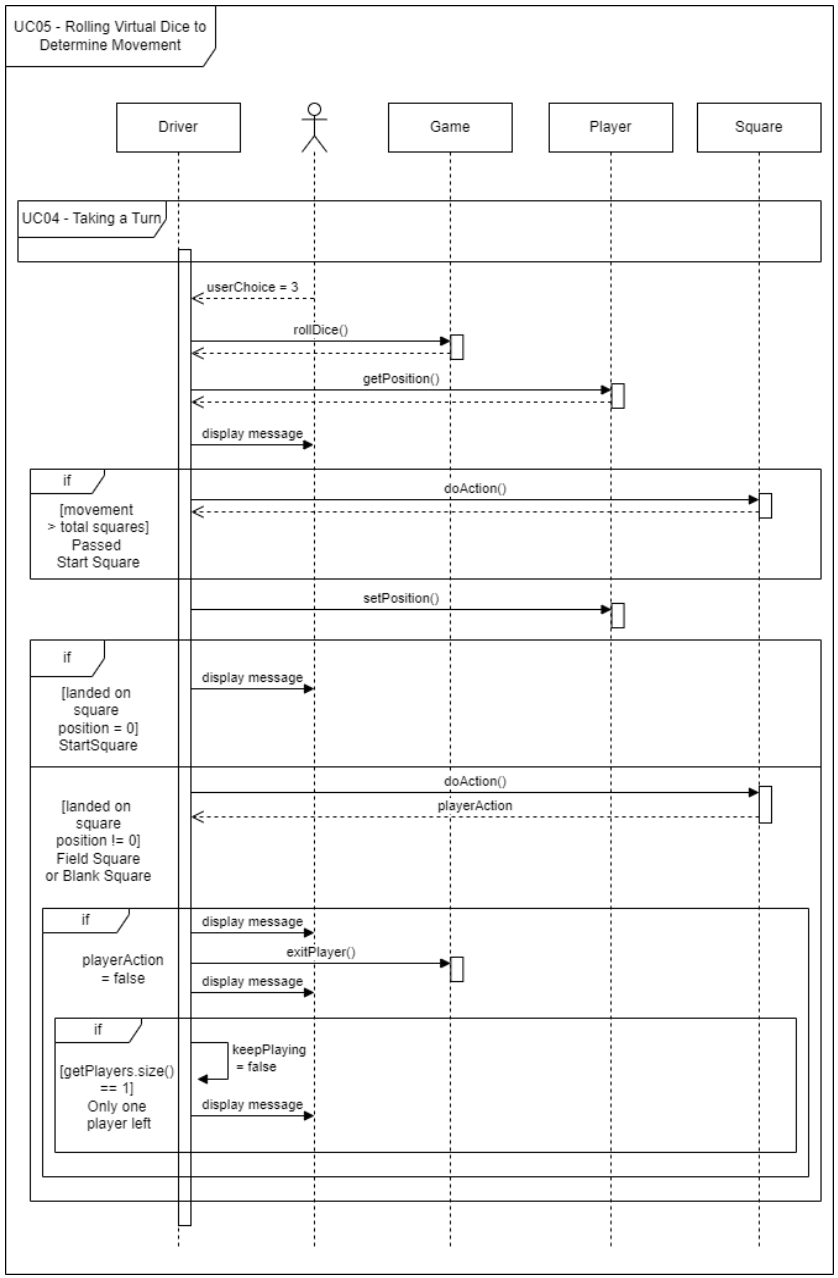


Figure 9: UC05 Sequence Diagram

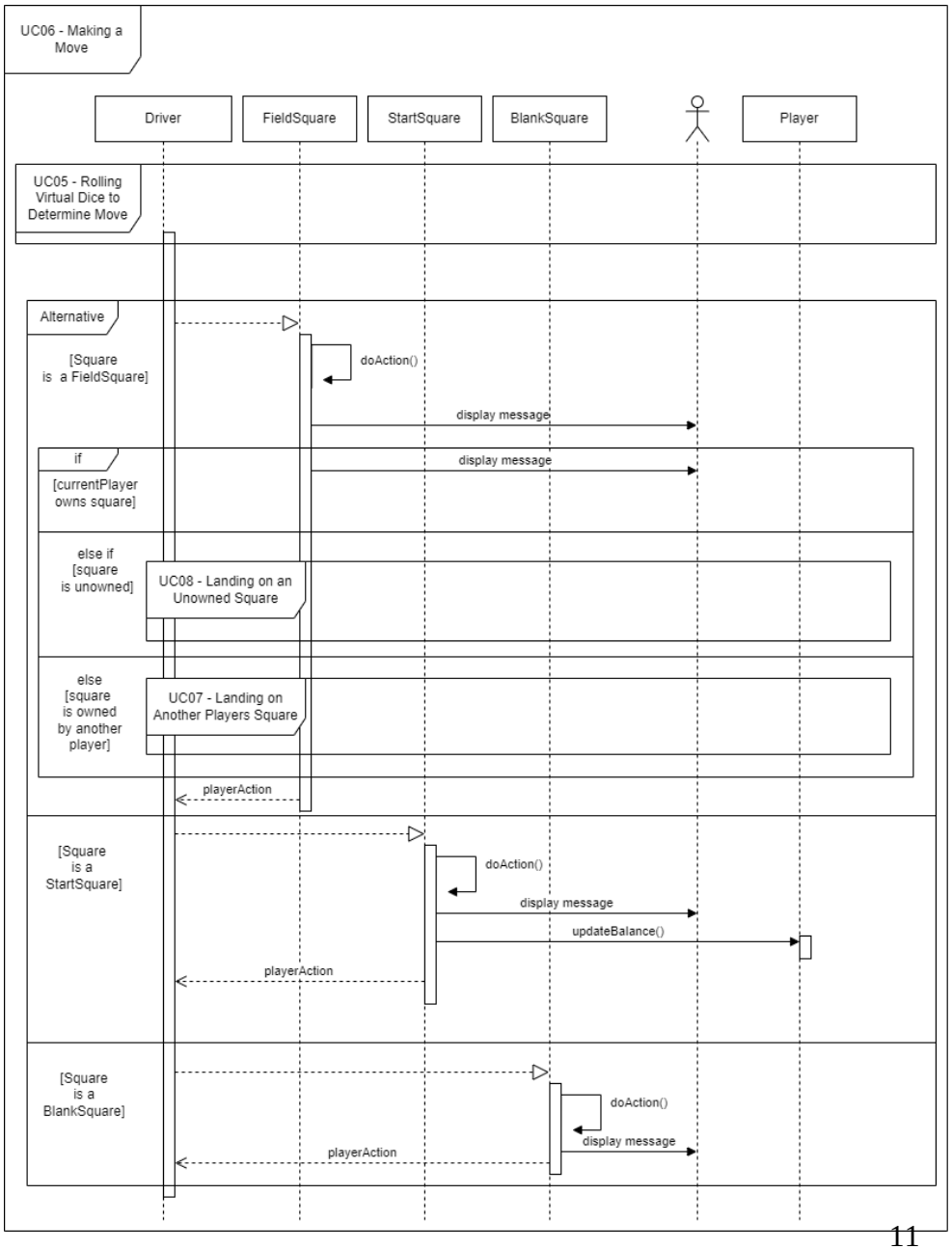


Figure 10: UC06 Sequence Diagram

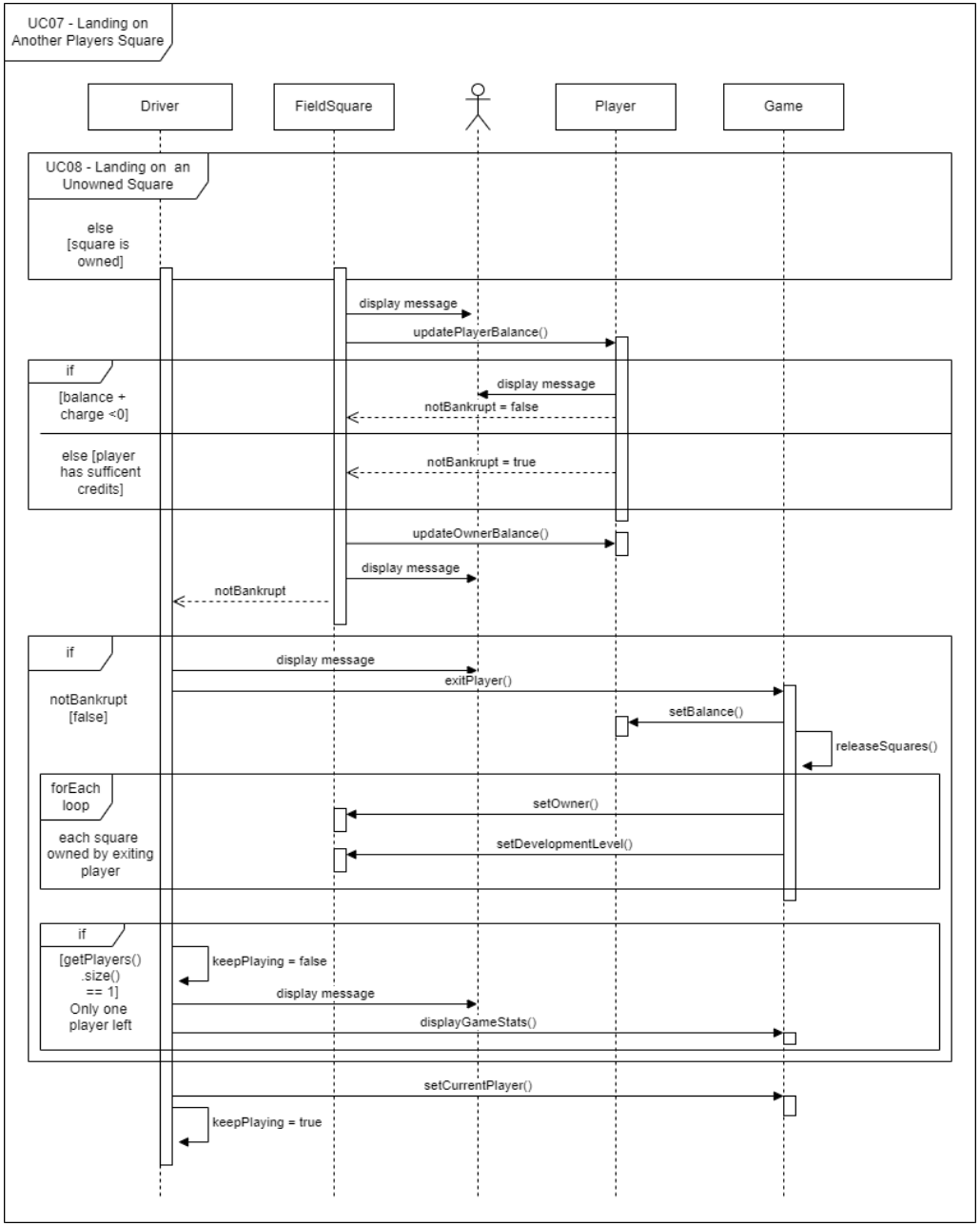


Figure 11: UC07 Sequence Diagram

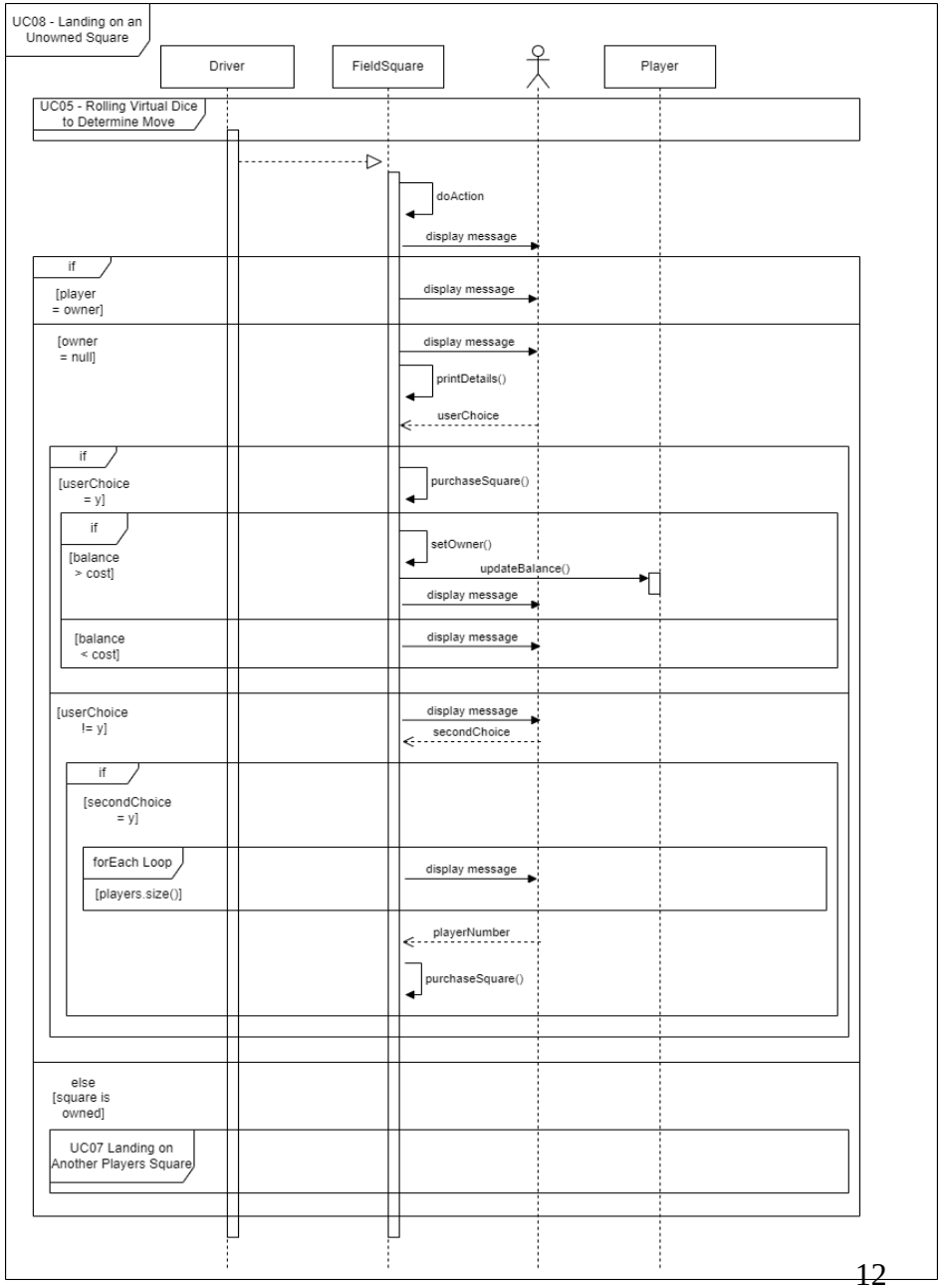


Figure 12: UC08 Sequence Diagram

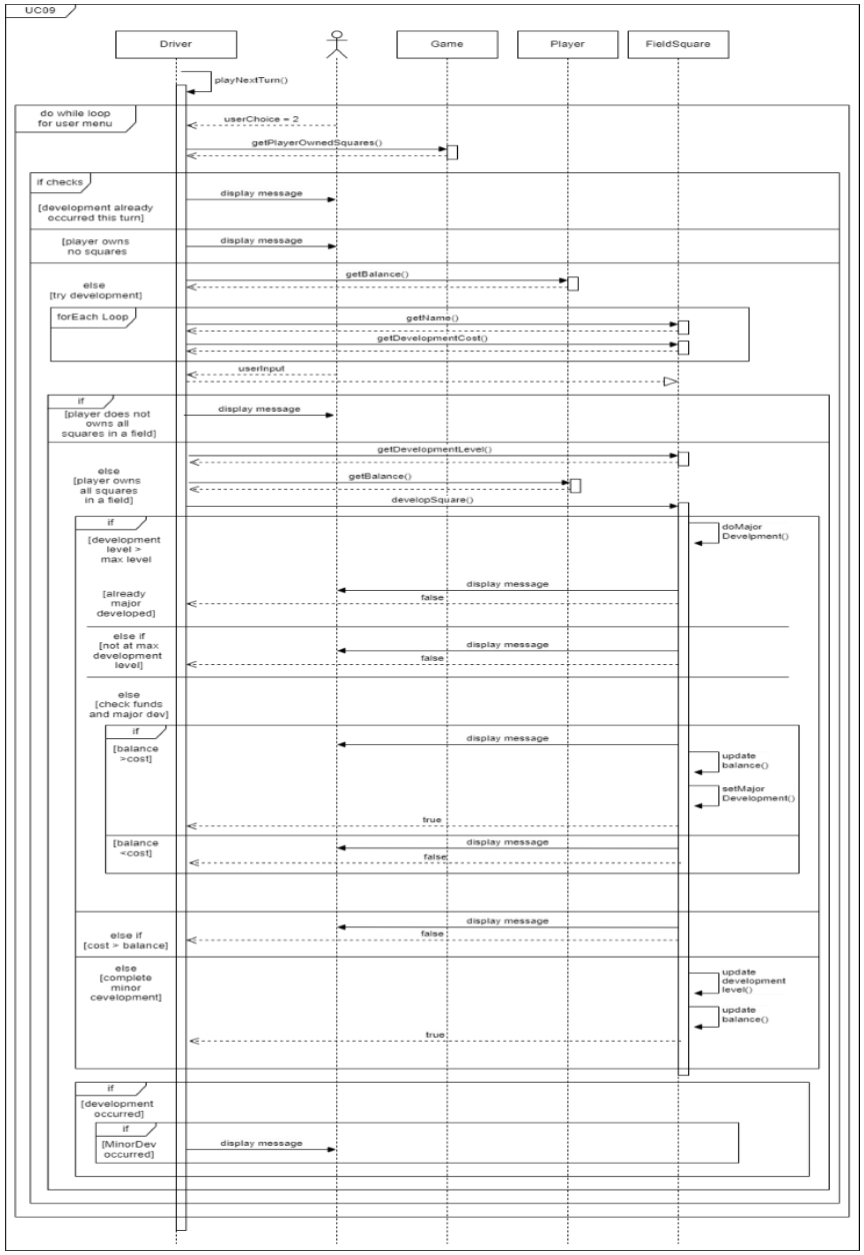


Figure 13: UC09 Sequence Diagram

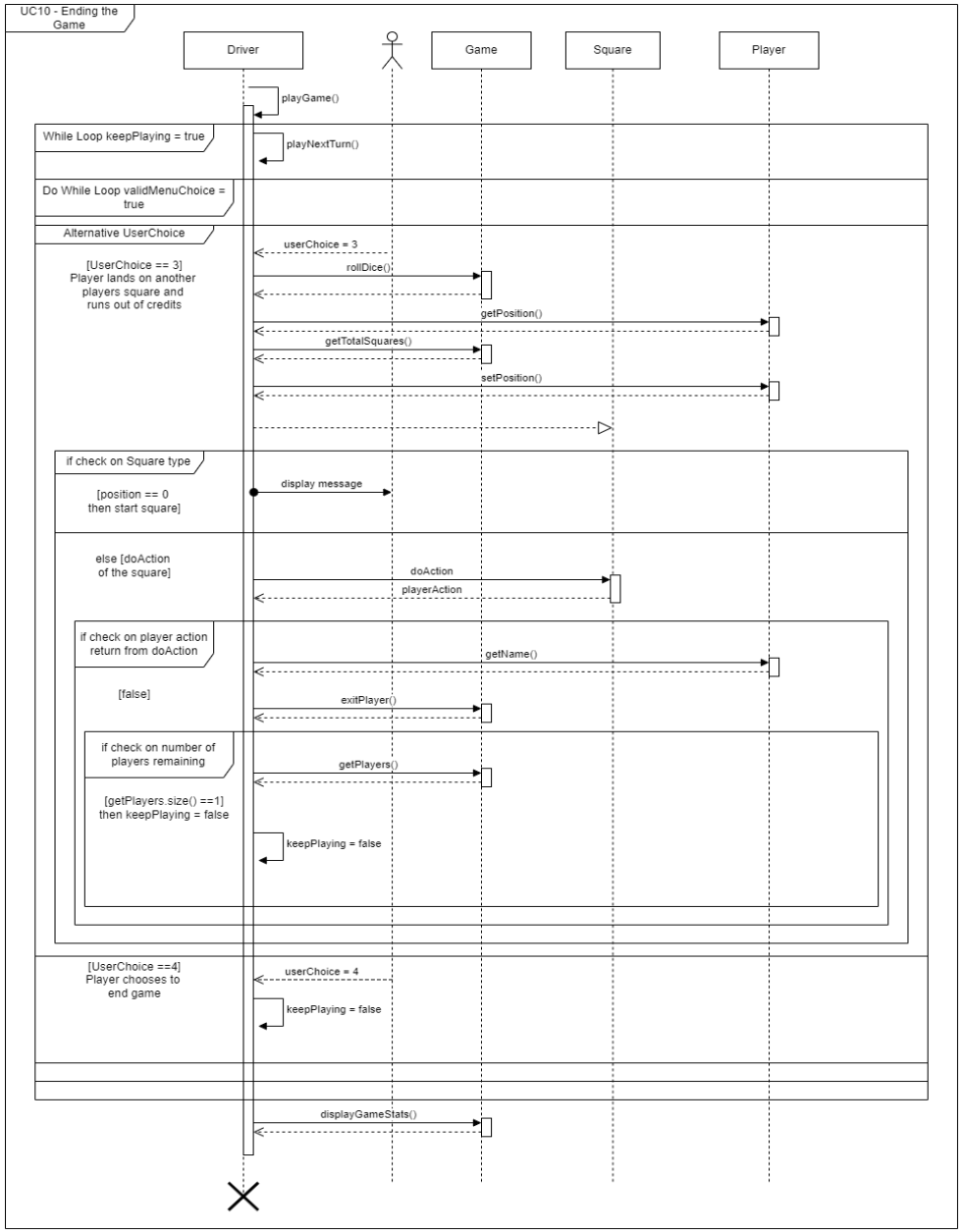


Figure 14: UC10 Sequence Diagram

Agile Iterations

Sprint 1: Identifying potential user requirements

Figure 15: Sprint 1 Jira burndown – requirement gathering

The first sprint we carried out was a simple requirement gathering exercise. Each group member was tasked with reviewing the specification and producing what they believed to be the user requirements of the game to be developed. The aim of this sprint was to be able to collate all our requirements into one encompassing set of user requirements. A relaxed deadline of two weeks was set to allow everyone time to review the documentation and to familiarise themselves with the software.

During the sprint, we had several meetings and online discussions, with each team member sharing their ideas and collaborating with each other to gain a better understanding of what was expected.

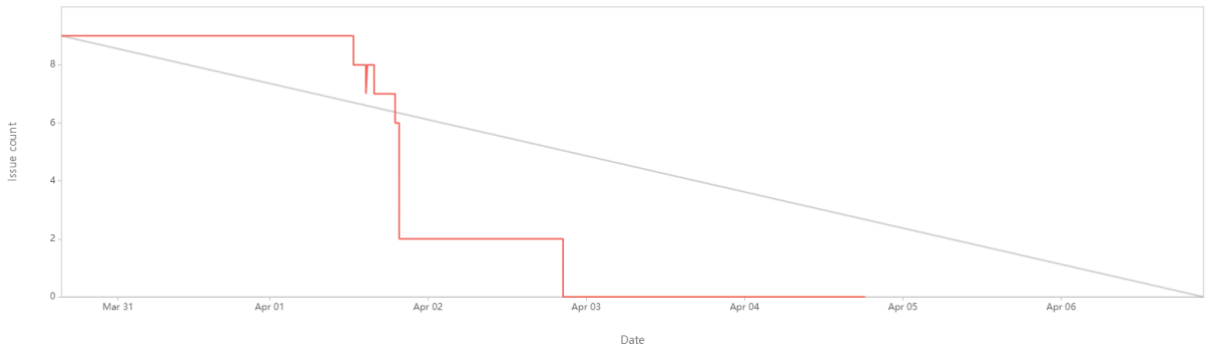
Once the sprint was over, we reached a consensus on our user requirements. These were then used to build our user stories, which formed the basis of the game. The requirements were iteratively reviewed to ensure they remained relevant and appropriate for the specification.

This initial information gathering sprint exercise emphasised the importance of Agile methodology and thorough proactive sprint planning.

Sprint 2: High Priority User Stories

Although technically our second sprint, this was our first sprint in terms of game development – where we focused on implementing the high priority user stories. This is because they made up the core functionality of the game and therefore it was imperative we coded this first to ensure we had a minimum viable product for the client.

We planned an equitable split of each sprint into several user stories per person, with the use of branches within GitLab and one person committing code at a time, so as not to inadvertently create any merge conflicts. We gave ourselves a deadline of one week to complete this sprint as we anticipated it would be the most time consuming and challenging part of the build. This involved collaboration between the group about how each class should be coded and use of our UML diagram.



Completed issues

View ir

Key	Summary	Issue type	Epic	Status	Assignee	Issue count
CT7-14	US01: Allowing 2-4 human players to play a single game	Story	E1: SETTING UP THE GAME E...	DONE	GG	1
CT7-15	US02: Entering unique player names	Story	E1: SETTING UP THE GAME E...	DONE	GG	1
CT7-19	US06: Updating player position on game board	Story	E2: PLAYING THE GAME	DONE	PD	1
CT7-21	US08: Presenting obligations and opportunities based on landed square	Story	E2: PLAYING THE GAME	DONE	CG	1
CT7-22	US09: Choosing one player selected action per turn	Story	E2: PLAYING THE GAME	DONE	PD	1
CT7-31	US18: Paying credits for landing on another player controlled area	Story	E3: MANAGING DEVELOPME...	DONE	CG	1
CT7-32	US19: Exiting the game due to insufficient credits	Story	E3: MANAGING DEVELOPME...	DONE	PD	1
CT7-25	US12: Designing game board with 12 squares and four separate fields	Story	E4: GAME BOARD DESIGN A...	DONE	PD	1
CT7-33	US20: Ending the game when a player chooses to stop playing	Story	E5:TURN MANAGEMENT AN...	DONE	PD	1
CT7-18	US05: Rolling virtual dice to determine movement	Story	E2: PLAYING THE GAME	DONE	PD	1

Figure 16: Sprint 2 Jira burndown – High Priority

There were proactive discussions within the group throughout this sprint to clarify queries and update any build that was not working as anticipated.

The main issue we identified during this first sprint was not creating a standardised naming convention for branches that team members were working on from the offset. Moving forward onto the next sprints we realised that creating branches named after the current user story being implemented would be hugely beneficial to us as we could see what is currently being worked on, and it gave better oversight in tracking the development of the code alongside each user story.

At the end of this sprint, we had created the basis of the game, implementing the main classes and interactions between classes as indicated by our UML diagram. We had the integral components of the game coded and did basic user acceptance testing during development to ensure outputs were as anticipated. At the end of the sprint period, we reviewed the code as a group to ensure we were happy with the outputs and that the sprint was completed satisfactorily.

Sprint 3: Medium Priority User Stories

Our approach to addressing the medium priority user stories within our sprint planning was guided by their collective impact on enriching the gameplay experience. These user stories, spanning various aspects of player interaction and strategic decision-making, formed the focal points of our development efforts, aimed at expanding the game's depth and engagement. We set a deadline of three days for this sprint as we were getting more comfortable with coding and this sprint would be less complex than the last sprint. Allocation was discussed and agreed

between team members, and we had detailed discussions evaluating the expected code outputs for each user story.

Similar to our first sprint, we had proactive collaborative discussions if the build was not proceeding as anticipated, and came to a solutions as a group when issues arose. Again, we completed basic user acceptance testing during this sprint to ensure functionality was as anticipated.

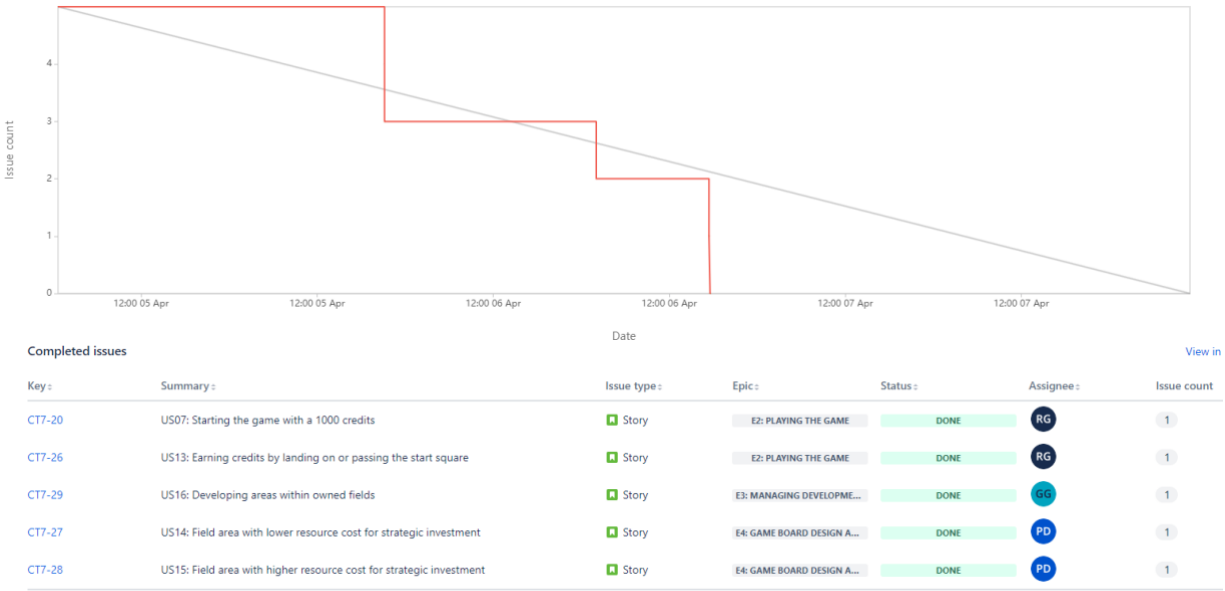


Figure 17: Sprint 3 Jira burndown – Medium Priority

Upon review of the sprint, we had successfully added more features which enhanced the gameplay experience and was working to the agreed standard. We concluded that we had collaborated very efficiently to correct issues in a timely manner and were becoming more cohesive unit, and formally agreed to incorporate peer reviews and updates via electronic communication into future sprints.

Sprint 4: Low Priority User Stories

Our approach to addressing the low priority user stories within our sprint planning was related to them focussing primarily on functionalities aimed at improving player experience, that were not essential for the game to be played. Therefore, they were assigned a lesser urgency compared to medium priority tasks.

In line with our established agile sprint methodology, we agreed a deadline of three days once more and allocated tasks as per team request. With a pragmatic timeline set, reflective of the lower complexity and urgency of these tasks, we set about completing the code and creating not just a functioning game, but one with an enhanced player experience.

During this sprint we used our past learning and experience from the previous sprints to work in a cohesive and effective manner, with issues that required correction arising less frequently. Again, we completed basic user acceptance testing to ensure functionality and experience for the user was as anticipated.

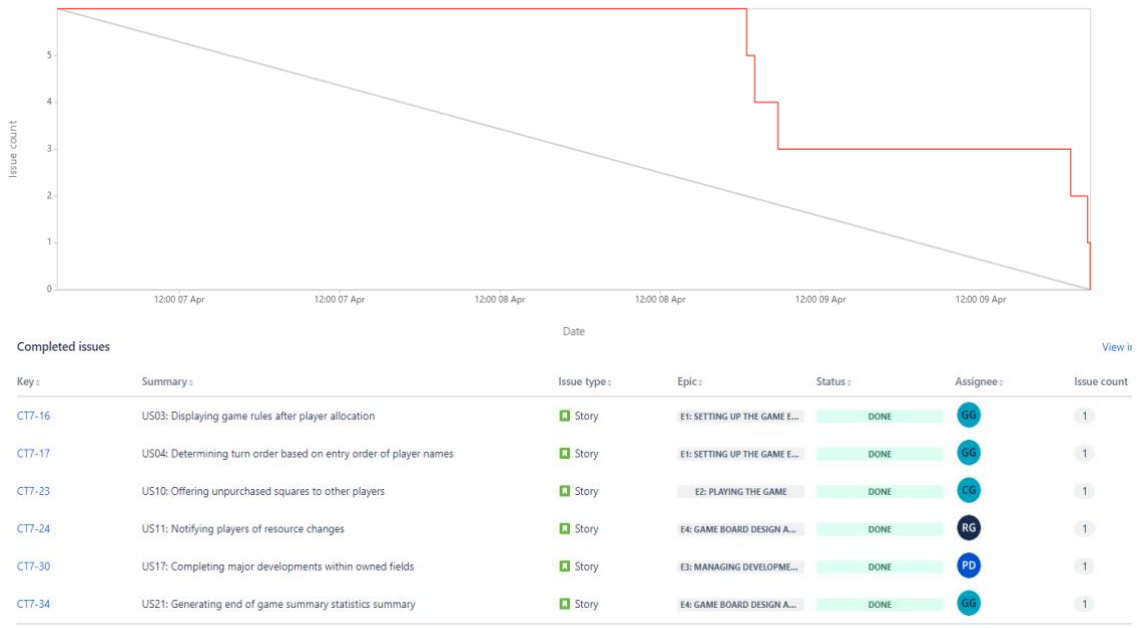


Figure 18: Sprint 4 Jira burndown – Low Priority

Upon completion of the sprint, we reviewed the game in its entirety as a group. This gave us an excellent oversight of functionality, ensuring it worked as expected based on our sequence diagrams/UML, and that it met the requirement of the specification.

Sprint 5: JUnit and UAT Testing

Although we completed basic user acceptance testing during our previous sprints, on review of our product after the previous sprint was complete, we decided it would be prudent to do further user acceptance testing and Junit testing on our finished product to make sure developer bias was not inadvertently missing errors. As such, we created a fifth and final sprint to more critically and independently quality assure our system, ensuring we had our desired levels of reliability and functionality. This sprint focused on implementing comprehensive testing strategies to validate the correctness of our code and ensure alignment with user expectations.

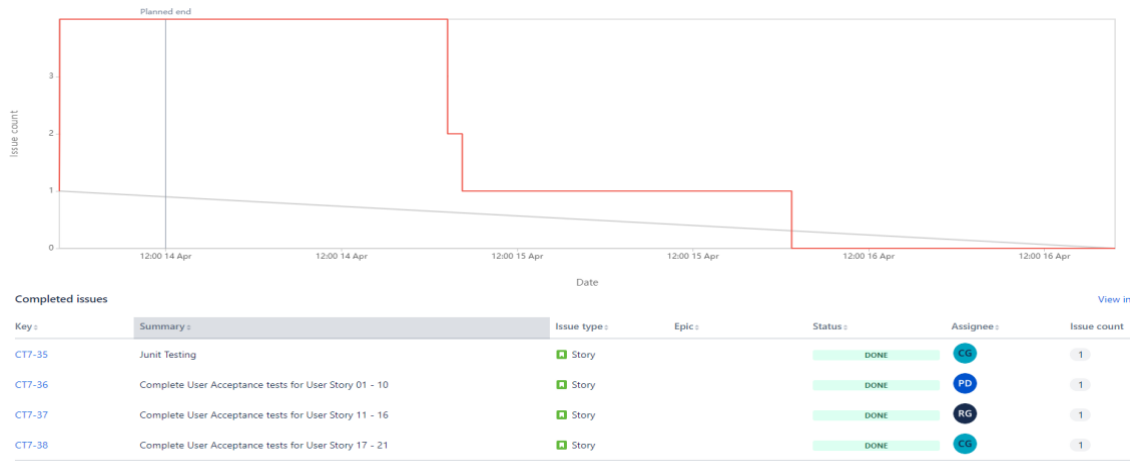


Figure 19: Sprint 5 Jira burndown – Testing

In line with our established sprint methodology, we set a short deadline of two days for the testing sprint as we could give more time and effort into it collectively as the game was already fully developed.

Through this sprint, we identified that one area of functionality was not quite working as expected (see 'Working System and Testing' section below for further details) but we used proactive communication and collaboration to swiftly resolve this discrepancy.

Upon completion of the sprint, we reviewed the tests as a group to ensure the outcomes were as expected and that all requirements had been met.

Working System and Testing

We successfully implemented a working game system with an environmental theme, as per the specifications. This can be seen from the submitted source code, group video, but perhaps most importantly, through our testing components.

Software testing is a critical part of the software development lifecycle (SDLC) and is a component of software validation and verification. Testing is conducted to identify errors or deviations from expected or required behaviour. When done well, it can ensure the quality, reliability, functionality and acceptability of a software product. Issues identified earlier in development are easier and less costly to fix, and therefore is an important part of the agile iterative approach.

Testing methodology

User acceptance testing

Exhaustive functional user-acceptance-testing (UAT) was carried out based on user stories, which were used to develop the test cases. Again, valid and invalid paths (where possible) were tested to demonstrate compliance with all user requirements.

Other aspects of UAT e.g. interface, performance, and security testing were out of scope for the project.

User acceptance testing results

On the initial run, all UAT tests, except one, passed. Partial failure was noted in the case where a player is forced to exit the game due to having insufficient funds to pay a fine to another player. Although the former was removed as per requirements, the credit balances of both players were not updated correctly. The code was corrected, and all tests were run again (basic regression testing) and all passed.

The full suite of user acceptance tests (valid and invalid), and associated evidence of successful tests is available for review in Appendix 10, but enclosed below is a working example for User story 1 (US01) for illustrative purposes:

Main Flow:

Test ID	User Story ID	Test Description	Test Initialisation	Test Inputs	Test Procedure	Expected results	Passed ?
1	US01	Set up the game with a valid number of players	Launch the game application	Number of players: 2	Player launches the game. Player enters number of human players	The game should accept the number of players and move on to asking for Player 1's name	Y

Alternative Flow:

Test ID	User Story ID	Test Description	Test Initialisation	Test Inputs	Test Procedure	Expected results	Passed ?
25	US01	Set up the game with an invalid number of players (low)	Launch the game application	Number of players: 1	User launches the game. User enters number of human players	The game should display an error message and ask player to enter valid number	Y
26	US01	Set up the game with an invalid number of players (high)	Launch the game application	Number of players: 5	User launches the game. User enters number of human players	The game should display an error message and ask player to enter valid number	Y

```
Enter number of players:
2
Player 1 Enter name:
```

Figure 20: Representative example of User Acceptance Testing and associated evidence – US01

Unit testing

Unit testing was applied selectively as proof of concept to Use Case 1 (UC01), demonstrating that the principles of unit testing were understood. If this were a real-life production system, we would ensure full Unit Testing would take place for all components of code.

For the JUnit testing, the ***setUpPlayers()*** method of the **Game** class was identified as the portion of code that fulfilled the requirements of UC01. Both the **Game** class constructor and its ***setUpPlayers()*** method were tested with valid and invalid data.

For valid data testing, assertions were used to test that the parameters were assigned appropriately in each method. For invalid testing, assertions were used to check that the expected exception types were thrown and that the expected exception messages were generated. Mocked scanner input was used to fully test ***setUpPlayers()*.**

Unit testing results

All four Junit tests passed completely:

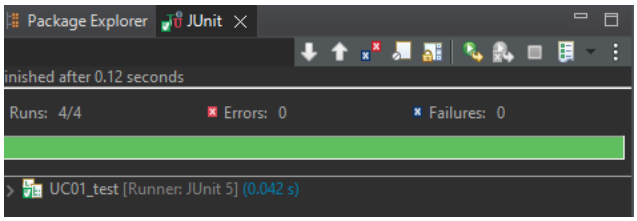


Figure 21: Representative example of Junit testing passing

For further details of the Junit tests, its rationale and evidence of success, please see Appendix 11.

Compatibility testing

Basic compatibility testing was carried out in that one group member ran the game on macOS while the rest of the group ran on Windows.

Improvements

Using Agile methodology, we successfully developed and implemented an environmental based console game as per the specification. Given the particular specifications and time limited nature of this assignment, we were precluded from implementing further sprints that could deliver further iterative improvements, but would have been in scope if this had been a real world software engineering project. Representative samples are included below:

- We identified that if you start a game and end it immediately, the first listed player is placed 1st and 'winning' the game. A future iteration would be to update this so that all players are placed equally in that instance.
- Player turn order could be random, rather than related to order of player entry
- Updating functionality so that game does not end if a single player quits, improving the player experience for the remaining players.
- Given our board was developed to be extendable, an interesting future game consideration could be that they board would be scalable and would increase based on number of players.
- Regex could be used for name validation.

In terms of testing improvements:

- Implementing A ***showBoard()*** method in the player's turn options that would display the details of all squares on the board. This would confirm compliance with test cases 15,17, and 18 in a single method call rather than requiring the game to be played until the details were seen in the console.
- Full Junit testing of the source code to ensure comprehensive test coverage.
- Create tests based on the sequence diagram to validate the interactions between intra-system classes and functions.
- Test-driven-development (TDD): in line with agile principles, TDD could be employed to enhance the development process by having testing at the centre of code development rather than being conducted once the product was built.

Conclusion

As evidenced in this report, video submission, Jira backlogs and GitLab evidence, we successfully achieved the core functionality of the assignment specification.

Although on a much smaller scale, this assignment emulated the software engineering processes required for real world product development and implementation, and as such, we as a group aspired to equally emulate the industry gold standards.

We implemented Agile methodology to mitigate any challenges we faced, but equally to keep our achievements and project timelines on track. This, in in combination with our shared understandings and belief in our product, we successfully delivered on the specification within the requested timeline.

Appendix 1:

List of Requirements

Requirement ID	Description
REQ1	The game will allow 2-4 human players to play a single game
REQ2	The game will allow human players to enter a unique name consisting of alphanumeric characters at the beginning of the game
REQ3	After players have been allocated, the rules of the game shall be displayed
REQ4	Players will take their turn to move in the same order they entered their name
REQ5	During player turn, they will be assigned a value equivalent to throwing two virtual dice which determines number of spaces to move
REQ6	Players will be updated to their location on the board after die roll (REQ4) which determines how many spaces they move
REQ7	Players will start with resources equal to 1000 credits
REQ8	Players shall be given a list of obligations and opportunities based on the square they have landed on
REQ9	Players shall be able to choose one action they want to complete per turn
REQ10	Players that choose to not purchase a square, it will be offered to other players
REQ11	After a Players action, they shall be notified about changes to their resources and why
REQ12	The game board should consist of 12 squares, with four separate 'fields' consisting of two squares and three squares respectively. It will also include a start square and an empty square
REQ13	Players that land on or pass the start square shall gain 200 credits
REQ14	One of the two field areas shall cost the least amount of resources
REQ15	One of the two field areas shall cost the most amount of resources
REQ16	A player shall be able to develop a square within a field when they own or control the whole field on their turn regardless of their board position to create a development
REQ17	A player shall be able to complete a major development in a square within a field when they own or control the whole field when they have three developments established in that square
REQ18	If a player lands on another players-controlled square, they must pay credits. The value of resources shall be scaled in line with the developments on the square
REQ19	If a player runs out of credits, they leave the game. Their areas are reverted to undeveloped squares and become open board spaces for other players to land and purchase
REQ20	If a player wants to stop playing, the game ends
REQ21	At the end of the game, game statistics summary shall be produced which includes the amount of credits each player holds

Appendix 2:

List of User Stories

US01. Title: Allowing 2-4 Human Players to Play a Single Game

Description: As a player, I want to be able to play a single game with 2-4 human players so that I can enjoy the game with friends or family.

Acceptance Criteria:

- The game should provide an option to select the number of human players before starting.
- The game should allow a minimum of 2 and a maximum of 4 human players to join.
- Once the determined number of players is reached, the game should prevent further players from joining.
- The game should proceed to the next stage once all human players have joined.

US02. Title: Entering Unique Player Names

Description: As a player, I want to be able to enter a unique name at the beginning of the game so that I can personalise my gaming experience. The game should validate the uniqueness and format of the entered name.

Acceptance Criteria:

- The game should prompt each player to enter their name at the beginning of the game.
- The entered name must be unique among all players and case sensitivity should not matter.
- The name must be at least 1 character, and no longer than 20 characters.
    - If a duplicate or invalid name is entered, the game should display an error message and prompt the player to enter a new name.
- The game should allow the player to proceed only after entering a unique and valid name.

US03. Title: Displaying Game Rules before Player Allocation

Description: As a player, I want to see the rules of the game displayed before players have been allocated so that I can understand how to play the game properly. The game should provide clear and concise explanations of the rules to ensure all players understand the gameplay mechanics.

Acceptance Criteria:

- Before all players have been allocated, the game should automatically display the rules on the screen.
- The rules should cover all essential aspects of gameplay, including movement, resource management, purchasing squares, and resolving interactions between players.
- The rules should be presented in an easily readable format with clear headings and bullet points for each rule.

US04. Title: Determining Turn Order Based on Entry Order of Player Names

Description: As a player, I want my turn order to be determined by the order in which I entered my name so that the gameplay remains fair and consistent.

Acceptance Criteria:

- The game should establish the turn order based on the sequence in which players enter their names at the beginning of the game.
- The first player to enter their name should take the first turn, followed by subsequent players in the order they entered their names.
- The turn order should be clearly displayed on the game interface, indicating which player's turn it is currently.
- The game should ensure that turn order remains consistent throughout the game, even in the event of player actions that may affect the order.

US05. Title: Rolling Virtual Dice to Determine Movement

Description: As a player, I want to roll two virtual dice during my turn, to determine the number of spaces I move on the game board. Rolling dice adds an element of chance to the game and affects strategic decision-making.

Acceptance Criteria:

- Player selects option to roll the dice, the game interface should automatically simulate the rolling of two virtual dice.
- The result of the dice roll should be displayed to the player, indicating the total value obtained from both dice.
- The player's game piece should move the number of spaces corresponding to the total value obtained from the dice roll.

US06. Title: Updating Player Position on the Game Board

Description: As a player, I want my position on the game board to be updated after rolling the dice so that I can see how many spaces I've moved.

Acceptance Criteria:

- After rolling the virtual dice, the game interface should display the player's current position on the game board.
- The player's game piece should move the appropriate number of spaces on the game board based on the result of the dice roll.
- The updated position of the player's game piece should be clearly visible to the player on the game interface.

US07. Title: Starting the Game with 1000 Credits

Description: As a player, I want to start the game with 1000 credits so that I have resources to use during gameplay. Starting with credits allows me to participate in various game mechanics such as purchasing squares, paying fees, or investing in developments.

Acceptance Criteria:

- At the beginning of the game, each player should be allocated 1000 credits.
- The player's initial credit balance should be clearly displayed on the game interface.
- Players should be able to use these credits to perform various actions throughout the game, such as purchasing squares, paying fees, or investing in developments.
- If any actions deplete the player's credit balance, the updated balance should be immediately reflected on the game interface.

US08. Title: Presenting Obligations and Opportunities Based on Landed Square

Description: As a player, I want to be presented with a list of obligations (game enforced actions) and opportunities (player chosen actions) based on the square I've landed on, so that I can make informed decisions during my turn.

Acceptance Criteria:

- Whenever a player lands on a square, the game should display a list of obligations and opportunities associated with that specific square.
- The list should include any actions or decisions the player needs to make as a result of landing on the square, such as paying fees, purchasing developments, or triggering events.
- The obligations and opportunities should be presented in a clear and concise manner, with brief descriptions to aid player understanding.
- The player should have the option to proceed with their turn only after reviewing the obligations and opportunities presented to them.
- Obligations such as paying fees/rent are automatically enforced on players during the turn

---

US09. Title: Player Selected Actions per Turn

Description: As a player, I want to be able to choose player selected actions to complete per turn so that I can strategically manage my resources and progress in the game.

Acceptance Criteria:

- At the beginning of each turn, the game should prompt the player to choose actions to complete.
- The player should be able to select their desired action from a menu or list provided by the game interface.
- Once the player selects an action, the game should execute the chosen action and update the game state accordingly.
- The player should be notified about the outcome of their chosen action, including any changes to their resources or game board position.

US10. Title: Offering Unpurchased Squares to Other Players

Description: As a player, if I choose not to purchase a square, it should be offered to other players, ensuring that all squares have the potential to be owned.

Acceptance Criteria:

- If a player decides not to purchase a square during their turn, the game should offer that square to the other players.
- The offer to purchase the square should be clearly communicated to all players.
- The player who wishes to purchase the square will enter their player number
- The game should then confirm that the player has purchased the square and displays their new balance.

US11. Title: Notifying Players of Resource Changes

Description: As a player, I want to be notified about changes to my resources after completing an action, along with the reason for the change, so that I can track my progress and make informed decisions.

Acceptance Criteria:

- After a player completes an action that affects their resources (e.g., purchasing a square, paying fees, earning credits), the game should display a notification informing the player of the change.
- The notification should include details about the specific action that caused the resource change (e.g., "You purchased Square X for Y credits").
- The notification should clearly indicate whether the change resulted in an increase or decrease in the player's resources.

US12 Title: Designing Game Board with 12 Squares and Four Separate Fields

Description: As a player, I expect the game board to consist of 12 squares, with four separate 'fields' consisting of two squares and three squares respectively. It will also include a start square and an empty square. This structured layout provides a clear and organized environment for gameplay, allowing for strategic planning and decision-making.

Acceptance Criteria:

- The game board should contain a total of 12 squares, including a start square and an empty square.
- Ten of the squares should form separate 'fields,' with two of the fields consisting of two squares each and the remaining two fields consisting of three squares each.
- Each field should be uniquely labelled on the game board.
- The start square and empty square should be positioned appropriately within the layout of the game board, providing balanced access and navigation for players.

US13. Title: Earning Credits by Landing on or Passing the Start Square

Description: As a player, if I land on or pass the start square, I should gain 100 credits, rewarding me for progressing in the game.

Acceptance Criteria:

- When a player lands on the start square during their turn, the game should automatically credit them with 100 credits.
- If a player passes the start square without landing on it directly, they should still receive the 100 credits.
- The credit bonus should be immediately added to the player's total credit balance, and the updated balance should be displayed on the game interface.
- The credit bonus should only be awarded once per complete circuit around the game board to prevent players from repeatedly gaining credits by circling the board.

US14. Title: Field Area with Lower Resource Cost for Strategic Investment

Description: As a player, I expect one of the two field areas to cost the least amount of resources, providing strategic opportunities for investment.

Acceptance Criteria:

- The game should designate one of the two square fields as having the lowest resource cost.
- The difference in resource cost between the two field areas should be significant enough to provide meaningful strategic choices for players.
- The lower resource cost of the designated field should be clearly communicated to players before making investment decisions.

US15. Title: Field Area with Higher Resource Cost for Strategic Investment

Description: As a player, I expect one of the two field areas to cost the most amount of resources, providing strategic opportunities for investment.

Acceptance Criteria:

- The game should designate one of the two square fields as having the highest resource cost.
- The difference in resource cost between the two field areas should be significant enough to provide meaningful strategic choices for players.
- The higher resource cost of the designated field should be clearly communicated to players before making investment decisions.

US16. Title: Developing Areas within Owned Fields

Description: As a player, if I own or control the whole field, I should be able to develop an area within that field during my turn, creating a development and potentially increasing its value.

Acceptance Criteria:

- When a player owns or controls all squares within a particular field, they should be eligible to develop a square within that field during their next turn.
- The player options should be updated to include an option for the player to initiate the development process once they meet the ownership criteria for the entire field.
- Upon selecting the development option, the player should be prompted to choose a specific square within the field to develop, only one square can be developed per turn, at a unique value resource.
- After selecting the square for development, the player should be required to pay a specified amount of resources as the development cost.
- Upon payment of the development cost, the selected square should be upgraded or enhanced, signifying the creation of a development.
- The creation of a development should provide strategic benefits to the player, such as increased resource generation

---

US17. Title: Completing Major Development in Owned Fields

Description: As a player, if I own or control the whole field and have established three developments in a square, I should be able to complete a major development, enhancing its value further.

Acceptance Criteria:

- When a player owns or controls all squares within a particular field and has established three developments in a specific square within that field, they should be eligible to complete a major development on their next turn.
- The creation of a major development should provide strategic benefits to the player, such as increased resource generation
- Upon payment of the major development cost, the selected square should be upgraded or enhanced, signifying the creation of a development.

US18. Title: Paying Credits for Landing on Another Player-Controlled Area

Description: As a player, if I land on another player-controlled square, I should pay credits based on the value of resources in that square, ensuring fair and balanced gameplay.

Acceptance Criteria:

- When a player lands on a square that is controlled by another player, the game should calculate the value of resources associated to that square and its field (number and level of developments).
- The game should display the calculated value of resources in the player-controlled area to the player who landed on it.
- The player who landed on the square should be required to pay a specified amount of credits as determined by the value of resources in the area.
- Upon payment of the credits, the player's credit balance should be deducted accordingly, and the transaction should be displayed in the game interface.

- The amount of credits to be paid should be reasonable and balanced to ensure fair gameplay and prevent excessive penalties for players.
- The game should provide clear notifications to both players involved in the transaction, indicating the payment of credits and any associated changes in game state.

US19. Title: Exiting the Game due to Insufficient Credits

Description: As a player, if I run out of credits, I should leave the game, and my areas should revert to undeveloped spaces, providing opportunities for other players to continue.

Acceptance Criteria:

- If a player's credit balance reaches zero during gameplay, the game should initiate the exit process for that player.
- Upon exiting the game, all areas owned or controlled by the departing player should revert to undeveloped spaces.
- The game should remove any developments or modifications made by the departing player within the reverted areas, restoring them to their original state.
- The game should provide notifications or alerts to all remaining players about the departure of a player and the resulting changes in game state.
- The exit process should be executed seamlessly and without causing significant delays or interruptions for other players.

US20. Title: Ending the Game When a Player Chooses to Stop Playing

Description: As a player, if I choose to stop playing, the game should end, concluding the game for all players. This feature ensures that players can gracefully exit the game whenever they choose, allowing all participants to conclude the game session together.

Acceptance Criteria:

- The game interface should provide a player selected action labelled "Quit Game", allowing players to indicate their decision to stop playing.
- When a player selects the "Quit Game" option, the game should prompt them to confirm their decision to stop playing.
- Upon confirmation, the game should initiate the process of ending the game session.
- The game should notify all remaining players about the decision of the exiting player to stop playing and the impending end of the game session.
- The game should conclude immediately once decision has been confirmed.
- The game should display a final summary screen or message indicating the end of the game session.
- The game should provide options for players to review game statistics, such as credits held by each player, developments made, and other relevant information, before ending the game session.

US21. Title: Generating End-of-Game Statistics Summary

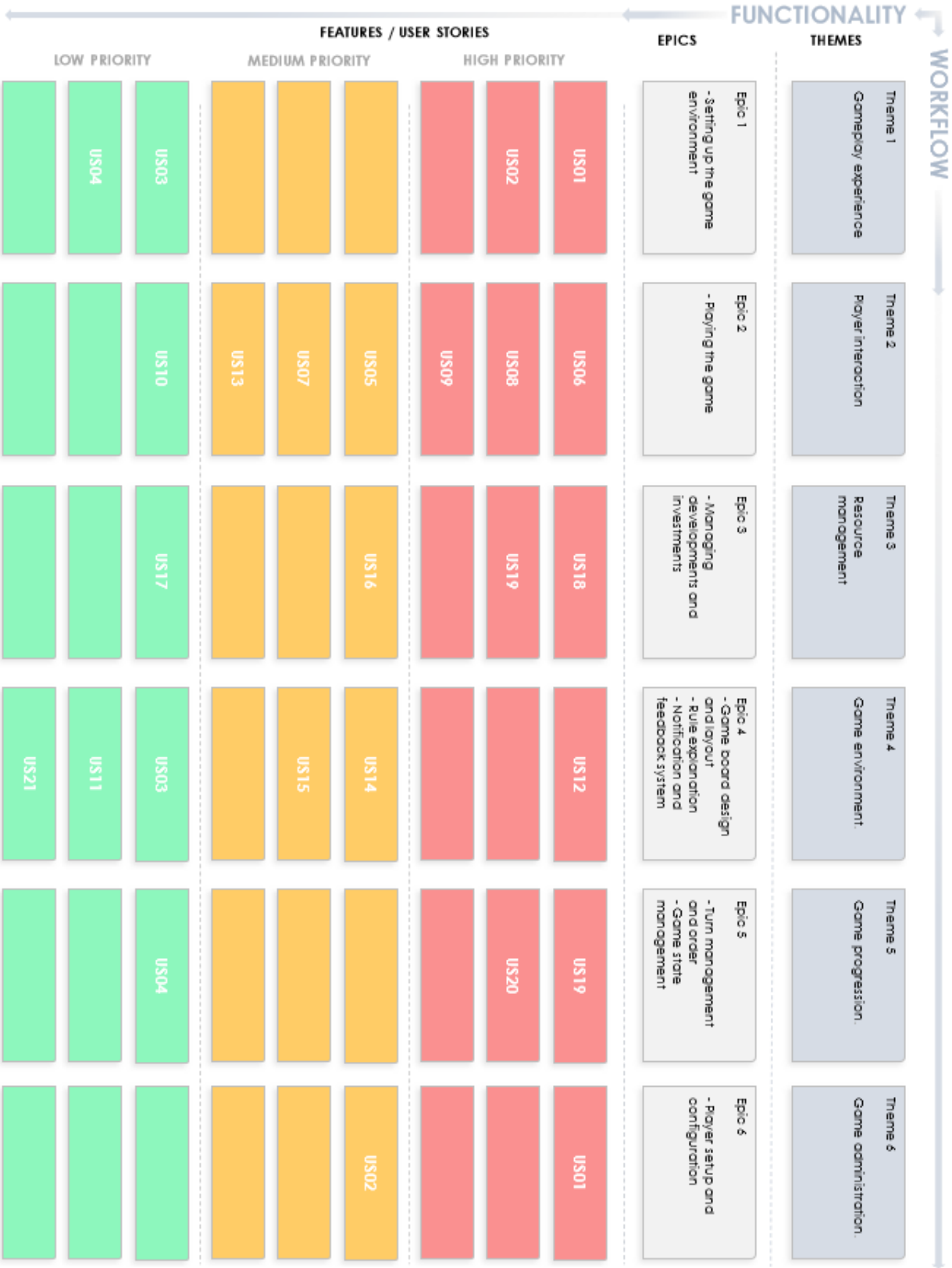
Description: As a player, I expect a game statistics summary to be produced at the end of the game, including the amount of credits each player holds, providing insights into our performance.

Acceptance Criteria:

- When the game ends, a statistics summary should be generated to include the total credits for each player.
- The statistics summary should be presented in a clear and organised format, making it easy for players to interpret and analyse the information.

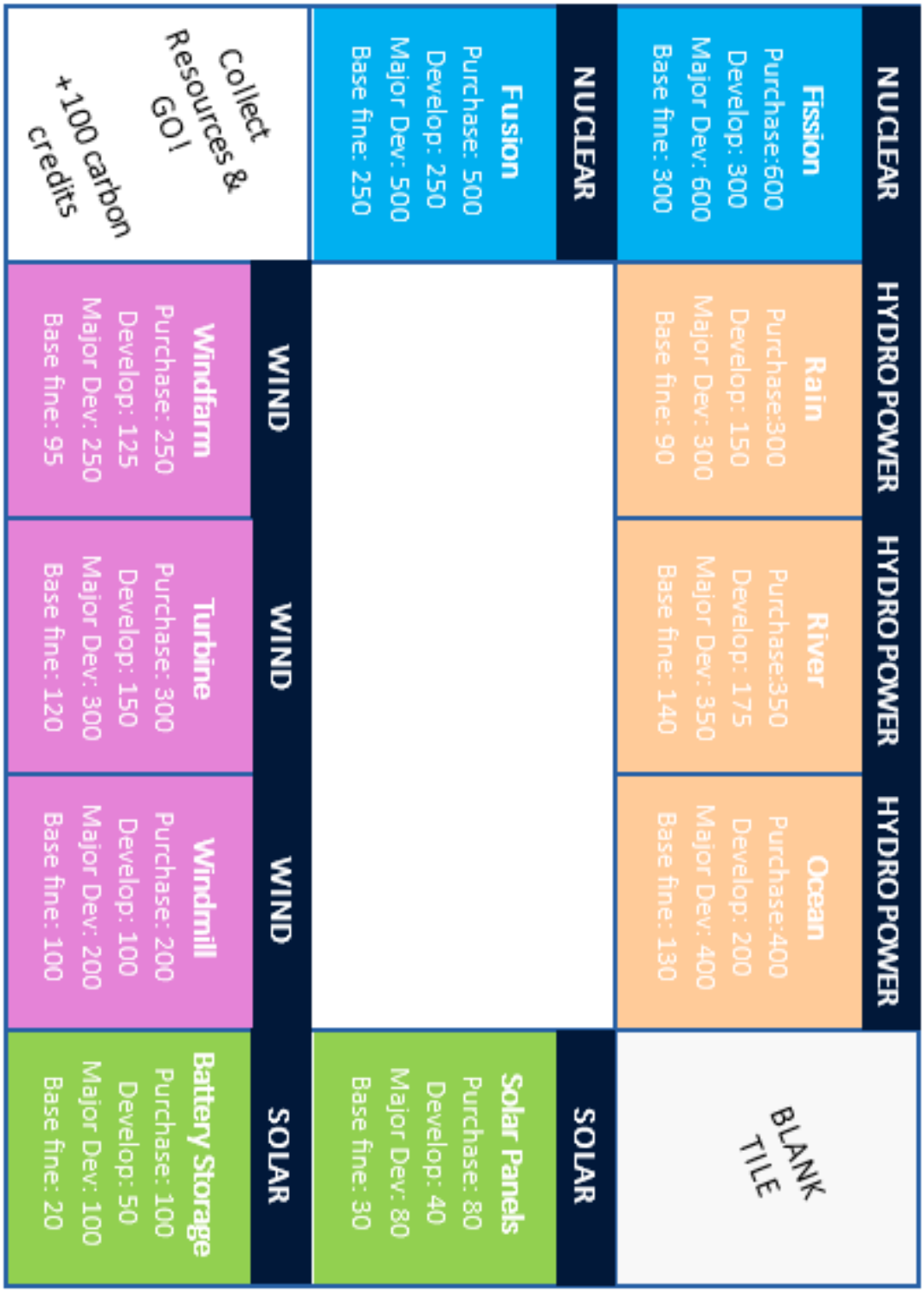
Appendix 3:

User Story Map



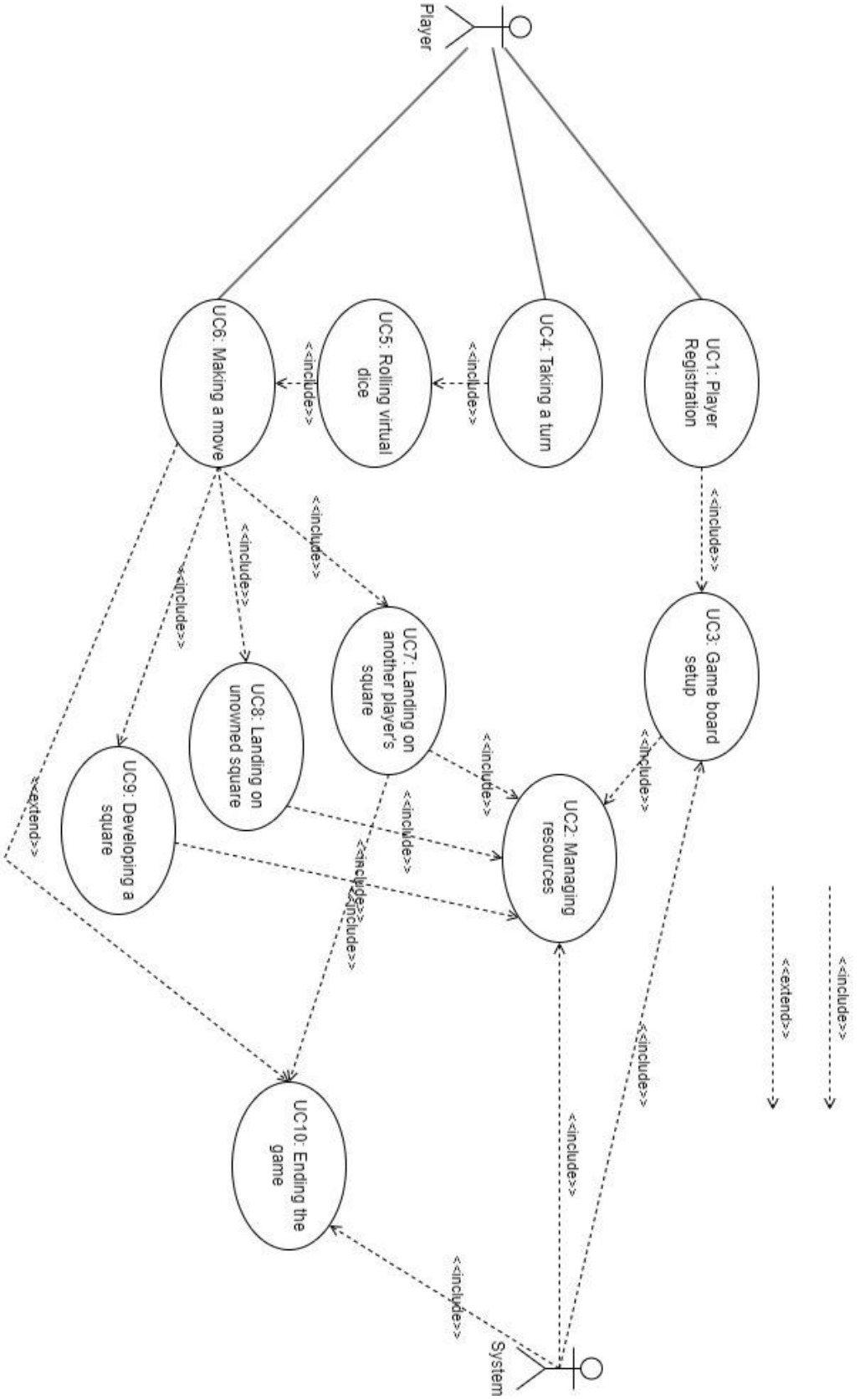
Appendix 4:

Virtual Game Board



Appendix 5:

Use Case Diagram



Appendix 6:

Use Case Tables

Name	Player Registration (UC1)
Objective	This use case covers the process of players registering for a new game, including selecting the number of players and entering unique player names.
Actor(s)	Player
Pre-conditions	1. The game application is launched and ready for a new game setup.
Main Flow	1. Player selects the option to start a new game. 2. Player specifies the number of human players (2-4). 3. Game validates the selected number of players. 4. If the selected number is valid (2-4 players), the game proceeds to player name entry. 5. Player enters a unique name when prompted, at least 1 character in length, and no longer than 20 characters in length. 6. Game validates the uniqueness and format of the entered name (ignore case) 7. If the name is valid, it is stored, and the next player is prompted to enter their name. 8. If the name is invalid (not unique, <1 character in length, >21 characters in length), an error message is displayed, and the player re-enters a name. 9. Steps 5-8 are repeated for each player until all players have entered unique names.
Alternative Flows	1. If the player selects an invalid number of players (less than 2 or more than 4), display an error message and prompt the player to select a valid number. 2. If the entered name is not unique, <1 character in length, >21 characters in length, display an error message and prompt the player to enter a valid and unique name. Return to Step 5.
Post-conditions	1. Each player has been allocated a space on the game board, and their unique names are stored for further use. 2. The game is ready to proceed to the next phase of gameplay.

Name	Updating Resources and Notifying Players of Resource Changes (UC 2)
Objective	This use case involves managing player resources, including starting credits allocation, earning credits during gameplay, and notifying players of any changes to their resources after completing an action.
Actor(s)	Game System / Player
Pre-conditions	1. Game is initialised and ready to start. 2. Player has completed an action that affects their resources.

Main Flow	1. At the beginning of the game, the game system allocates 1000 credits to each player. 2. The initial credit balance for each player is displayed on the game interface. 3. Players utilize these credits to perform various in-game actions throughout the game. 4. When a player lands on or passes the start square during their turn, the game system detects the event. 5. The player is awarded 100 credits as a bonus for landing on or passing the start square. 6. The credit bonus is added to the player's total credit balance. 7. The player's updated credit balance is displayed on the game interface. 8. After completing an action that affects their resources: a. The game system calculates the change in the player's resources based on the completed action. b. Notification is displayed to the player, indicating the resource change and the reason for it. c. Player acknowledges the notification and proceeds with their turn.
Alternative Flows	1. If the game encounters an error while allocating credits at the beginning of the game: a. The game system notifies the players about the issue. b. Players are prompted to try starting the game again or contact support if the problem persists. 2. If the player's resources cannot cover the cost of the action: a. A warning message is displayed, indicating insufficient resources. b. The player is prompted to take corrective action (e.g., do not proceed with purchase), or if computer-enforced action of paying credits to another player, will exit the game.
Post-conditions	1. Each player has been allocated 1000 credits at the beginning of the game. 2. The initial credit balance for each player is displayed on the game interface. 3. Players can utilize these credits to perform various in-game actions. 4. When a player lands on or passes the start square, they are awarded 200 credits as a bonus, which is added to their total credit balance. 5. The player's updated credit balance is displayed on the game interface. 6. Player has completed an action that affects their resources. 7. Game system calculates the change in the player's resources based on the completed action. 8. Notification is displayed to the player, indicating the resource change and the reason for it. 9. Player acknowledges the notification and proceeds with their turn.

	10. If the player's resources cannot cover the cost of the action, a warning message is displayed, and the player is prompted to take corrective action.

Name	Game Board Set Up with Field Areas for Strategic Investment (UC 3)
Objective	This use case involves designing the game board layout, organising squares into separate fields, and determining resource costs for strategic investment opportunities.
Actor(s)	Game Designer / Game System
Pre-conditions	Game setup phase
Main Flow	1. The game designer determines the layout of the game board, ensuring it contains 12 squares. 2. Squares are organised into four separate fields, with two fields consisting of two squares each and the remaining two fields consisting of three squares each. 3. Start square and empty square are positioned appropriately within the layout. 4. Field labels are added to distinguish between the different areas on the game board. 5. The game system selects one field area with the lowest resource cost and another with the highest resource cost to offer strategic investment opportunities. 6. The difference in resource cost between the two field areas is determined to provide meaningful strategic choices for players. 7. Lower resource cost of the designated field is communicated to players to indicate potential areas for profitable investments. 8. Higher resource cost of the designated field is communicated to players to highlight potential challenges and rewards associated with investments.
Alternative Flows	1. If the game designer decides to change the number of squares or fields: a. The layout is adjusted accordingly to accommodate the changes. b. Field labels and positions are updated to reflect the new layout. 2. If the game system encounters issues in determining the field with the lowest or highest resource cost: a. The system recalculates the resource costs based on predefined criteria. b. Manual intervention by the game designer may be required to resolve any discrepancies.
Post-conditions	1. The game board layout contains 12 squares organised into four separate fields. 2. Two fields consist of two squares each, and the remaining two fields consist of three squares each. 3. The start square and empty square are positioned appropriately within the layout.

	4. Field labels are added to distinguish between the different areas on the game board. 5. One field area is designated with the lowest resource cost, and another with the highest resource cost, offering strategic investment opportunities. 6. Players are informed about the resource costs of designated field areas, facilitating informed decision-making during gameplay.

Name	Taking a turn (UC 4)
Objective	This use case encompasses both the process of determining turn order and providing players with access to game rules for reference during gameplay.
Actor(s)	Player / Game System
Pre-conditions	All players have entered their names, and the game is ready to start
Main Flow	1. The game system establishes the turn order based on the sequence of player name entry. 2. The first player to enter their name takes the first turn, followed by subsequent players in entry order. 3. Turn order is clearly displayed on the game interface. 4. Before turn order is established, the game automatically displays the rules on the screen. 5. The rules cover all essential aspects of gameplay and are presented in an easily readable format with clear headings and bullet points. 6. Players have the option to review the rules at any time during the game from the game menu.
Alternative Flows	1. If the game fails to automatically display the rules: a. Players can access the game menu. b. Players select the "View Rules" option from the menu. c. The game displays the rules on the screen as per the basic flow. 2. If there are technical issues or conflicts in determining the turn order: a. The game system prompts players to re-enter their names. b. After resolving any conflicts, the turn order is recalculated based on the new sequence of name entry.
Post-conditions	1. Players have been informed of the game rules, which are accessible from the game menu for reference during gameplay. 2. Rules cover essential gameplay mechanics, providing players with a clear understanding of the game dynamics. 3. Turn order has been established and displayed on the game interface. 4. Players are aware of their position in the turn order.

Name	Rolling Virtual Dice to Determine Movement (UC 5)
Objective	This use case allows players to roll virtual dice during their turns to determine movement on the game board
Actor(s)	Player

Pre-conditions	It is the player's turn
Main Flow	1. Player selects the option to roll the dice. 2. Game simulates the rolling of two virtual dice. 3. Game displays the result of the dice roll. 4. Player's 'game piece' moves the corresponding number of spaces on the game board.
Alternative Flows	1. If there are technical issues with simulating the dice roll: a. The game prompts the player to retry rolling the dice. b. If the issue persists, the player may choose to roll again or exit the turn.
Post-conditions	1. The player's 'game piece' has moved the appropriate number of spaces on the game board. 2. The result of the dice roll is displayed to the player. 3. The player is ready to proceed with their turn.

Name	Making a Move (UC 6)
Objective	This use case combines the process of updating the player's position on the game board after rolling the dice with handling obligations and opportunities associated with the landed square
Actor(s)	Game System / Player
Pre-conditions	Player has rolled the virtual dice and landed on a square
Main Flow	1. The game system updates the player's current position on the game board based on the dice roll. 2. The player's game piece moves the appropriate number of spaces on the game board. 3. The game system identifies the square the player has landed on. 4. Obligations and opportunities specific to that square are retrieved from the game database. 5. If there are obligations or opportunities associated with the square: a. The list of obligations and opportunities is displayed to the player on the game interface. b. The player can review the list and choose the appropriate action to proceed with their turn. 6. If there are no obligations or opportunities associated with the square: a. The game system notifies the player that there are no specific actions required or available. b. The player proceeds with their turn as usual.
Alternative Flows	1. If there are obstacles, special conditions, or events on the board affecting movement or square events: a. The game system adjusts the player's position accordingly. b. Special events or actions triggered by landing on specific squares are executed.
Post-conditions	1. The player's position on the game board has been updated. 2. Any obligations and opportunities associated with the landed square have been presented to the player. 3. The player has reviewed the list and chosen the appropriate action to proceed with their turn, if applicable.

	4. If no obligations or opportunities are associated with the landed square, the player is informed accordingly, and they continue their turn without any specific actions.

Name	Landing on Another Player's square (UC 7)
Objective	This use case outlines the process of the game system automatically enforcing the payment of credits when a player lands on a square controlled by another player
Actor(s)	Player / Game System
Pre-conditions	Player lands on a square controlled by another player during their turn
Main Flow	1. The game system calculates the value of resources in the player-controlled area. 2. The calculated value of resources is displayed to the player who landed on the square. 3. The player is required to pay the specified amount of credits based on the value of resources in the area. 4. The player's credit balance is automatically deducted by the amount paid. 5. Transaction details, including the payment and any changes in credit balance, are recorded and displayed on the game interface.
Alternative Flows	1. If the player cannot afford to pay the required credits: a. The system notifies the player about insufficient credits. b. The player is prompted to choose a different action or end their turn.
Post-conditions	1. The player pays the specified amount of credits to the owner of the player-controlled area. 2. The player's credit balance is deducted by the amount paid. 3. Transaction details, including the payment and any changes in credit balance, are recorded and displayed on the game interface. 4. If the player lacks sufficient credits to pay, they are notified, and no payment occurs – that player exits the game.

Name	Landing on Unowned Square (UC 8)
Objective	This use case enables players to choose an action if they land on a square that is not owned by any player. They can either purchase the square themselves or offer it to another eligible player
Actor(s)	Player / Game System
Pre-conditions	It is the player's turn, and the square landed on is not owned by any player
Main Flow	1. Player is prompted to choose one action from a list of available options. 2. If the player selects to purchase the square: a. The game system calculates the cost of the square. b. Player's credit balance is checked to ensure affordability. c. If the player can afford it, the square is purchased, and ownership is transferred to the player.

	d. Transaction details are recorded and displayed on the game interface. 3. If the player selects to offer the square to another player or has insufficient funds to purchase the square: a. The offer to purchase the square should be clearly communicated to all players. b. They player who wishes to purchase the square will enter their player number c. The game should then confirm that the player has purchased the square area and displays their new balance 4. After either purchasing the square or offering it to another player, the game state is updated accordingly.
Alternative Flows	1. If the player fails to select an action within a specified time limit: a. The game system automatically selects a default action (e.g., skipping the turn). b. The game state is updated accordingly. c. Player is notified about the default action chosen by the system.
Post-conditions	1. It is the player's turn. 2. Player is prompted to choose one action from a list of available options: Purchase or Offer to Other Player. 3. After either purchasing the square or offering it to another player, the game state is updated accordingly. 4. If the player fails to select an action within a specified time limit, a default action is automatically chosen by the system, and the game state is updated accordingly.

Name	Developing a Square (UC 9)
Objective	This use case allows players to choose an action if they own or control all squares within a particular field. They can either develop individual squares within the field or pass their turn
Actor(s)	Player
Pre-conditions	It is the player's turn, and the player owns or controls all squares within a specific field
Main Flow	1. Player is prompted to choose one action from a list of available options: Develop a Square or Pass Turn. 2. If the player selects to develop a square: a. Player chooses a specific square within the field to develop during their turn. b. The system calculates the development cost for the selected square. c. Player's resources are checked to ensure affordability. d. If the player can afford it, the square is developed, and the development is initiated. e. The development provides strategic benefits to the player, potentially increasing resource generation or other advantages. 3. If the player chooses to pass their turn: a. The player's turn ends without taking any further actions. b. The game proceeds to the next player's turn.

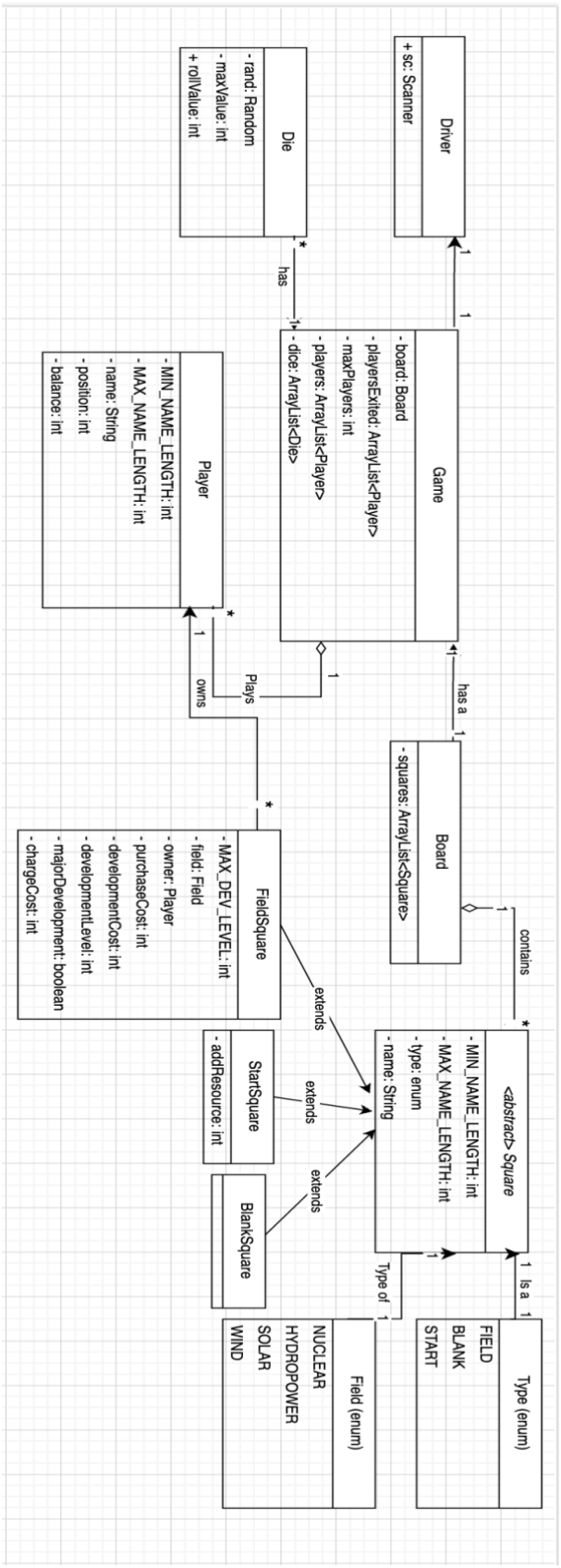
Alternative Flows	1. If the player fails to select an action within a specified time limit: a. The game system automatically selects a default action (e.g., passing the turn). b. The player's turn ends without taking any further actions.
Post-conditions	1. It is the player's turn. 2. Player is prompted to choose one action from a list of available options: Develop a Square or Pass Turn. 3. After either developing a square or passing the turn, the game state is updated accordingly. 4. If the player fails to select an action within a specified time limit, a default action is automatically chosen by the system, and the game state is updated accordingly.

Name	Ending the Game (UC 10)
Objective	This use case involves concluding the game session, either due to a player's decision to quit, insufficient credits leading to a player's exit, or the natural end of the game, followed by generating end-of-game statistics
Actor(s)	Player / Game System
Pre-conditions	Player's credit balance reaches zero during gameplay, or a player decides to quit the game, or the game session concludes
Main Flow	If a player's credit balance reaches zero during gameplay: 1. Game detects that a player's credit balance has reached zero. 2. Game initiates the exit process for the departing player. 3. All areas owned or controlled by the departing player revert to undeveloped spaces. 4. Any developments or modifications made by the departing player within the reverted areas are removed. 5. Notifications or alerts are sent to all remaining players about the departure of the player and resulting changes in the game state. If a player decides to quit the game: 1. Player selects the "Quit Game" option from the game interface. 2. Game prompts the player to confirm their decision to stop playing. 3. Upon confirmation, the game initiates the process of ending the game session. 4. Notifications are sent to all remaining players about the decision of the exiting player and the impending end of the game session. 5. The game concludes immediately after confirmation, displaying a final summary screen or message indicating the end of the game session. 6. Options are provided for players to review game statistics before ending the game session. If the game session concludes:

	1. Game session ends, triggering the generation of end-of-game statistics. 2. Statistics summary is generated, including total credits held by each player. 3. The statistics summary is presented in a clear and organised format on the game interface. 4. Players can interpret and analyse the information provided in the statistics summary to gain insights into their performance during the game.
Alternative Flows	If there are errors in any of the above processes: 1. The game system handles the errors accordingly, displaying relevant error messages.
Post-conditions	For a player's credit reaching zero: The departing player exits the game, and relevant game areas revert to undeveloped squares. For a player deciding to quit: The game session ends, and all players are notified. A final summary screen or message is displayed, and players are provided with options to review game statistics. For the natural end of the game session: The end-of-game statistics summary is successfully generated and presented to players for analysis.

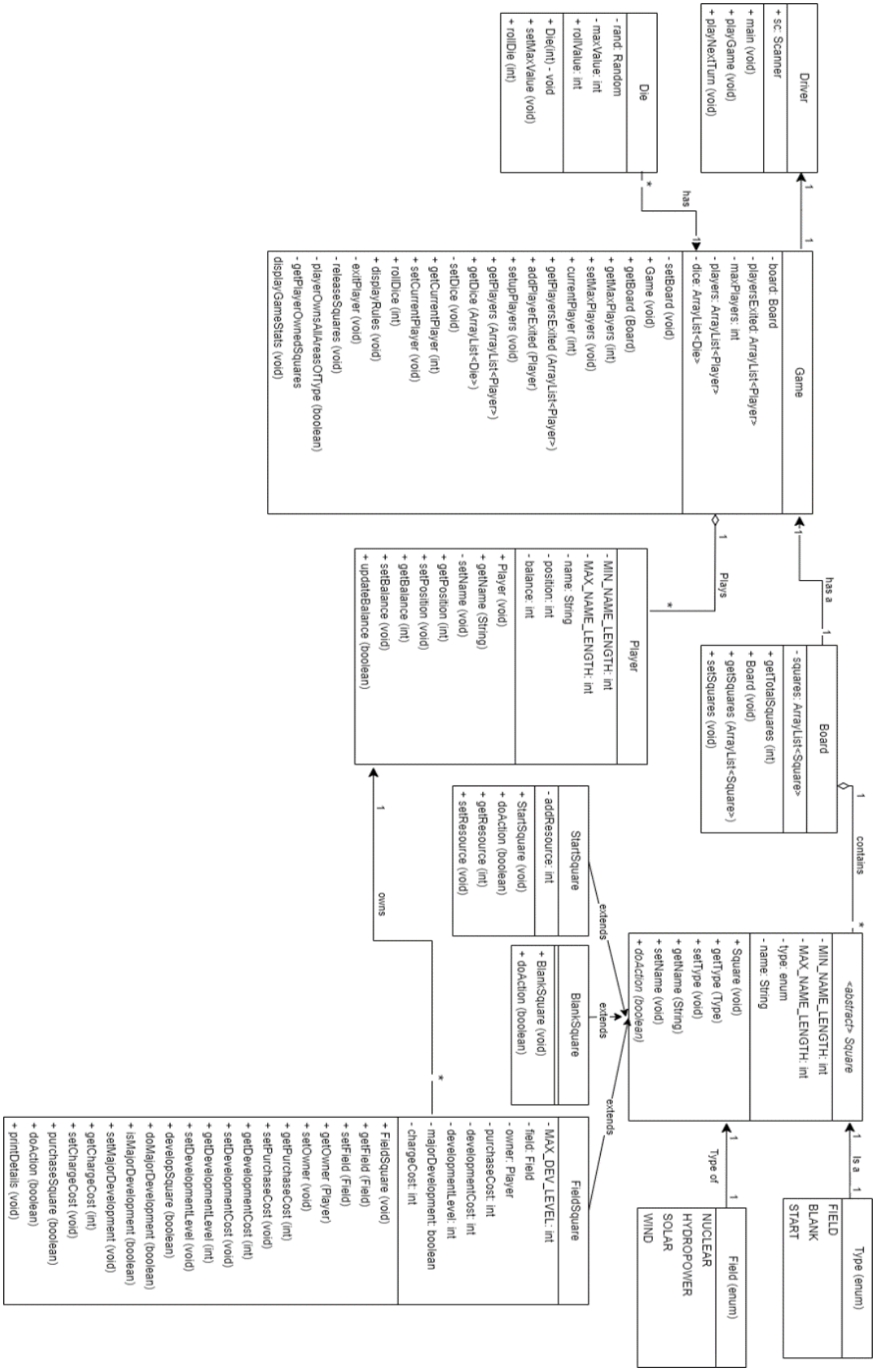
Appendix 7:

Domain Level Diagram



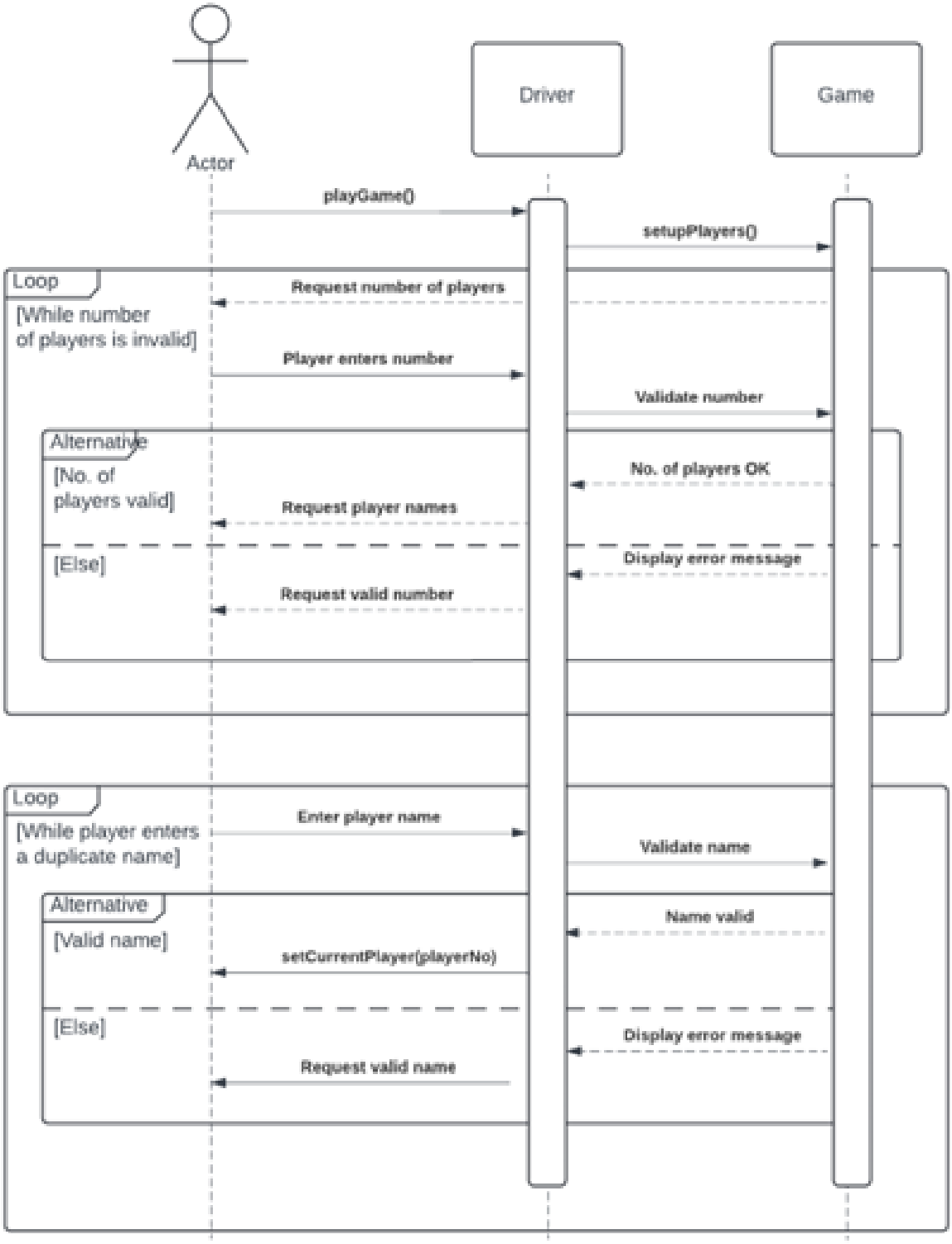
Appendix 8:

UML Diagram

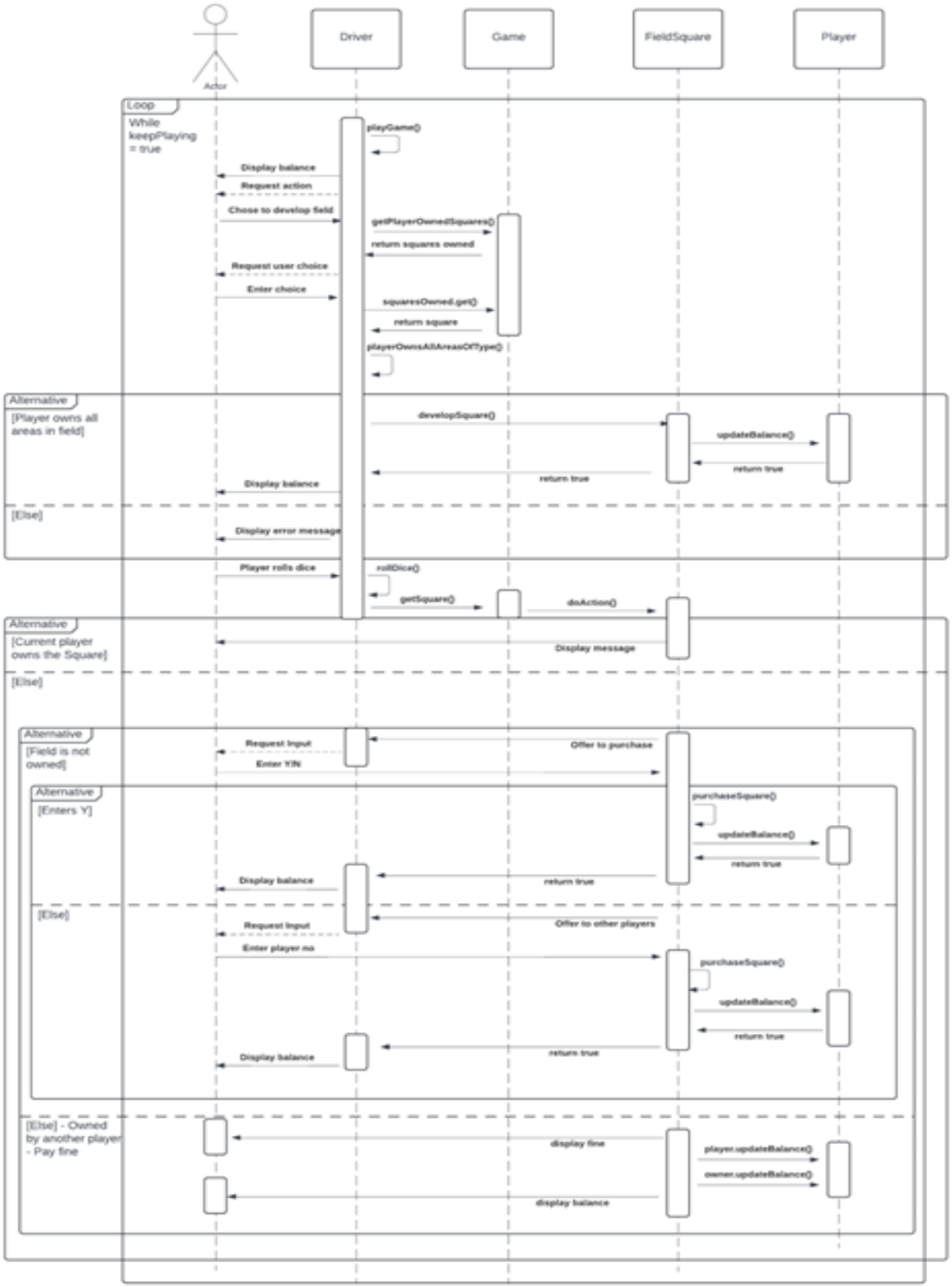


Appendix 9:

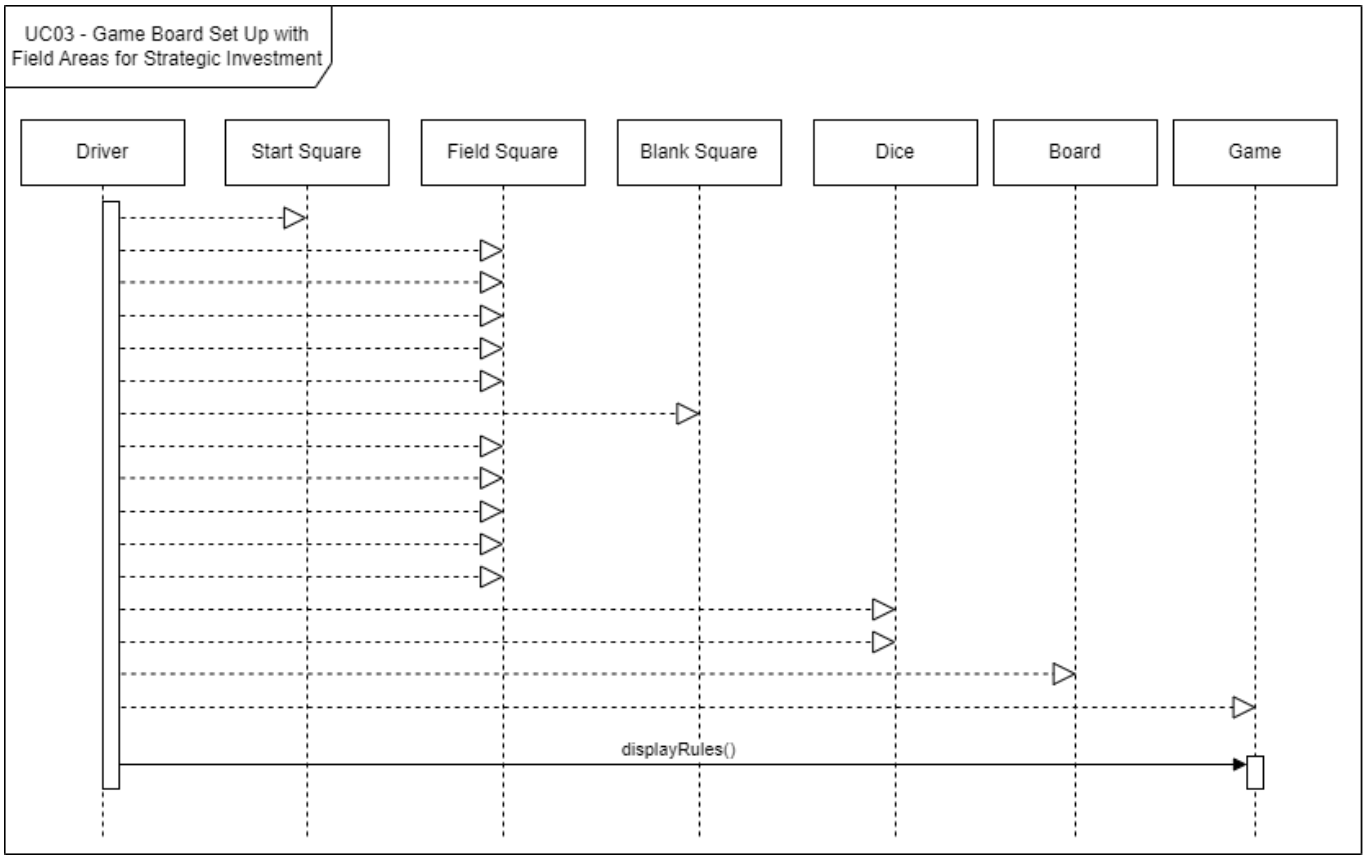
Sequence Diagrams



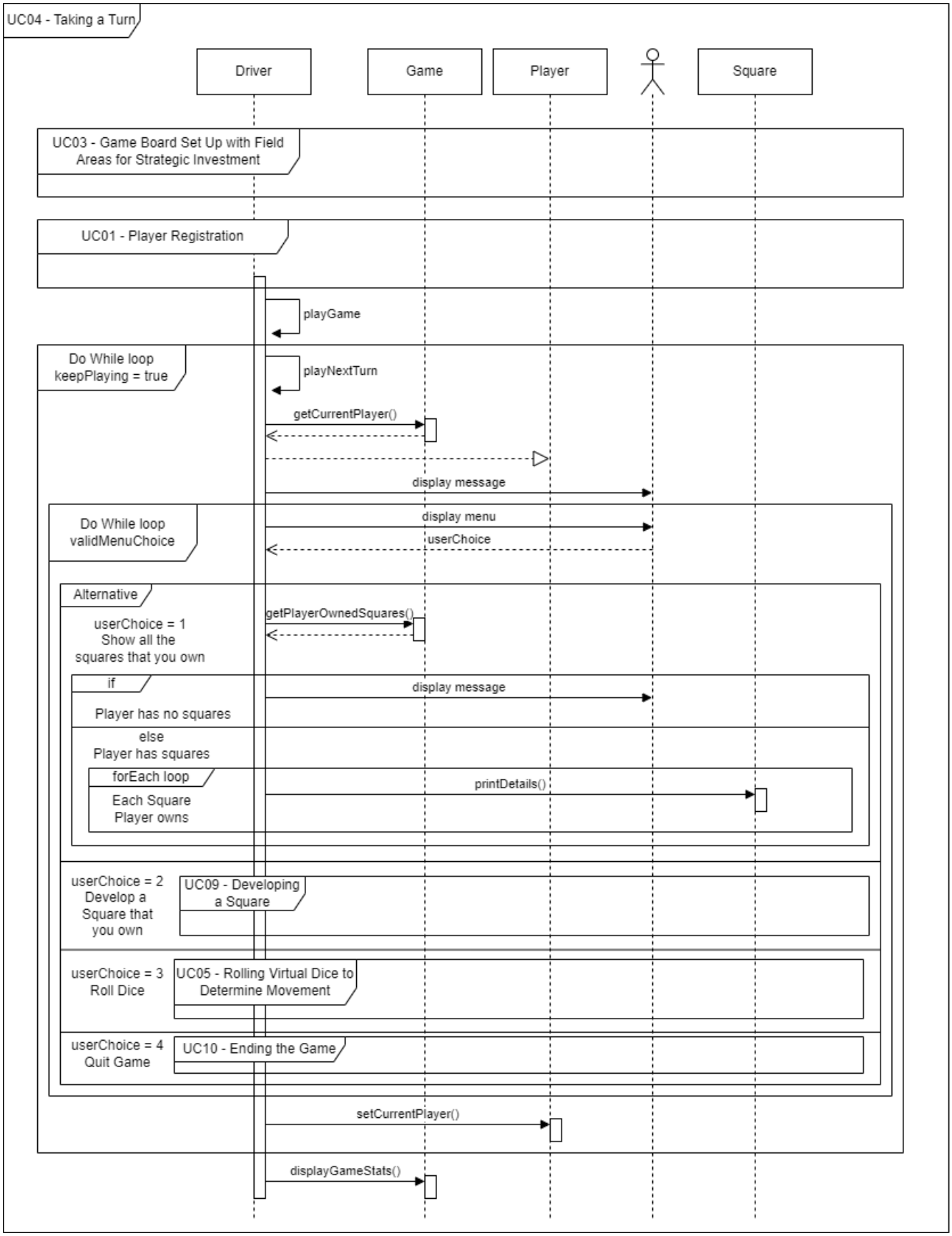
UC01 Sequence Diagram



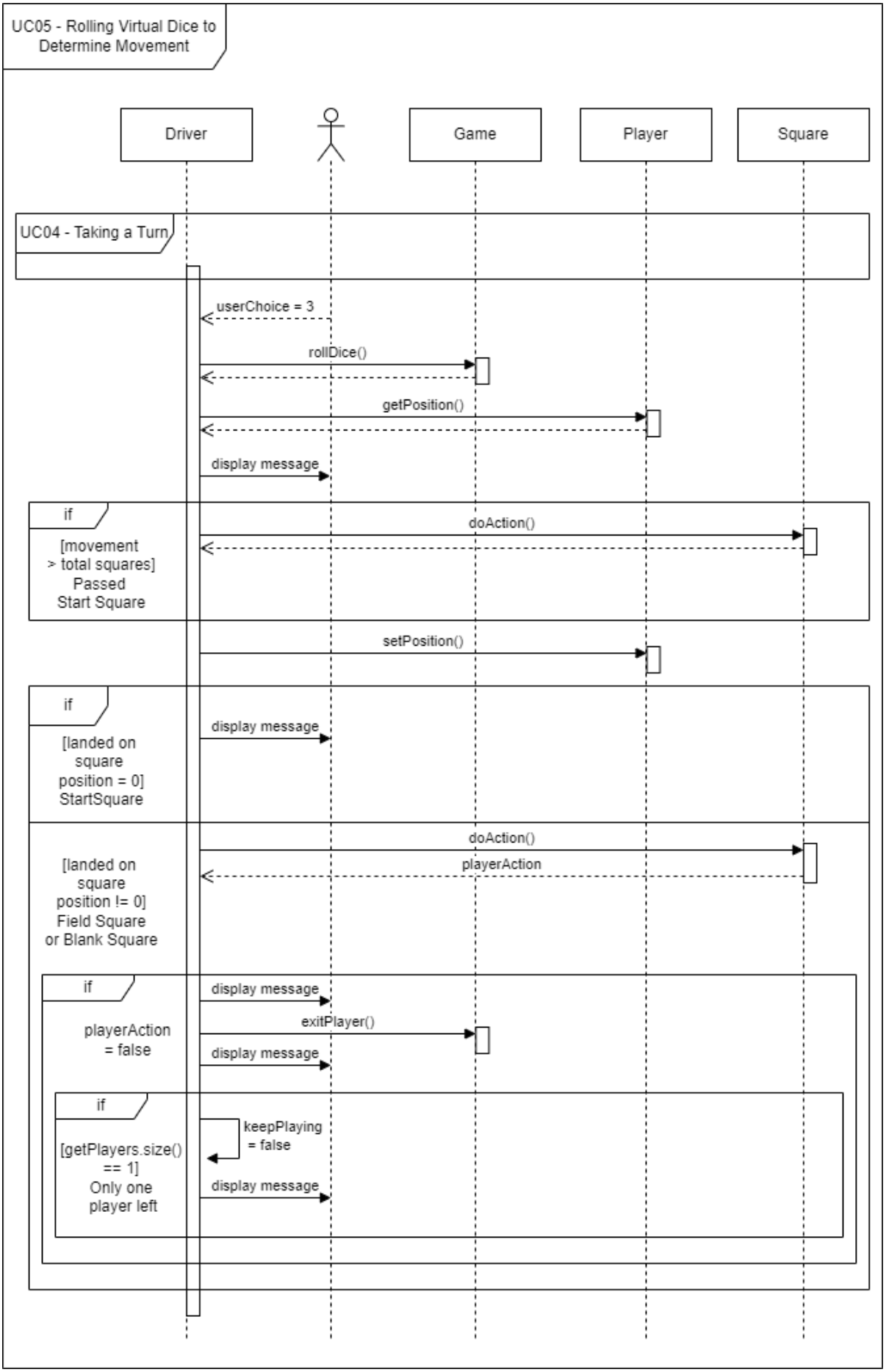
UC02 Sequence Diagram



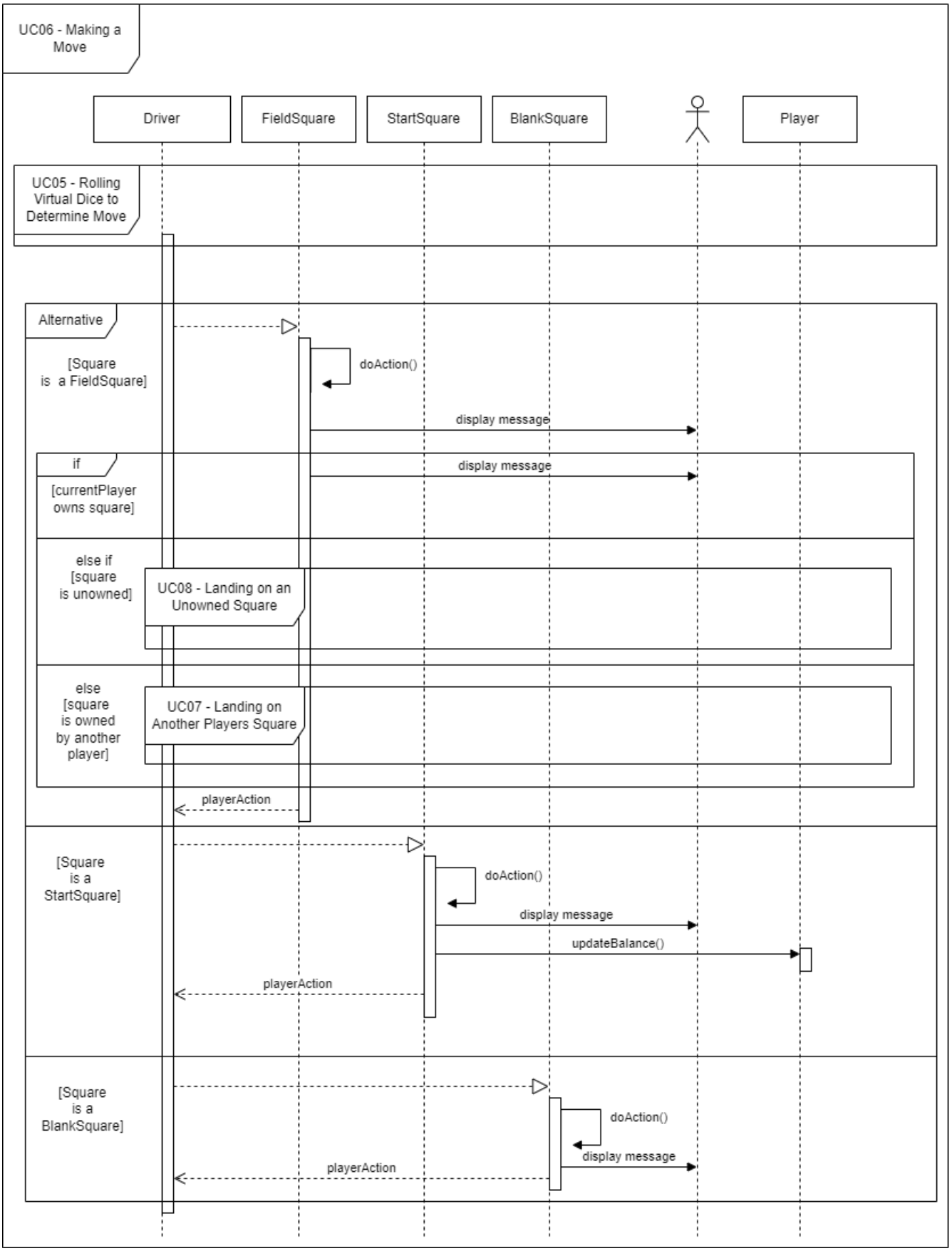
UC03 Sequence Diagram



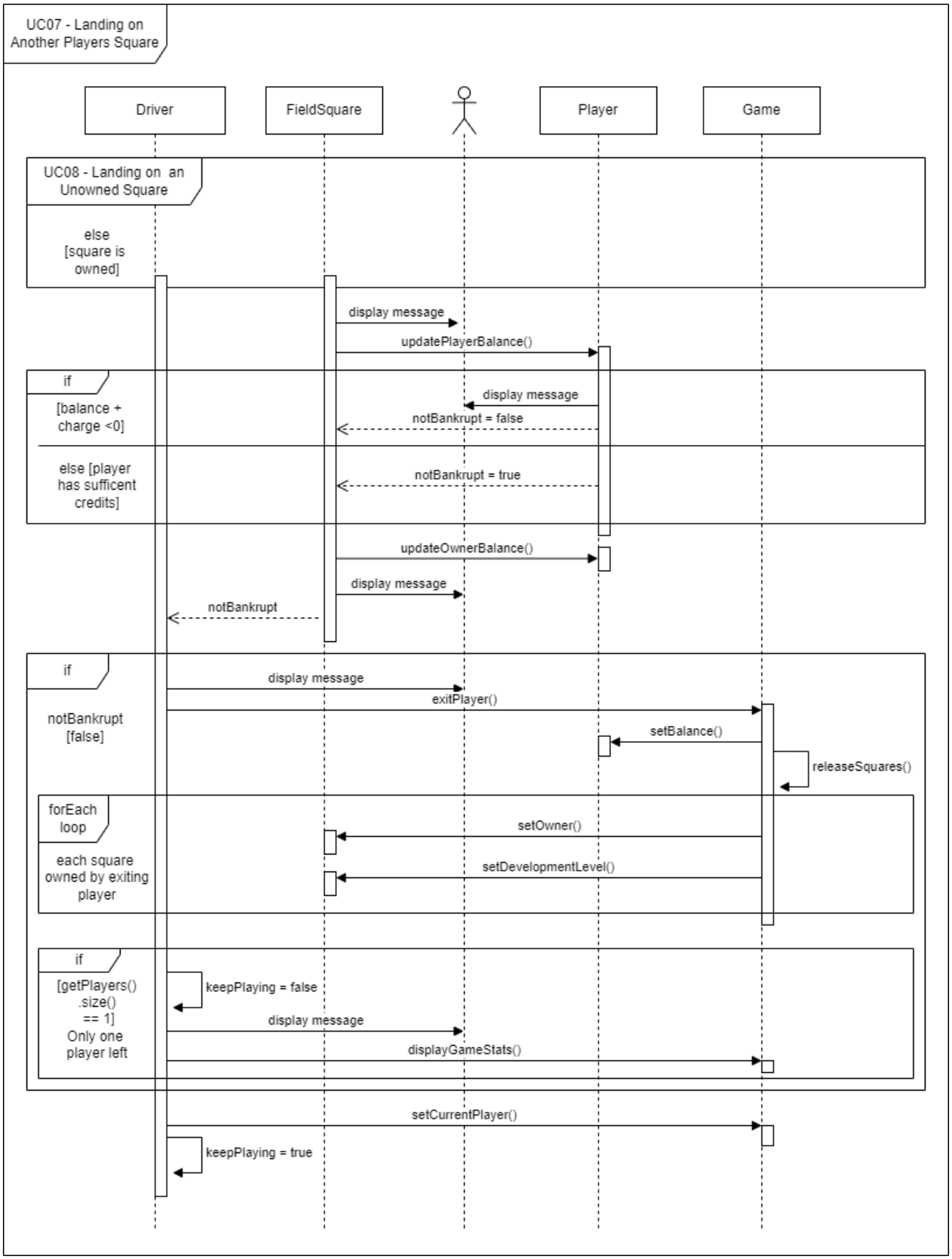
UC04 Sequence Diagram



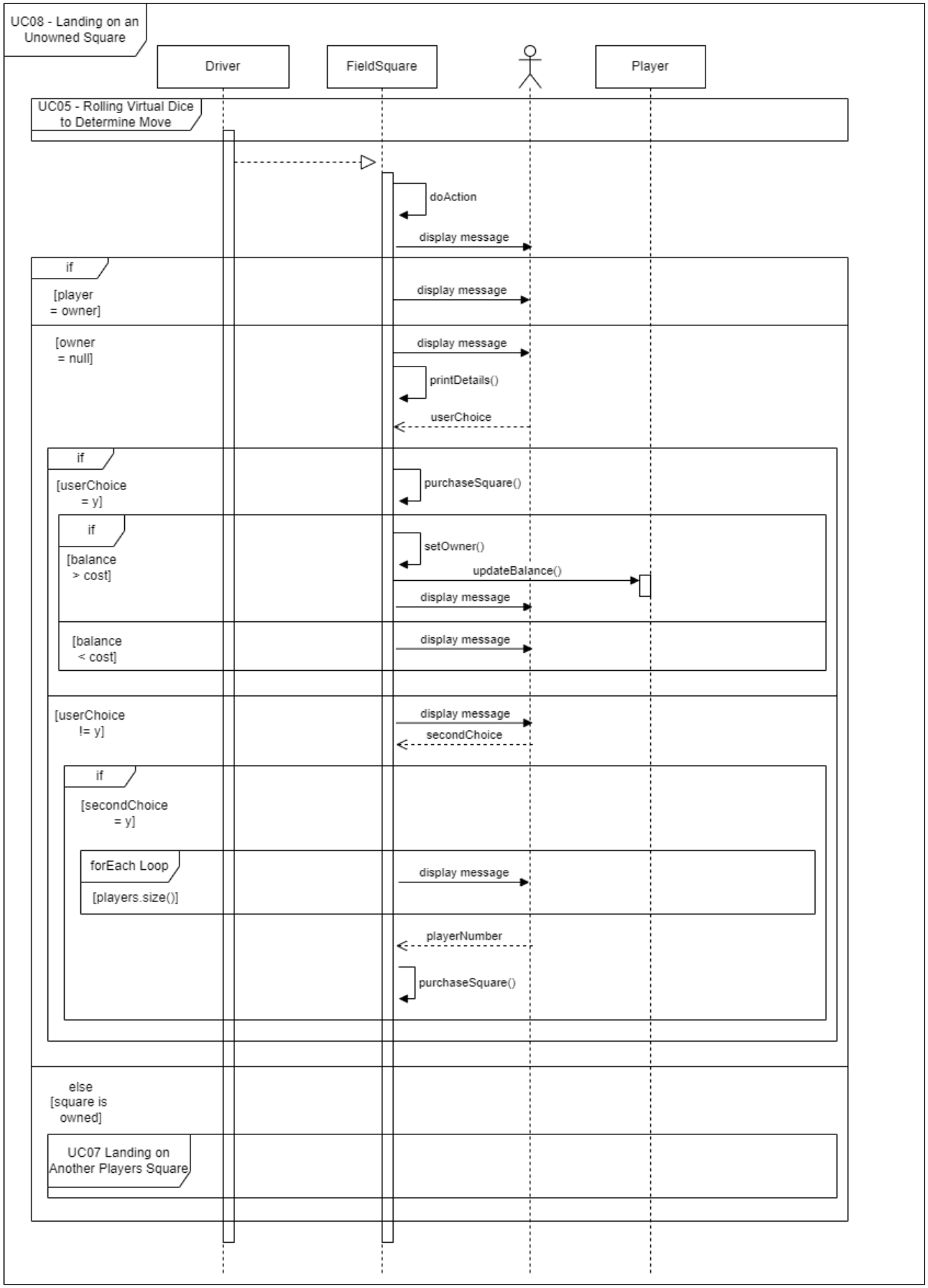
UC05 Sequence Diagram



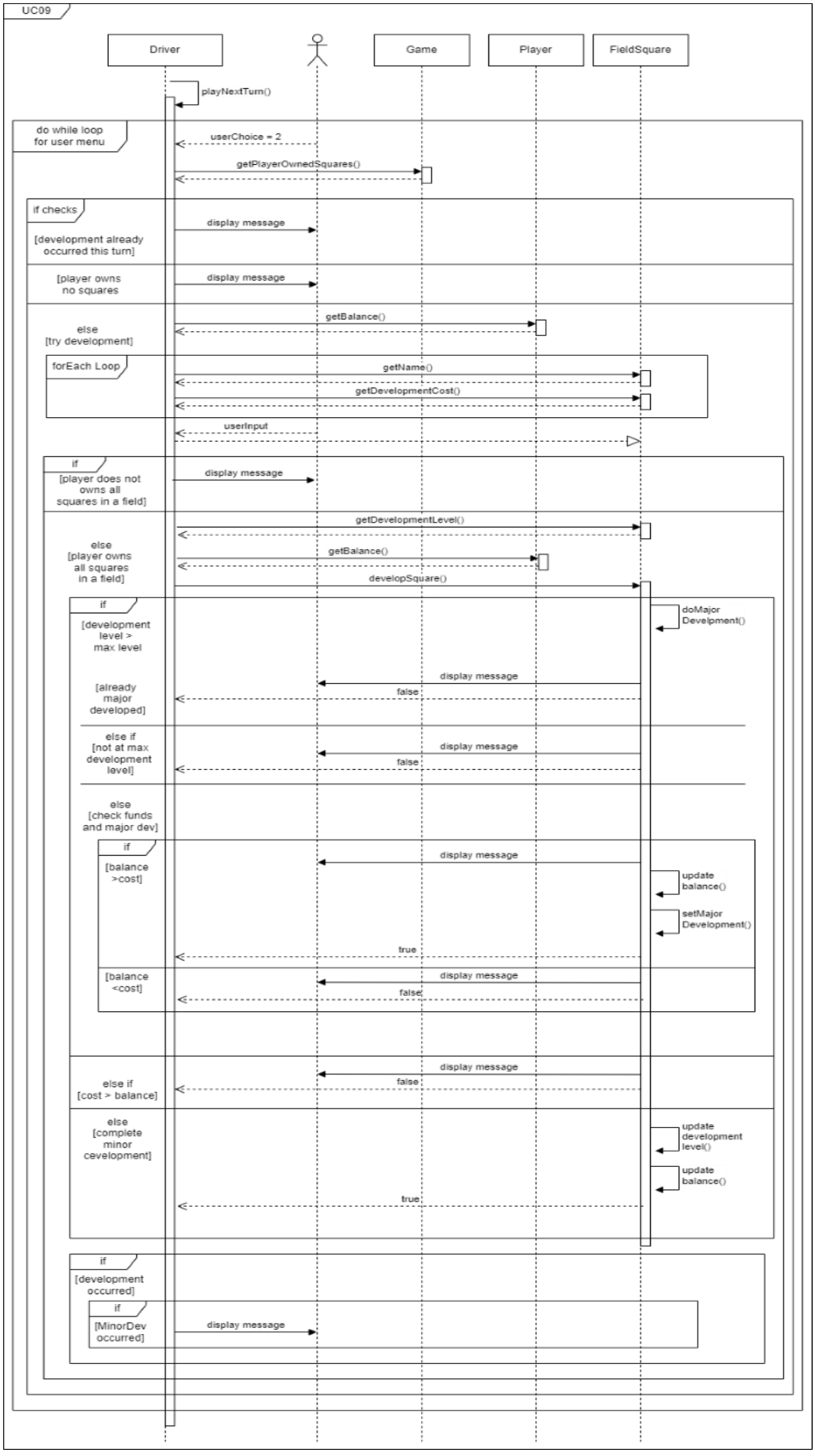
UC06 Sequence Diagram



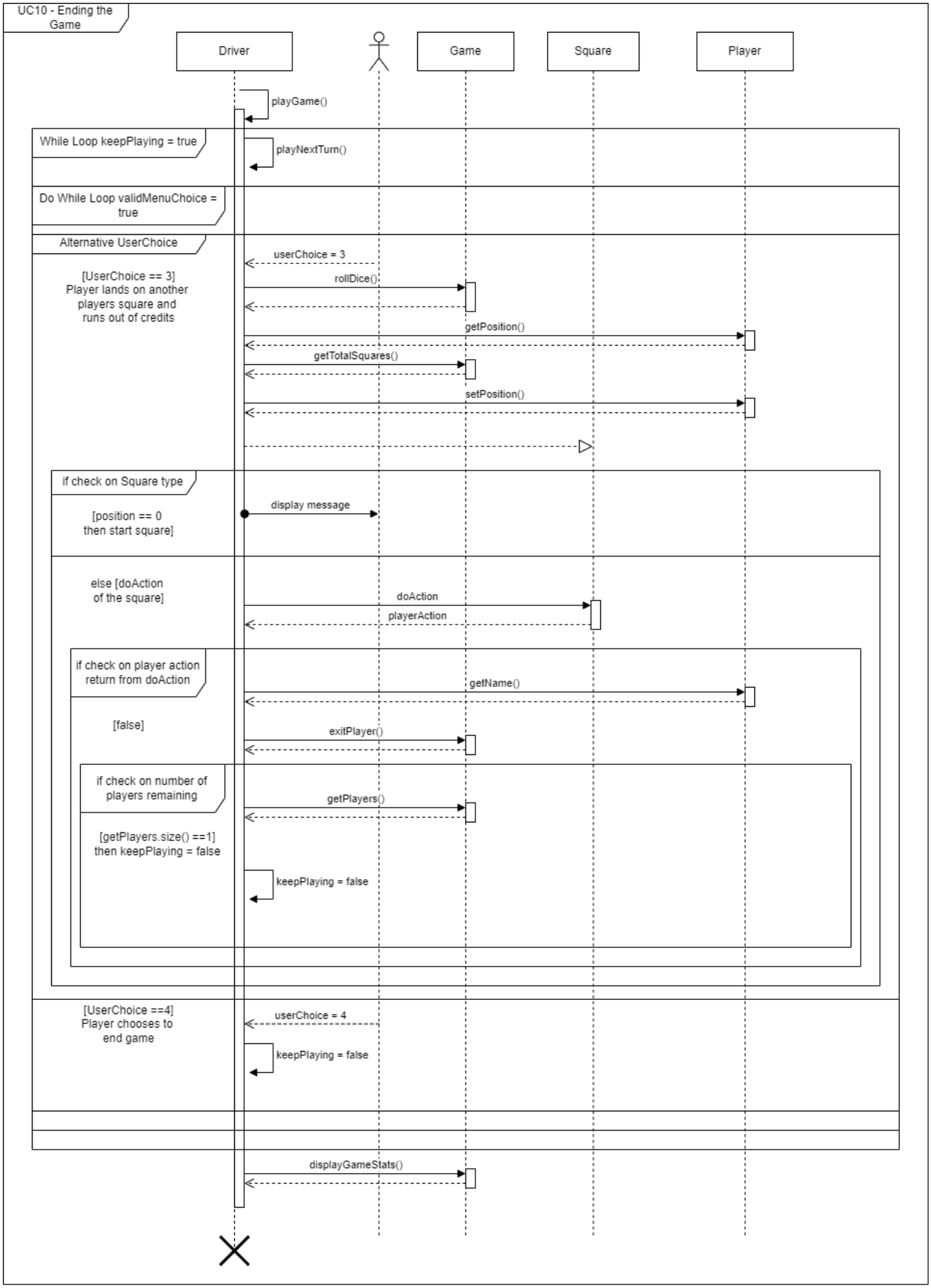
UC07 Sequence Diagram



UC08 Sequence Diagram



UC09 Sequence Diagram



UC10 Sequence Diagram

Appendix 10: User Acceptance Tables and console evidence

MAIN FLOW

Test ID	User Story ID	Test Description	Test Initialisation	Test Inputs	Test Procedure	Expected results	Passed ?
1	US01	Set up the game with a valid number of players	Launch the game application	Number of players: 2	Player launches the game. Player enters number of human players	The game should accept the number of players and move on to asking for Player 1's name	Y
2	US02	Entering unique player names	Launch game and insert number of players	Player 1 Name: Paddy Player 2 Name: Rebecca	Player inserts unique player names for all players	The game should accept entered names and begin game with Player 1's turn	Y
3	US03	Game rules displayed before player allocation	Launch the game application	N/A	Player launches the game	Upon launching the game, the rules are displayed	Y
4	US04	Turn-based order should be based on entry order of player names	Launch game and enter number of players and each player's name	Number of players: 2 Player 1 Name: Paddy Player 2 Name: Rebecca	Player launches the game. Player enters number of human players. Player inserts unique player name for all players. Play the game for 3 turns	The game should begin with Player 1's turn. After Player 1's turn, it should be Player 2's turn, and so on	Y
5	US05	Rolling virtual dice to determine movement	Current player's turn has begun and ready for input	Player selects option to roll the dice – Player input: 3	Player selects option 3 by inputting the number 3 and pressing enter	The result of the dice roll should be displayed to the player, indicating the total value obtained from both dice	Y
6	US06	Updating player position on game board	Current player's turn has begun and ready for input	Player selects option to roll the dice – Player input: 3	Player selects option 3 by inputting the number 3 and pressing enter	The game interface should display the player's current position on the game board as well as their updated position based on the dice roll	Y

Test ID	User Story ID	Test Description	Test Initialisation	Test Inputs	Test Procedure	Expected results	Passed ?
7	US07	Players should start the game with 1000 credits	Launch game and enter number of players and each player's name	Number of players: 2 Player 1 Name: Paddy Player 2 Name: Rebecca	Player launches the game. Player enters number of human players. User inserts unique player name for all players	The game interface should display a user balance of 1000 credits upon each player's first turn	Y
8	US08	Players should be presented with obligations and opportunities based on landed square	Player has rolled the dice	Player selects option to roll the dice – Player input: 3	Player selects option 3 by inputting the number 3 and pressing enter	Player should be presented with either an obligation or an opportunity based on the square they land on	Y
9	US09	Choosing One Player Selected Action per Turn. Option 1 – Show all the squares that you own	Player's turn has begun, and they are displayed with their options	Player selects option to show all squares that they own – Player input: 1	Player selects option 1 by inputting the number 1 and pressing enter	Player should be presented with all of the squares that they currently own	Y
10	US09	Choosing One Player Selected Action per Turn. Option 2 – Develop a square that you own	Player's turn has begun, and they are displayed with their options	Player selects option to develop a square that they own - Player input: 2	Player selects option 2 by inputting the number 2 and pressing enter	Player should be presented with their current balance, the squares that they own and asked to pick one to develop	Y

Test ID	User Story ID	Test Description	Test Initialisation	Test Inputs	Test Procedure	Expected results	Passed ?
11	US10	Unpurchased squares offered to other players	Player has landed on an unowned square and is presented with option to purchase	Player input: 'N'	Player types 'N' to indicate that they don't want to purchase a square	Player decision should be displayed, and game should ask if any other players wish to purchase this square	Y
12	US11	Notifying Players of Resource Changes – buying a square, sufficient credits	Player has rolled dice and landed on a square that is not owned, with sufficient credits to proceed with purchase	Player input: 'Y'	Player types 'Y' to proceed with purchase	Player informed they have purchased square and updated balance is displayed	Y
13	US11	Notifying Players of Resource Changes – landing on someone else's square, sufficient credits	Player has rolled dice and landed on a square that is owned by another player, with sufficient credits to complete transaction	N/A	Play game until player lands on square owned by another player	Player informed they have landed on another player's square and updated balance is displayed	Y
14	US12	Designing Game Board with 12 Squares and Four Separate Fields	Current player's turn has begun and ready for input	Player selects option to roll the dice – Player input: 3	Player selects option to roll the dice, and continues to select option to roll dice each turn until passing start square	The result of the dice roll should be displayed to the player, with original square position and new square position displayed to player - verifying mathematically that board is 12 squares	Y

Test ID	User Story ID	Test Description	Test Initialisation	Test Inputs	Test Procedure	Expected results	Passed ?
15	US12	Designing Game Board with 12 Squares and Four Separate Fields	Current player's turn has begun and ready for input	Player selects option to roll the dice - Player input: 3	Player selects option to roll the dice, and continues to select option to roll dice each turn until all 4 fields have been displayed	The details of each square landed on, including unique name and field information, are displayed to the player, verifying that there are unique squares and 4 fields in the game	Y
16	US13	Earning Credits by Landing on or Passing the Start Square	Player has taken turn and selects option to roll dice	Player selects option to roll the dice – Player input: 3	Player selects option to roll the dice, and continues to select option to roll dice each turn until landing on, or passing start square	Player informed that their balance has increased	Y
17	US14	Field Area with Lower Resource Cost for Strategic Investment	Current player's turn has begun and ready for input	Player selects option to roll the dice – Player input: 3	Player selects option to roll the dice, and continues to select option to roll dice each turn until all 4 fields have been displayed	The details of each square landed on, including field information and resource cost, are displayed to the player, verifying that there is one field area with the lowest resource cost	Y
18	US15	Field Area with Higher Resource Cost for Strategic Investment	Current player's turn has begun and ready for input	Player selects option to roll the dice – Player input: 3	Player selects option to roll the dice, and continues to select option to roll dice each turn until all 4 fields have been displayed	The details of each square landed on, including field information and resource cost, are displayed to the player, verifying that there is one field area with the highest resource cost	Y

Test ID	User Story ID	Test Description	Test Initialisation	Test Inputs	Test Procedure	Expected results	Passed ?
19	US16	Developing Areas within Owned Fields – squares/fields owned as required, sufficient credits	Player has taken turn and selects option to develop a square that they own from presented options	Player selects option to develop a square that they own - Player input: 2	Player selects Option 2 by inputting the number 2 and pressing enter	Player informed they have developed square and updated balance is displayed	Y
20	US17	Completing Major Development in Owned Fields – required developments already completed	The game board is in a state such that at least one player has performed three developments on at least one square of a field they own.	Player selects option to develop a square that they own - Player input: 2, then selects numerical value of the square to be developed	Player selects Option 2 by inputting the number 2 and pressing enter, and inputs numerical value of the square to be developed and pressing enter	If all criteria are met, the game confirms that a major development has taken place and shows the new resource generation of the square	Y
21	US18	Paying Credits for Landing on Another Player-Controlled Area	Game where the board has at least one square owned by a player	Player selects option to roll the dice – Player input: 3	Play game until the criteria are met	The game confirms that Player 1 has landed on Player 2's square; it specifies the fine and shows both players' updated balances	Y
22	US19	Exiting the Game due to Insufficient Credits	Game where the board has at least one square owned by a player	Player selects option to roll the dice – Player input: 3	Play until a player lands on another player's owned square and has insufficient credits to pay the fine due	Player 1's balance is reduced to 0; Player 1 is then exited from the game and forfeits all their currently owned squares. Player 2's resources are increased by an amount equal to Player 1's remaining balance prior to landing on Player 2's square	Y

Test ID	User Story ID	Test Description	Test Initialisation	Test Inputs	Test Procedure	Expected results	Passed ?
23	US20	Ending the Game When a Player Chooses to Stop Playing	Player has taken turn and selects option to quit game	Player selects option to quit game – Player input: 4	Player selects Option 4 by inputting the number 4 and pressing enter	The game will end and a message displays on the screen confirming the game has ended	Y
24	US21	Generating End-of-Game Statistics Summary	Player has taken turn and selects option to quit game	Player selects option to quit game – Player input: 4	Player selects Option 4 by inputting the number 4 and pressing enter	The game displays the players' names and their game-end balances	Y

ALTERNATIVE FLOW

Test ID	User Story ID	Test Description	Test Initialisation	Test Inputs	Test Procedure	Expected results	Passed ?
25	US01	Set up the game with an invalid number of players (low)	Launch the game application	Number of players: 1	User launches the game. User enters number of human players	The game should display an error message and ask player to enter valid number	Y
26	US01	Set up the game with an invalid number of players (high)	Launch the game application	Number of players: 5	User launches the game. User enters number of human players	The game should display an error message and ask player to enter valid number	Y
27	US02	Entering non-unique player names	Launch game and insert number of players	Player 1 Name: Paddy Player 2 Name: Paddy	User inserts non-unique player names	The game should display an error message stating that player names must be unique and ask user to enter a unique name	Y
28	US02	Entering unique player names – low invalid	Launch game and insert number of players	Player 1 Name: (blank)	User presses enter with blank player name	System should throw an error and say player name must be between 1-20 characters long	Y
29	US02	Entering unique player names – high invalid	Launch game and insert number of players	Player 1 Name: Aaaaaaaaaaaaaaaaaaaaa.	Player enters 21-character player name and presses enter	System should throw an error and say player name must be between 1-20 characters long	Y
30	US09	Choosing One Player Selected Option per Turn with invalid number	Player's turn has begun, and they are displayed with their options.	Player input: 5	Player inputs 5.	Error message should be displayed, menu options displayed again for re-entry.	Y

Test ID	User Story ID	Test Description	Test Initialisation	Test Inputs	Test Procedure	Expected results	Passed ?
31	US11	Notifying Players of Resource Changes – buying a square – insufficient credits	Player has rolled dice and landed on a square that is not owned	Player input: Y	Player types 'Y' to proceed with purchase	Error message displays and options are redisplayed	Y
32	US11	Notifying Players of Resource Changes – landing on someone else's square, insufficient credits	Player has rolled dice and landed on a square that is owned by another player, with insufficient credits to complete transaction	N/A	Play game until player lands on square owned by another player	Player informed they have landed on another player's square and as insufficient credits to proceed, they exit the game	Y
33	US16	Developing Areas within Owned Fields – no squares are owned	Player has taken turn and selects option to develop a square that they own from presented options	Player input: 2	Player selects option 2 by inputting the number 2 and pressing enter	Error message displays and options are redisplayed	Y
34	US16	Developing Areas within Owned Fields – not all required squares are owned within field	Player has taken turn and selects option to develop a square, and chooses square by entering the appropriate number on the console	Player input: 2, numerical value of chosen square to develop	Player selects option 2 by inputting the number 2 and pressing enter, and choosing the square by inputting the numerical value from the displayed list of squares owned	Error message displays and options are redisplayed	Y
35	US16	Developing Areas within Owned Fields – not enough resources	Player has taken turn and selects option to develop a square that they own from presented options	Player input: 2	Player selects option 2 by inputting the number 2 and pressing enter	Error message displays and options are redisplayed	Y

Test ID	User Story ID	Test Description	Test Initialisation	Test Inputs	Test Procedure	Expected results	Passed ?
36	US16	Developing Areas within Owned Fields – enters string value for option rather than number	Player has taken turn and selects option to develop a square that they own from presented options	Player input: "Two"	Player selects option 2 by inputting the word "Two" and pressing enter	Error message displays and options are redisplayed	Y
37	US16	Developing Areas within Owned Fields – enters string value for chosen square rather than number	Player has taken turn and selects option to develop a square that they own from presented options,	Player input: 2, "string"	Player selects option 2 by inputting the number 2 and pressing enter, and then enters a "string" value for the square	Error message displays and options are redisplayed	Y
38	US17	Completing Major Development in Owned Fields – verifying cannot proceed as player has not previously completed 3 developments	The game board is in a state such that at least one player owns all squares in a field but has not developed any of the field's squares to level 4	Player input: 2, numerical value of the square to be developed	Player selects Option 2 by inputting the number 2 and pressing enter, and inputs numerical value of the square to be developed and pressing enter	The 'major dev' column in the player square ownership grid shows as false	Y

```
Enter number of players:
2
Player 1 Enter name:
```

Test 1: Game asks for player name after valid player number input

```
Enter number of players:
2
Player 1 Enter name:
Paddy
Player 2 Enter name:
Rebecca

<Player 1> Paddy's turn! <Balance:
------------------------------
Pick an action and press enter:
1. Show all the squares that you o
2. Develop a square that you own
3. Roll Dice
4. Quit Game
```

Test 2: Entering unique player names

```
**********************************************************************
Welcome to Green Priority

Game Rules
----------
The aim of the game is to be the last player standing!
Each player will start with 1000 carbon credits.
Purchase squares - when other players land on these squares they will pay a fine. You will recieve their credits.
You can develop squares to increase the fines when other players land on them.
There are three levels of development ranging from 1-4, and a major development. A major development is only possible when the square is at max
You can only develop a square once you own all squares belonging to that field.
If a player declines to buy a square, other players will have the option to purchase it - even if its not their turn.

If one player quits, the game will end for all players.
When a player runs out of carbon credits they will exit the game. The last player standing wins!

Good luck!
**********************************************************************
```

Test 3: Game rules displayed before player allocation

```
<Player 1> Paddy's turn! <Balance: 2000>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
3
You rolled 1 - that makes 1
You were on square 0, you have moved to square: 1
Paddy you have landed on square Fusion (Belonging to Field NUCLEAR)
Square name    Field          Dev Level      Major Dev?     Charges        Purchase Cost  Development Cost
Fusion         NUCLEAR        1              false          250            500            250

You currently have 2000 credits
**This square is available for purchase, would you like to buy it for 500? Y/N**
n
You have declined to purchase.. Do any other players wish to purchase this square? Y/N
n

<Player 2> Rebecca's turn! <Balance: 2000>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
3
You rolled 2 - that makes 2
You were on square 0, you have moved to square: 2
Rebecca you have landed on square Fission (Belonging to Field NUCLEAR)
Square name    Field          Dev Level      Major Dev?     Charges        Purchase Cost  Development Cost
Fission        NUCLEAR        1              false          300            500            250

You currently have 2000 credits
**This square is available for purchase, would you like to buy it for 500? Y/N**
n
You have declined to purchase.. Do any other players wish to purchase this square? Y/N
n

<Player 1> Paddy's turn! <Balance: 2000>
```

Test 4: Turn-based order should be based on entry order of player names

```
<Player 1> Paddy's turn! <Balance: 2000>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
3
You rolled 5 & 4 - that makes 9
You were on square 0, you have moved to square: 9
```

Test 5 and 6: Rolling virtual dice to determine movement and updating player position

```
Enter number of players:
2
Player 1 Enter name:
Paddy
Player 2 Enter name:
Rebecca

<Player 1> Paddy's turn! <Balance: 1000>
--------------------------------
```

```
<Player 2> Rebecca's turn! <Balance: 1000>
```

Test 7: Players should start the game with 1000 credits

```
<Player 1> Paddy's turn! <Balance: 1000>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
3
You rolled 2 & 1 - that makes 3
You were on square 0, you have moved to square: 3
Paddy you have landed on square Rain (Belonging to Field HYDROPOWER)
Square name     Field          Dev Level      Major Dev?     Charges        Purchase Cost  Development Cost
Rain            HYDROPOWER     1              false          90             300            150

You currently have 1000 credits
**This square is available for purchase, would you like to buy it for 300? Y/N**
```

Test 8: Players should be presented with obligations and opportunities based on landed square

```
<Player 2> Rebecca's turn! <Balance: 900>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
1
Square name     Field          Dev Level      Major Dev?     Charges        Purchase Cost  Development Cost
Solar Panels    SOLAR          1              false          30             100            50
```

Test 9: Choosing One Player Selected Action per Turn– Show all the squares that you own

```
<Player 1> Seamus's turn! <Balance: 1550>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
2
Your current balance is 1550 carbon credits
These are the squares you own, pick one to develop:
0. Fusion development cost: 250
1. Fission development cost: 250
```

Test 10: Choosing One Player Selected Action per Turn- option to develop a square you own

```
<Player 1> Paddy's turn! <Balance: 1000>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
3
You rolled 5 & 2 - that makes 7
You were on square 0, you have moved to square: 7
Paddy you have landed on square Solar Panels (Belonging to Field SOLAR)
Square name    Field          Dev Level      Major Dev?     Charges        Purchase Cost  Development Cost
Solar Panels   SOLAR          1              false          30             100            50

You currently have 1000 credits
**This square is available for purchase, would you like to buy it for 100? Y/N**
n
You have declined to purchase.. Do any other players wish to purchase this square? Y/N
```

Test 11: Unpurchased squares offered to other players

```
<Player 2> Rebecca's turn! <Balance: 1000>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
3
You rolled 6 & 5 - that makes 11
You were on square 0, you have moved to square: 11
Rebecca you have landed on square Windfarm (Belonging to Field WIND)
Square name    Field          Dev Level      Major Dev?     Charges        Purchase Cost  Development Cost
Windfarm       WIND           1              false          95             250            125

You currently have 1000 credits
**This square is available for purchase, would you like to buy it for 250? Y/N**
y
****************************************
Rebecca, you have purchased Windfarm
Your new balance is 750 carbon credits
****************************************
```

Test 12: Notifying Players of Resource Changes – buying a square, sufficient credits

```
<Player 1> Seamus's turn! <Balance: 2000>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
3
You rolled 2 - that makes 2
You were on square 0, you have moved to square: 2
Seamus you have landed on square Fission (Belonging to Field NUCLEAR)
********************************************************************************
Square is owned by Bart, you must pay a fine of 300 carbon credits
Your balance is now: 1700
The owner (Bart's) balance has increased from 240 to 540 (+300) carbon credits
********************************************************************************
```

Test 13: Notifying Players of Resource Changes – landing on someone else's square, sufficient credits

```
<Player 2> Dermot's turn! <Balance: 2000>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
3
You rolled 1 & 2 - that makes 3
You were on square 9, you have moved to square: 0
For passing the start square, you recieve <100> carbon credits!
Your new balance has increased from 2000 to 2100
You have landed on the start square
```

Test 14: Designing Game Board with 12 Squares

```
Seamus you have landed on square Solar Panels (Belonging to Field SOLAR)
Square name    Field          Dev Level      Major Dev?     Charges        Purchase Cost  Development Cost
Solar Panels   SOLAR          1              false          30             100            50
```

```
Dermot you have landed on square Ocean (Belonging to Field HYDROPOWER)
Square name    Field          Dev Level      Major Dev?     Charges        Purchase Cost  Development Cost
Ocean          HYDROPOWER     1              false          130            400            200
```

```
Seamus you have landed on square Fission (Belonging to Field NUCLEAR)
Square name    Field          Dev Level      Major Dev?     Charges        Purchase Cost  Development Cost
Fission        NUCLEAR        1              false          300            500            250
```

```
Seamus you have landed on square Windmill (Belonging to Field WIND)
Square name    Field          Dev Level      Major Dev?     Charges        Purchase Cost  Development Cost
Windmill       WIND           1              false          100            200            100
```

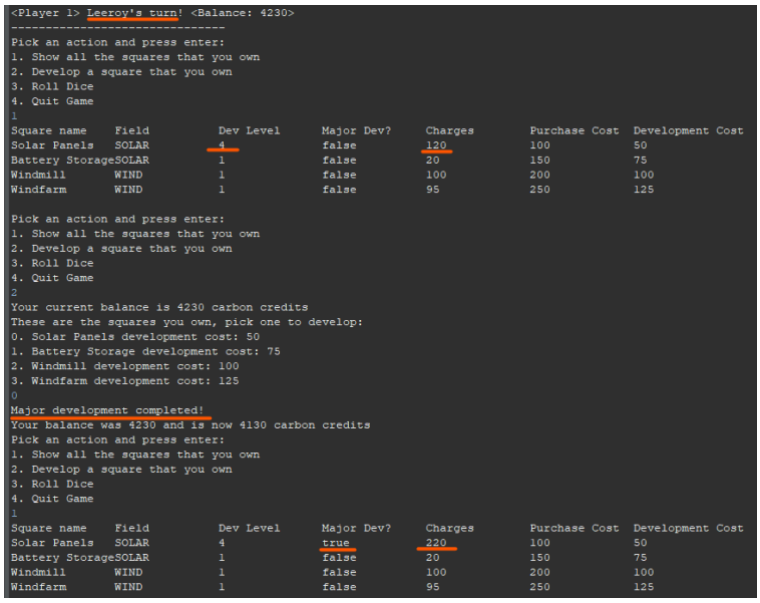
Test 15,17,18: Verify four unique fields exist, one with lowest and one with highest resource cost.

```
<Player 2> Dermot's turn! <Balance: 2000>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
3
You rolled 1 & 2 - that makes 3
You were on square 9, you have moved to square: 0
For passing the start square, you recieve <100> carbon credits!
Your new balance has increased from 2000 to 2100
You have landed on the start square
```

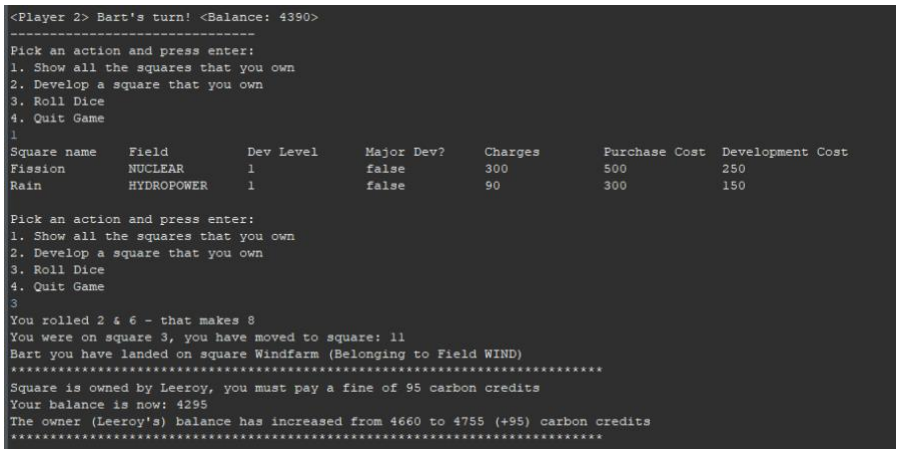
Test 16: Earning Credits by Landing on or Passing the Start Square

```
<Player 1> Seamus's turn! <Balance: 1550>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
2
Your current balance is 1550 carbon credits
These are the squares you own, pick one to develop:
0. Fusion development cost: 250
1. Fission development cost: 250
0
Square successfully developed from level 1 to level 2
Your balance was 1550 and is now 1300 carbon credits
```

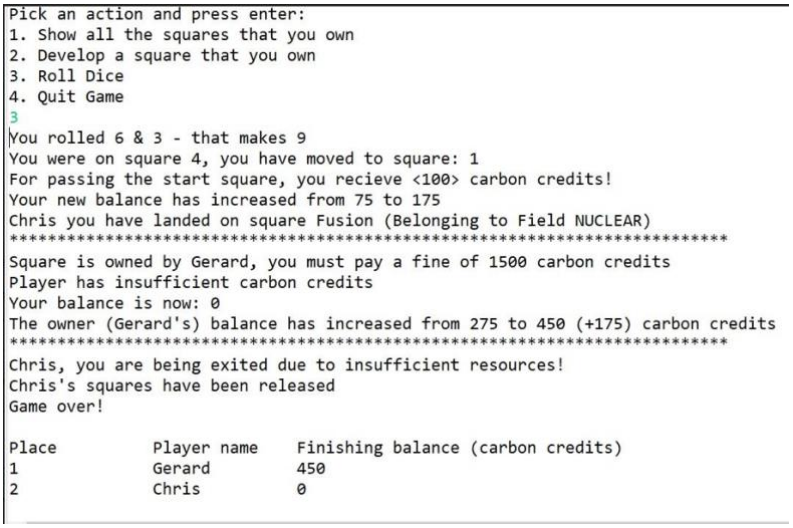
Test 19: Developing Areas within Owned Fields



Test 20: Completing Major Development in Owned Fields – required developments already completed



Test 21: Paying Credits for Landing on Another Player-Controlled Area



Test 22 and test 32: Exiting the Game due to Insufficient Credits

```
<Player 2> Bart's turn! <Balance: 4170>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
4
Quitting the game..
Game over!
```

Test 23: Ending the Game When a Player Chooses to Stop Playing

```
<Player 2> Bart's turn! <Balance: 4170>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
4
Quitting the game..
Game over!
Place          Player name    Finishing balance (carbon credits)
1              Leeroy         4230
2              Bart           4170
```

Test 24: Generating End-of-Game Statistics Summary

```
Enter number of players:
1
You need at least two players to play!
Enter number of players:
```

Test 25: Invalid number of players (low)

```
Enter number of players:
5
There is a maximum player limit of 4
Enter number of players:
```

Test 26: Invalid number of players (high)

```
Enter number of players:
2
Player 1 Enter name:
Paddy
Player 2 Enter name:
Paddy
duplicate name, enter a unique name! Try again..
Player 2 Enter name:
```

Test 27: Duplicate player names

```
Enter number of players:
2
Player 1 Enter name:

Exception in thread "main" java.lang.IllegalArgumentException: Name must be between land 20 characters long
        at game.Player.setName(Player.java:41)
        at game.Player.<init>(Player.java:20)
        at game.Game.setupPlayers(Game.java:129)
        at game.Driver.main(Driver.java:70)
```

Test 28: Blank player name entered

```
Enter number of players:
2
Player 1 Enter name:
Aaaaaaaaaaaaaaaaaaaa
Exception in thread "main" java.lang.IllegalArgumentException: Name must be between land 20 characters long
        at game.Player.setName(Player.java:41)
        at game.Player.<init>(Player.java:20)
        at game.Game.setupPlayers(Game.java:129)
        at game.Driver.main(Driver.java:70)
```

Test 29: Player name of 21 characters entered

```
<Player 1> Leeroy's turn! <Balance: 1000>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
5
Invalid option, try again!
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
```

Test 30: Invalid option choice

```
<Player 1> Leeroy's turn! <Balance: 100>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
3
You rolled 2 - that makes 2
You were on square 0, you have moved to square: 2
Leeroy you have landed on square Fission (Belonging to Field NUCLEAR)
Square name     Field          Dev Level      Major Dev?     Charges        Purchase Cost  Development Cost
Fission         NUCLEAR        1              false          300            500            250

You currently have 100 credits
**This square is available for purchase, would you like to buy it for 500? Y/N**
y
Leeroy, you dont have sufficient resources to purchase this square
Your balance: 100, Cost to buy: 500carbon credits
```

Test 31: Insufficient credits to purchase square

```
<Player 1> Leeroy's turn! <Balance: 100>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
2
You dont currently have any squares to develop!
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
```

Test 33: Player tries to develop squares when none are owned

```
<Player 1> Leeroy's turn! <Balance: 800>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
2
Your current balance is 800 carbon credits
These are the squares you own, pick one to develop:
0. Fission development cost: 250
0
You must own all squares in the field before you can do a development
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
```

Test 34: Player tries to develop field without owning all required squares

```
<Player 1> Seamus's turn! <Balance: 200>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
1
Square name    Field          Dev Level      Major Dev?     Charges        Purchase Cost  Development Cost
Fusion         NUCLEAR        1              false          250            500            250
Fission        NUCLEAR        1              false          300            500            250

Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
2
Your current balance is 200 carbon credits
These are the squares you own, pick one to develop:
0. Fusion development cost: 250
1. Fission development cost: 250
0
You have insufficient resources to develop this square
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
```

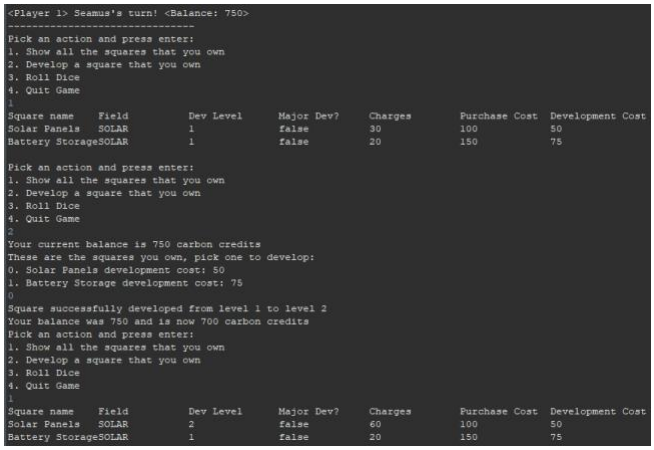
Test 35: Developing Areas within Owned Fields – not enough resources

```
<Player 1> Seamus's turn! <Balance: 1250>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
two
Invalid input type.. Must be a number
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
```

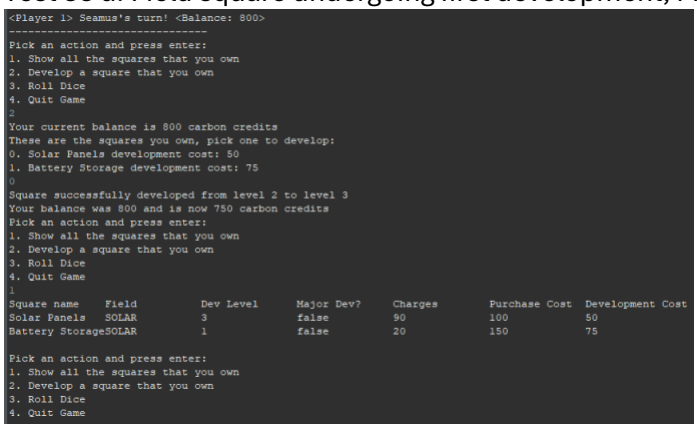
Test 36: Entering string input for player action instead of numeric input

```
<Player 1> Seamus's turn! <Balance: 1000>
------------------------------
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
2
Your current balance is 1000 carbon credits
These are the squares you own, pick one to develop:
0. Fusion development cost: 250
1. Fission development cost: 250
String
Invalid selection, please enter a valid number
Pick an action and press enter:
1. Show all the squares that you own
2. Develop a square that you own
3. Roll Dice
4. Quit Game
```

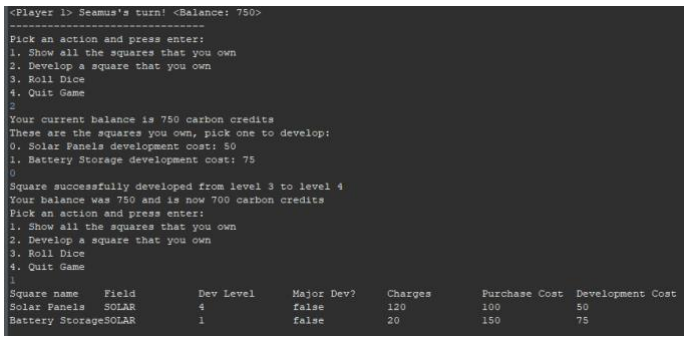
Test 37: Entering string input for development square instead of numeric input.



Test 38 a: Field square undergoing first development; Major dev? shows as false.



Test 38 b: Field square undergoing 2nd development; Major dev? shows as false.



Test 38 c: Field square undergoing 3rd development; Major dev? shows as false.

Appendix 11:

Junit Testing

UC01_test.java

```java
package game;

import static org.junit.jupiter.api.Assertions.*;
import java.io.InputStream;
import java.io.ByteArrayInputStream;
import java.util.Scanner;
import java.util.ArrayList;
import java.util.NoSuchElementException;
//import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;

class UC01_test {

    // Set up basic universal test data
    public final int MAX_PLAYERS = 4;
    public final int MIN_PLAYERS = 2;
    public final int MIN_PLAYERS_INVALID = 1;
    public final int START_POSITION = 0;
    public final int START_BALANCE = 1000;

    // Set up universal static objects for Game constructors
    //Set up a square arrayList
    private static ArrayList<Square> squares = new ArrayList<Square>();

    // Create a board for the game constructor
    private static Board board = new Board(squares);

    // Create a dice arrayList for Game constructor
    private static ArrayList<Die> dice = new ArrayList<Die>();


    /**
     * @throws java.lang.Exception
     */
    @BeforeAll
    public static void setUp() throws Exception {

        //Add  minimal squares to the squares arrayList that was used for the board
        squares.add(new StartSquare(Type.START, "START", 100));

        // Add die to the dice arrayList for the game constructor (add i die minimum in order to satisfy the setDice() method
        dice.add(new Die(6));

    }
```

Fig 1: Pre-test imports, variable definitions, and object initialisation

```java
/**
 * Test method for Game constructor with valid arguments
 */
@Test
void testGameConstructorValidArgs() {

    // Create game object for testing
    Game game = new Game(board, MAX_PLAYERS, dice);

    //start internal method testing
    assertEquals(board, game.getBoard());
    assertEquals(MAX_PLAYERS, game.getMaxPlayers());
    assertEquals(dice,game.getDice());

    // test that the constructor has initialised empty players and playersExited arrayLists
    assertEquals(0, game.getPlayers().size());
    assertEquals(0, game.getPlayersExited().size());

}
```

Fig 2: Test of Game class constructor with valid args.

```java
@Test
void testGameConstructorInvalidArgs() {
    // Can only test setMaxplayers() and  setDice() as there is no input validation in setBoard() or Board class
    // setMaxplayers() and  setDice() were not tested separately because they need a valid Game object to be accessed


    // Give an invalid argument for max number of players in the Game constructor
    // Test that it throws an IllegalArgumentException
    assertThrows(IllegalArgumentException.class, ()-> new Game(board, MIN_PLAYERS_INVALID, dice));

    // Test that the exception thrown contains the expected string message
    Exception e = assertThrows(IllegalArgumentException.class, ()-> new Game(board, MIN_PLAYERS_INVALID, dice));
    String message = e.getMessage();
    assertEquals("Game must have more than 1 player", message);

    // Give a null argument for the Dice arrayList  in the Game constructor
    // Test that it throws an IllegalArgumentException
    assertThrows(IllegalArgumentException.class, ()-> new Game(board, MAX_PLAYERS, null));

    // Test that the exception thrown contains the expected string message
    Exception e2 = assertThrows(IllegalArgumentException.class, ()-> new Game(board, MAX_PLAYERS, null));
    String message2 = e2.getMessage();
    assertEquals("Dice cant be null or < 1", message2);

    // Give an empty Dice arrayList as argument for the Dice arrayList  in the Game constructor
    // Test that it throws an IllegalArgumentException
    ArrayList<Die> emptyDice = new ArrayList<Die>(); // contains no dice
    assertThrows(IllegalArgumentException.class, ()-> new Game(board, MAX_PLAYERS, emptyDice));

    // Test that the exception thrown contains the expected string message
    Exception e3 = assertThrows(IllegalArgumentException.class, ()-> new Game(board, MAX_PLAYERS, emptyDice));
    String message3 = e3.getMessage();
    assertEquals("Dice cant be null or < 1", message3);

}
```

Fig 3: Test of Game class constructor with invalid args.

```java
@Test
void testSetupPlayersValidArgs() {
    /*
     *  approach adapted from
     *  https://stackoverflow.com/questions/31635698/junit-testing-for-user-input-using-scanner
     *
     *  and
     *  https://stackoverflow.com/questions/12729280/junit-how-to-check-if-string-is-equal-to-one-of-two-strings
     */

    // Create pre-defined string input
    // The input is in the specified format as it must follow how the Scanner methods are called in the setUpPlayers() method
    // i.e.  number of players, then new line, then player 1 name, then new line, then player 2 name
    String mockInput = "2\nBill\nBen";

    // Convert this string to ByteArrayInputStream to be fed to the Scanner - the scanner expects an input stream.
    InputStream mockInputAsBytes = new ByteArrayInputStream(mockInput.getBytes());

    // Feed this to the scanner to simulate user input
    Scanner mockedScanner = new Scanner(mockInputAsBytes);

    Game game = new Game(board, MAX_PLAYERS, dice);
    game.setupPlayers(START_POSITION,START_BALANCE,mockedScanner);

    // Check that the game has only 2 players
    assertEquals(2,game.getPlayers().size());

    // Check that the name of the first player matches the first name supplied to the scanner
    assertTrue("Bill".equals(game.getPlayers().get(0).getName()));

    // Check that the name of the second player matches the second name supplied to the scanner
    assertTrue("Ben".equals(game.getPlayers().get(1).getName()));
}
```

Fig 4: Testing of Game class's setUpPlayer() method with valid arguments and mocking

```
@Test
void testSetupPlayersInvalidArgs() {
    // See testSetupPlayersValidArgs() method for explanation of the code
    // Player number < 2 was tested previously, so now test player number > MAX_PLAYERS
    String badPlayerNumber = "5";
    // Repeat steps as per testSetupPlayersValidArgs()
    InputStream badPlayerNumberAsBytes = new ByteArrayInputStream(badPlayerNumber.getBytes());
    Scanner mockedScanner = new Scanner(badPlayerNumberAsBytes);
    Game game = new Game(board, MAX_PLAYERS, dice);

    // The setupPlayers() method does not directly throw exceptions upon receiving invalid input
    // As such, by proxy, we test for the Scanner object throwing exceptions due to unexpected input.
    assertThrows(NoSuchElementException.class, ()-> game.setupPlayers(START_POSITION,START_BALANCE,mockedScanner));

    // Test with duplicate names
    String duplicateNames = "2\nBill\nBill";
    InputStream duplicateNamesAsBytes = new ByteArrayInputStream(duplicateNames.getBytes());
    Scanner mockedScanner2 = new Scanner(duplicateNamesAsBytes);
    assertThrows(NoSuchElementException.class, ()-> game.setupPlayers(START_POSITION,START_BALANCE,mockedScanner2));
}
```

Fig 5: Testing of Game class's setUpPlayer() method with invalid arguments and mocking

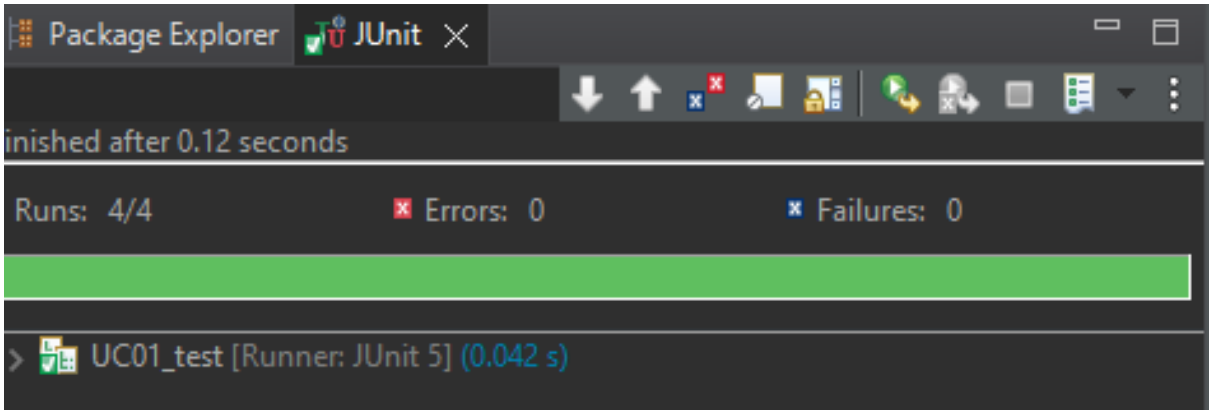
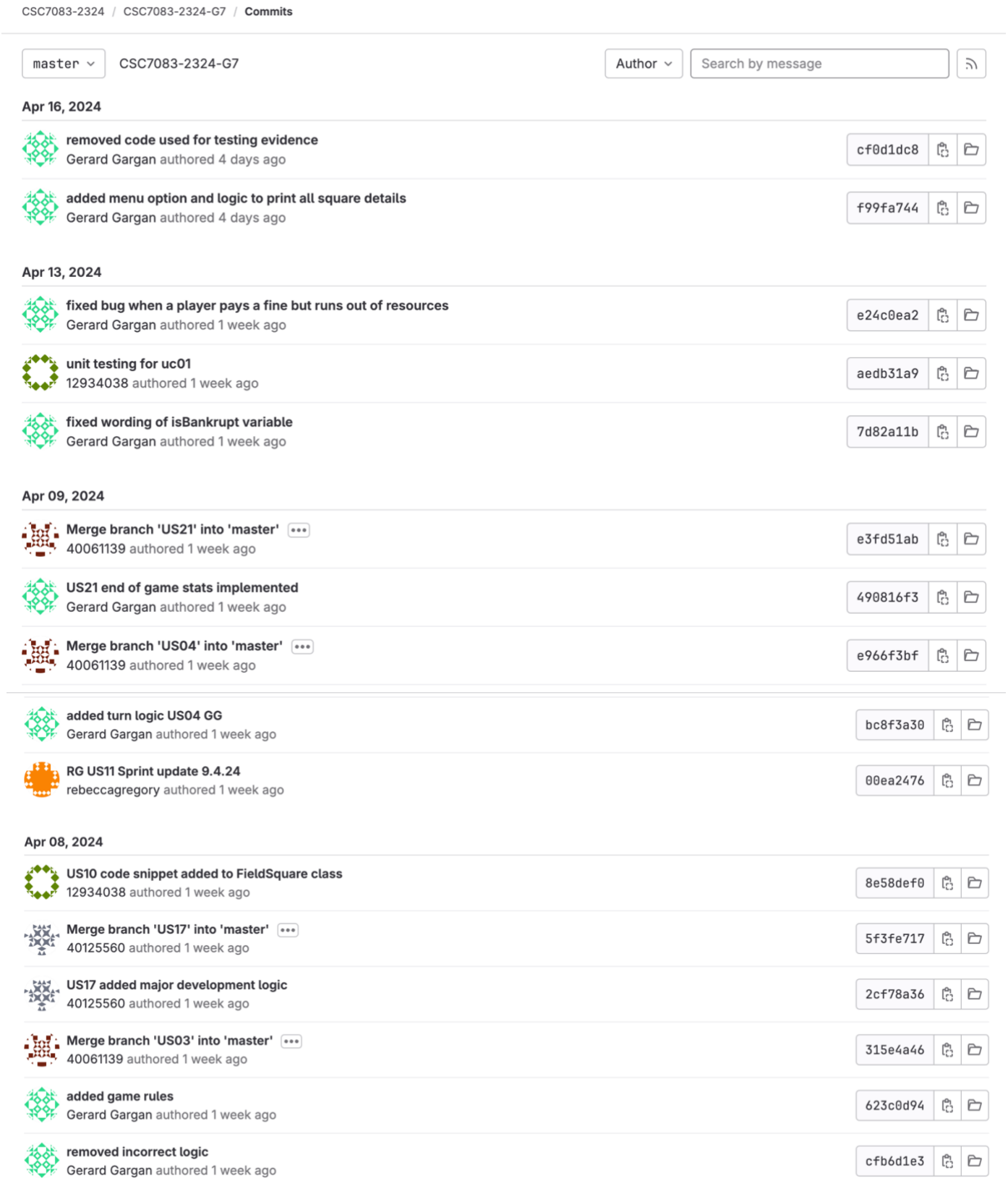


Fig 6: Proof of test success.

Appendix 12:

Gitlab Commits



Apr 06, 2024

RG US13 Update 6.4.24
rebeccagregory authored 2 weeks ago — e8a55d78

RG US07 Update 6.4.24
rebeccagregory authored 2 weeks ago — feef36f9

RG update 6.4.24
rebeccagregory authored 2 weeks ago — 3f5fa41f

Apr 05, 2024

Merge branch 'US16' into 'master'
40061139 authored 2 weeks ago — df0096f8

completed US16 method for developing a FieldSquare
Gerard Gargan authored 2 weeks ago — d0a4eb29

Merge branch 'US14US15' into 'master'
40125560 authored 2 weeks ago — 2c9deef5

US14 and US15 added - Updated values in Driver Class
40125560 authored 2 weeks ago — af163acf

Apr 04, 2024

removed incorrect logic
Gerard Gargan authored 2 weeks ago — 4b190d85

Apr 02, 2024

US08 AND US18
12934038 authored 2 weeks ago — f42832fe

Apr 01, 2024

Merge branch 'US06US09US19US20' into 'master'
40125560 authored 2 weeks ago — 776f6592

US06, US09, US19, US20 logic added
40125560 authored 2 weeks ago — 8a0066e7

Merge branch 'US05' into 'master'
40125560 authored 2 weeks ago — ad117a5f

US05 Added die class and logic
40125560 authored 2 weeks ago — 34aa4c91

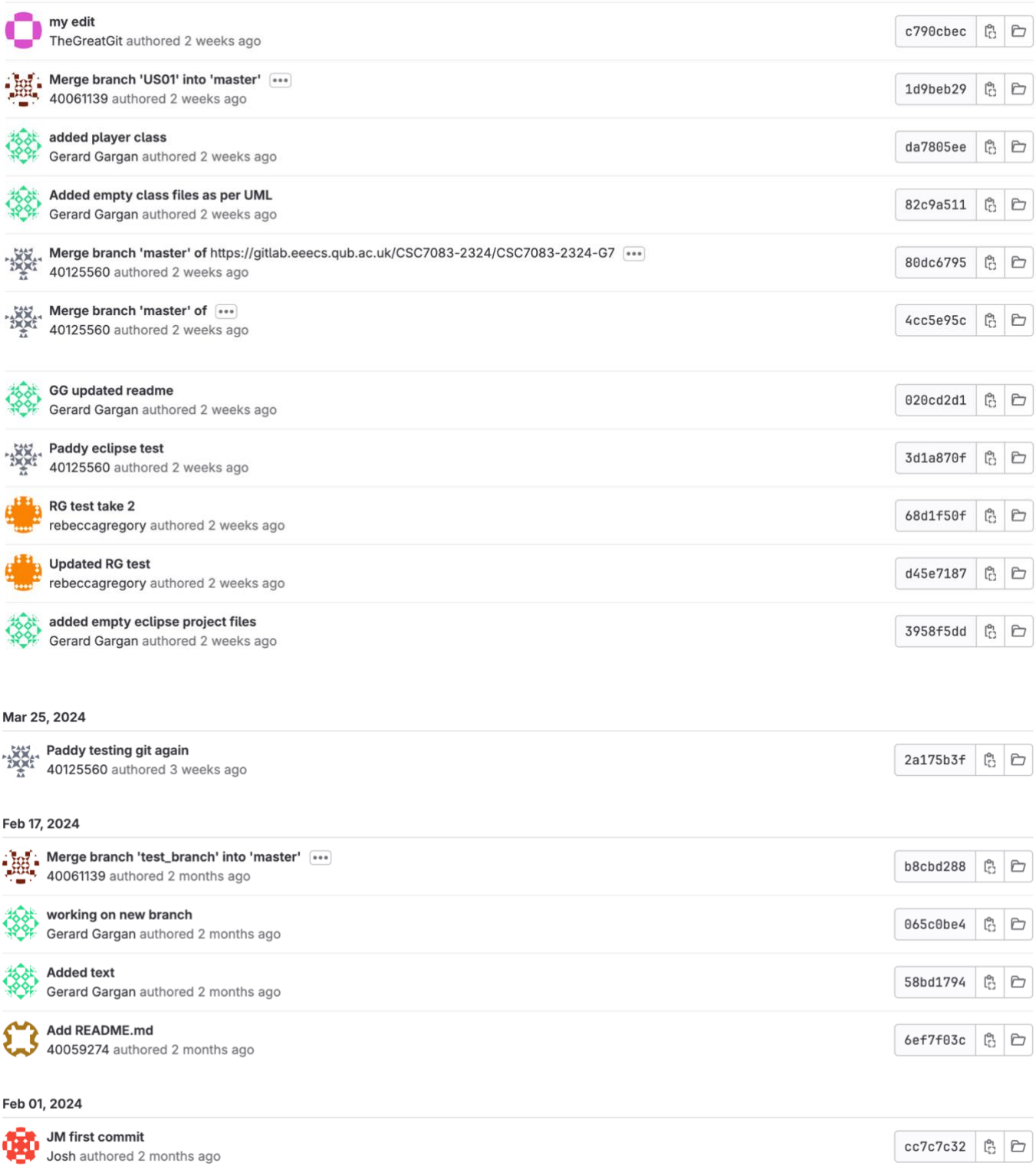
test profile change
12934038 authored 2 weeks ago — b9ee73e4

US12 Implemented Game Board and Squares
40125560 authored 2 weeks ago — e7dc0634

Merge branch 'US02' into 'master'
40061139 authored 2 weeks ago — 1d8601c4

Merge remote-tracking branch 'origin' into US02
Gerard Gargan authored 2 weeks ago — fb7e521d

added logic for setting up 2-4 players, basic game class outline completed
Gerard Gargan authored 2 weeks ago — 9b9f4d41


my edit
TheGreatGit authored 2 weeks ago
c790cbec
Merge branch 'US01' into 'master'
40061139 authored 2 weeks ago
1d9beb29
added player class
Gerard Gargan authored 2 weeks ago
da7805ee
Added empty class files as per UML
Gerard Gargan authored 2 weeks ago
82c9a511
Merge branch 'master' of https://gitlab.eeecs.qub.ac.uk/CSC7083-2324/CSC7083-2324-G7
40125560 authored 2 weeks ago
80dc6795
Merge branch 'master' of
40125560 authored 2 weeks ago
4cc5e95c
GG updated readme
Gerard Gargan authored 2 weeks ago
020cd2d1
Paddy eclipse test
40125560 authored 2 weeks ago
3d1a870f
RG test take 2
rebeccagregory authored 2 weeks ago
68d1f50f
Updated RG test
rebeccagregory authored 2 weeks ago
d45e7187
added empty eclipse project files
Gerard Gargan authored 2 weeks ago
3958f5dd
Mar 25, 2024
Paddy testing git again
40125560 authored 3 weeks ago
2a175b3f
Feb 17, 2024
Merge branch 'test_branch' into 'master'
40061139 authored 2 months ago
b8cbd288
working on new branch
Gerard Gargan authored 2 months ago
065c0be4
Added text
Gerard Gargan authored 2 months ago
58bd1794
Add README.md
40059274 authored 2 months ago
6ef7f03c
Feb 01, 2024
JM first commit
Josh authored 2 months ago
cc7c7c32

Appendix 13:

Meeting Minutes

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location: 15/01/2024 @ 8pm - Microsoft Teams

Meeting Objectives: Initial meeting to meet the team and plan going forward

Attendees: Joshua McCann, Gerard Gargan, Rebecca Gregory, Christopher Gillen

Apologies: Patrick Duggan

Agenda:
- **Meet other team members, introductions.**
- **Agree meeting frequency and format/location.**
- **Agree communication method outside of scheduled meetings.**
- **Schedule the next meeting date/time.**
- **Agree an agenda/plan for the next meeting.**
- **Decide on where documents will be stored (e.g. minutes)**

Discussions (challenges, and progress by each team member on previous tasks):

- Initial meeting – no previous actions or notes to review.
- Meeting frequency agreed – Mondays after lecture for 30-45 mins, Thursday evenings at 6pm for 1hr.
- Some find it difficult to keep up with communicating on teams. Agreement to set up a WhatsApp group. Rebecca took the action to set up the group and add everyone.
- Next meeting agenda/plan agreed.
  - Review the assessment specification together to gain more clarity/consensus on requirements.
  - Record anything that we need further clarification on and send this to the lecturer/connected learning tutor for feedback.
  - Split the project into goals/timelines to ensure we stay on track.
- Agreement to store documents related to the project in the teams environment – Gerard to set up a folder for Meeting Minutes and add today's minutes.
- Notetaker to be nominated at each meeting, documentation will be important.

Action Plan (with allocations to each team member):
- **Rebecca – Set up WhatsApp Group and add the team.**
- **Gerard – Set up folder structure in Teams, add minutes from today's meeting.**

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location: 18/01/2024 @ 6.30pm – Microsoft Teams

Meeting Objectives: Review Coursework Specification / Highlight significant leave

Attendees: Gerard Gargan, Christopher Gillen, Patrick Duggan, Josh McCann, Rebecca Gregory

Apologies: n/a

Agenda:
- **Review previous minutes.**
- **Read coursework specification.**
- **Discuss coursework specification.**
- **Plan next actions.**
- **AOB.**

Previous actions:
Rebecca – Set up WhatsApp group and add the team - Completed.
Gerard – Set up folder structure in Teams, add minutes from meeting - Completed.

Discussions (challenges, and progress by each team member on previous tasks):
CW Spec: CSC7083 Coursework Specification.pdf
Question raised on 'ellipses' on page 4. They are the notation of a use case in UML: (https://sparxsystems.com/resources/tutorials/uml2/use-case-diagram.html#:~:text=A%20use%20case%20is%20a,showing%20the%20direction%20of%20control.)
Significant discussion on system requirements, points 6 and 7 on page 6. What are 'fields' and how they relate to squares.
Similarities to Monopoly was established with colour equivalent to 'fields' and placename equivalent to 'areas'.
Number of squares was raised as potential decision point. Monopoly has 40, based on required number of fields, areas and Monopoly 'Go' square, minimum is 11.
Agreed that front end planning will help structure and focus code development.

Action Plan (with allocations to each team member):
Team: Start Java practice → Due: On-going
Team: Plan a visual or draw virtual gameboard → **Due:** Sunday Evening

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location: 22/01/2024 @ 8pm – Microsoft Teams

Meeting Objectives: Review Visuals / Highlight significant leave

Attendees: Gerard Gargan, Patrick Duggan, Josh McCann, Rebecca Gregory

Apologies: Christopher Gillen

Agenda:
- **Review visuals.**
- **Plan next actions.**
- **AOB.**

Previous actions:
Team - Plan a visual or draw virtual gameboard - Completed

Discussions (challenges, and progress by each team member on previous tasks):
Discussed planned leave – only Josh (March 20th – April 15th) but should have internet access throughout.

Should we read the marking criteria (Individual criteria & Group criteria) to better understand project focus? Agreed yes and to be actioned.

What's next?
- Set up software requirements ready for project
- Agree a layout for visual
- Have everyone come up with a use case each

Planning project meeting schedule in future. Three more lectures planned by Janak then time is scheduled for project.

Agreed to meet on Thur 25th Jan as originally scheduled

Action Plan (with allocations to each team member):
Team: Start Java practice → Due: On-going
Team: Get Jira set up → Due: 01/02/24
Team: Get GitLab set up → Due: 01/02/24
Team: Decide on visual approach you want to use → Due: 01/02/24
Team: Review marking criteria → Due: 01/02/24

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location: 29/1/24 MS Teams

Meeting Objectives: Review last week's minutes and actions; AOB.

Attendees: Joshua McCann, Patrick Duggan, Gerard Gargan, Rebecca Gregory, Christopher Gillen

Apologies: N/A

Agenda:

Review last week's minutes and actions
AOB

Discussions (challenges, and progress by each team member on previous tasks):

GitLab set up: need to await repositories – Jarak said hopefully in next day or two.

Jira team set up: action point continued for this week; invites sent by Joshua (with thanks).

Review of visual approach – similar themes amongst us all but some variations. MVP vs other taxable fields discussed etc. Agreement to focus on MVP and can add additional sections in an iterative approach.

Requirement engineering – discussion Re requirements for process e.g.random number generator – discussion Re fields and dice.Iterative approach – improvements/modifications after completing MVP. Paremeterise code to ensure adaptability and iterative approach.

AOB – re-discussion of any proposed team leave that may affect timelines – nil new updates.
Action Plan (with allocations to each team member):

Team invite to Jira (Joshua) – team to accept invite and complete set up (by next meeting).

Review use case/UML teachings to further understanding – all. To be completed by next meeting, but scope for intermittent team check ins via Whatsapp.

Complete GitHub set up once repositories available – all.

Next meeting – 5th Feb 2024.

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location: 05/02/2024 @ 7.55pm – Microsoft Teams

Meeting Objectives: Jira set up, plan for user requirements

Attendees: Gerard Gargan, Patrick Duggan, Josh McCann, Rebecca Gregory, Christopher Gillen

Apologies: N/A

Agenda:
- Review previous minutes.
- Plan next actions.
- AOB.

Previous actions:
Jira setup – confirmed with screen share.

Discussions (challenges, and progress by each team member on previous tasks):
Team discussed the use of Jira along with project, it will be used to manage work. Gerard shared his screen creating an epic, subsequent issues and pulling them through the board for the next planned action of identifying use cases.

During the lecture, Janak confirmed the project specifications require a 12 square board.
I.e 1 'pass go' square, 1 blank square, 10 field area squares.
Christoper raised possible Win Scenarios. This will likely come from use case development highlighting potential end game scenarios and the team will collectively select one to suit the specification.

Planned an in person meeting to go over git workflow on Sat 17th Feb @ The Graduate School

Planned next action steps to develop use cases, then we can order them by priority, develop out use cases, then build Jira backlog and start sprints.

Next meeting: 12/02/24, time: TBC by WhatsApp.
In person meeting 17/02/24, time: after 11am at earliest.

Action Plan (with allocations to each team member):
Team: Draft use cases → Due: 12/02/24
Gerard: Book meeting room for in person meeting for Sat 17th Feb → Due 15/02/24
Team: Get GitLab set up → Due: 17/02/24 (in-person meeting)

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location:
26/02/2024

Meeting Objectives: Meet with Janak, review progress and ask questions

Attendees:
Gerard Gargan, Joshua McCann, Patrick Duggan, Rebecca Gregory

Apologies: Christopher Gillen

Agenda:

- **Meet with Janak, review progress so far.**
- **Ask Janak any questions/clarifications.**

Discussions (challenges, and progress by each team member on previous tasks):

Janak – How are the team progressing?
Gerard – We have met several times. Last week we met in Person at the Graduate Building. We reviewed how Github works and some of the steps/commands needed for working with Git.

Josh – Showed user requirements on screen share.
Josh asked Janak – At what point do we start putting info into Jira?

Janak – Outlined the steps to take one at a time
1. Document user stories in detail, requiring specific actions and acceptance criteria.
2. Transfer user stories to Jira and create product backlog
3. Design phase
   a. UML Modelling
   b. Sequence Diagrams
   c. Class Diagrams
4. Development phase
   a. Expected to be done in a at least a few increments (sprints)

GG – Question – How much detail does user stories need to have? E.g. does it need to include implementation details such as limiting the number of characters for names, etc.
Janak – No, implementation details are not needed in this case. Something like not allowing duplicate names as per the spec should be included. But detailed implementation details are not required at this point.

Action Plan (with allocations to each team member):

Agreement to meet in person on Saturday 2nd of March at 11am to start working on user stories.
Action – Rebecca to book a room in the graduate building.

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location:
02/03/2024 & TR6 Graduate School QUB

Meeting Objectives: Meet in person to develop User Stories

Attendees:
Gerard Gargan, Joshua McCann, Patrick Duggan, Rebecca Gregory

Apologies: Christopher Gillen

Agenda:
- Review previous minutes.
- Translate requirements into user stories.
- Next actions.

Discussions (challenges, and progress by each team member on previous tasks):
Previous actions – RG booked room.
Previous minutes – reviewed and agreed.

User Requirements reviewed and agreed.
User Stories format agreed and developed as a group.

Next meeting – Mon 4th March to review user stories with Janak
Once reviewed, the user requirements/user stories can be uploaded to Jira backlog.

After web development coursework completed, the high-level modelling design work can be completed which includes completing the following:
- Use cases.
- Sequence diagrams.
- UML / class diagrams.

In person meeting – WhatsApp poll for either 16th, 17th ,18th to go over work to be done.

Action Plan (with allocations to each team member):
No actions, team to focus on web development coursework.

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location:
4/3/2024

Meeting Objectives: Meet with Janak to review progress and ask questions on next steps

Attendees:
Joshua McCann, Gerard Gargan, Rebecca Gregory, Patrick Duggan

Apologies: Christopher Gillen

Agenda:

Review user requirements
Determine next steps

Discussions (challenges, and progress by each team member on previous tasks):

- Reviewed requirements with Janak – good feedback.
- Asked what are the next typical steps to take?
  - The requirements can go into Jira as a backlog
  - User story mapping is next and will help to prioritise the sprint choices (Rank high medium low)
  - Then modelling such as use case diagram / sequence diagram
  - Don't need to be documented in too much detail
  - There should be a logical flow between use cases
- Agreement to meet in person weekend commencing 16th March (date TBC)

Action Plan (with allocations to each team member):
- **Focus on web dev module (Submission due next week)**
- **Meet in person the weekend after and continue through to the next steps outlined above**

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location:
16/03/2024 & The Graduate School

Meeting Objectives: Meet to agree use cases

Attendees:
Joshua McCann, Gerard Gargan, Rebecca Gregory, Patrick Duggan, Christopher Gillen

Apologies:

Agenda:
- **Review previous minutes.**
- **Complete User Story mapping.**
- **Agree system modelling approach.**

Discussions (challenges, and progress by each team member on previous tasks):
Completed user story mapping and prioritised user stories. Discussion around prioritising user stories and whether it should be based on complexity, functionality, or game experience. User stories prioritised in order of importance to functionality.

Reviewed the need to complete user journey mapping which seemed to overlap with user story mapping.

Questions to ask Janak:
- Do use cases align one to one to user story.
- Should we group use cases together.
- User Journey mapping difference from user stories.
- Do we need sequence diagrams per use case.

Some use cases were reviewed which team agreed were good but needed some reformatting to the provided template.

Plan:
- Ask Janak questions on Mon 18th March 6pm
- Draft use cases.
- Draw use case diagrams.
- Draw Domain model and sequence diagrams.
- Draw class model.

Action Plan (with allocations to each team member):
- **Input user stories to Jira = Josh**
- **Arrange meeting with Janak for Mon 18th March for 6pm = Josh**
- **Ask Janak questions = Team**

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location: 18th March 2024 MS Teams

Meeting Objectives:

Attendees: Rebecca Gregory, Joshua McCann, Gerard Gargan, Christopher Gillen

Apologies: Paddy Duggan

Agenda:
Meeting with Janak to review use cases and answer general queries – rescheduled to Wednesday 20th March 6pm
Review of course specification and timelines
Review of unaligned user stories to their appropriate epic/themes

Discussions (challenges, and progress by each team member on previous tasks):
JM away from middle of this week for 3 weeks – discussing mitigations and contact moving forward with time difference
Keen to review use cases before proceeding further to ensure along right lines

Action Plan (with allocations to each team member):
Meet Wednesday 20th March with Janak

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location:
20/03/2024 6pm – Teams meeting

Meeting Objectives: Meet with Janak and review progress + questions

Attendees:
Christopher Gillen, Gerard Gargan, Rebecca Gregory, Patrick Duggan

Apologies: Joshua McCann

Agenda:
- **Review use cases**
- **Review user story mapping**
- **Ask questions**

Discussions (challenges, and progress by each team member on previous tasks):

Review of use cases
JA – The use cases should be in tabular format. Alternative flow should be numbered to correspond with a particular step. Post conditions can be summarised as one big thing but also ok to leave as bullet points.

Question – Do use cases correspond one to one with user stories?
JA – No, you can have less use cases compared to user stories. Use cases can be a combination of multiple user stories.

GG – For example we have two use cases, one for user set up, and one specifically for entering unique names, can these be combined?
JA – Yes exactly.

GG – What is a rough ball park for a reasonable number of use cases?
JA – Around 10 is a good number, you don't want a high number due to the extra complexity this will add
JA – For example here are some high level functional aspects of the system that would be typical use cases:
- Registration
- Taking a turn
- Rolling dice
- Making a move
- Making a development
- Obligations / purchases… etc

Question about use case diagrams

Question – Should there be one use case diagram for each use case?
JA – No, there should be only one use case diagram for the entire system. Each use case will be an oval on the diagram.

Question – What software is best for doing these diagrams?
JA – Draw.io

Sequence diagrams
Question – Do we have one sequence diagram for the whole system?
JA – In this case no, you will have a sequence diagram for each use case. This should map only the main high level interactions between objects, it does not need to have the finer details. This is why it is also good to keep the number of use cases lower.

Evidence of requirements engineering
Question – How do we evidence requirements engineering

JA – This will be through a combination of me having access to Jira, and you will also summarise in the report. For example you should add all use cases to the backlog in Jira, and organise into sprints. Then In the report there should be a summary for each sprint. This will help to show the iterative approach as per the marking scheme.

Action Plan (with allocations to each team member):
Action – Meeting planned for Saturday at 11am – Rebecca to book the room in the grad building
Action – Rebecca will try to combine some user stories to reduce the number from 20+ for review on Saturday.

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location:
Grad Building – 23/03/24 11am

Meeting Objectives:

Attendees:

Christopher Gillen, Gerard Gargan, Rebecca Gregory, Patrick Duggan

Apologies: Joshua McCann

Agenda:

- **Review Rebecca's Use Case amalgamation**
- **Update User Story Map**
- **Discuss Use Case Diagram**
- **Discuss Sequence Diagram**

Discussions (challenges, and progress by each team member on previous tasks):

- Reviewed Janaks video on Use Case Diagrams
- Agreed on Rebecca's propose Use Cases
- Created a Use Case Diagram on Draw.io
- Sent Use Case Diagram to Janak on Teams for advice
- Started drawing the UML Class Digaram

- Sprint discussion and timing
- Planning to have 2 people per sprint to stop branches getting too confusing
- Want to get sprints started this week

Action Plan (with allocations to each team member):

- Ask Janak about testing from Agile Iterations part of spec – "the development and testing of the product increments". What sort of testing is expected?
- Await Janaks response on Use Case Diagram
- How do we evidence how much work each of us have done on GitLab?
- Gerard to have a look through spec to see what all needs added to the Class Diagram
- Ask Josh what sprint would suit him best then divide up 3 sprints between us 5 (sprints will be high, medium and low priority User Stories)

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location: 26th March via microsoft teams

Attendees:
Dr Janik Adikhari,
Christopher Gillen, Rebecca Gregory, Patrick Duggan, Gerard Dargan

Meeting Objectives: To clarify questions raised from team meeting on Sat 23rd March; see agenda for specifics

Agenda:

1. Get feedback on use-case diagram.
2. Get feedback on class diagram.
3. Get clarity on the scope of system testing.
4. Get clarity on requirements for sequence diagrams.
5. Clarity on how to evidence work for marking purposes.

Discussions (challenges, and progress by each team member on previous tasks):

Feedback received on use-case diagram; removed unnecessary items, realigned use-cases and corrected the associations and relationships between actor and cases.

Janik wanted us to finish the class diagram before giving definitive feedback e.g. add in the class relationship arrows to show inheritance relationships etc.

Unit testing will be relatively minor; it is sufficient to complete testing of one case or aspect of functionality to show that we understand unit-testing for the project rather than showing detailed unit testing of every class/object.

User acceptance testing should be focussed on and should be exhaustive.

Sequence diagram- one per use-case and include the objects it involves and their key interactions. Does not need to be super-detailed- just what is main purpose of use-case and its functionality i.e.

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location:
30/3/24 QUB Graduate Building

Meeting Objectives: Review/finalisation of UML/Use Cases/Use Case Diagrams

Attendees:

Gerard Gargan, Rebecca Gregory, Patrick Duggan, Christopher Gillen

Apologies: Joshua McCann

Agenda:

Completion of UML diagram
Review of use cases
Completion of use case diagram
Start of Sprint1

Discussions (challenges, and progress by each team member on previous tasks):

UML completed and agreed upon by all team members in attendance
Use cases reviewed – happy with terminology
RG will update bullets -> numbers and tabulate for report
Use case diagram updated as per JA feedback – team happy with outcome
Review of sequence diagrams – lectures/videos. How best to document these discussed
Sprint 1 commenced

Action Plan (with allocations to each team member):
Meet again Monday 1st April for mid sprint update
Room booked by RG
All team members to retest Jira and Gitlab accounts

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location:
1/4/24 QUB Graduate Building

Meeting Objectives: Review/finalisation of UML/Use Cases/Use Case Diagrams

Attendees:

Gerard Gargan, Rebecca Gregory, Patrick Duggan, Christopher Gillen

Apologies: Joshua McCann

Agenda:

Sprint 1 mid point check in

Discussions (challenges, and progress by each team member on previous tasks):

Challenges with GitLab/Eclipse connectivity – troubleshooting took place. All team members now able to connect as required for sprints
Discussion Re progress of sprint and allocations
Timelines for Sprint 2 and Sprint 3 discussed
Plan to link in with JM and plan further Teams meeting this week for when he can attend given time difference
Next FTF meeting planned for Sat 6th April

Action Plan (with allocations to each team member):
Sprint 1 to be completed by GG/PD/CG Thursday 4th April
RG to complete tabulation of use cases and review report requirements
Meet via Teams Thursday 4th April to review end of Sprint 1/start and allocate Sprint 2
Next FTF meeting planned for Sat 6th April
Team in agreement for ongoing discussions/troubleshooting via Whatsapp if required before next meeting

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location: 04/04/2024 Microsoft Teams

Meeting Objectives: Review Sprint 1, Start sprint 2

Attendees:
Gerard Gargan, Rebecca Gregory, Joshua McCann, Christopher Gillen, Patrick Duggan

Apologies: N/A

Agenda:

- **Review Sprint 1 & complete on Jira**
- **Assign user stories on Jira for sprint 2**
- **Start sprint 2**
- **Sequence diagrams**

Discussions (challenges, and progress by each team member on previous tasks):

Gerard, Chris & Paddy have completed Sprint 1 tickets ahead of time.
Sprint 1 reviewed and completed in Jira.
Sprint 2 backlog reviewed and tickets assigned to Gerard, Paddy and Rebecca.
Sprint 2 started – due for completion Sunday.

Challenge has been the sequence diagrams. Agreement that Josh and Chris will attempt to do one each for review at the next meeting with Janak on the 8th of April.

Becca to change use cases to tabular format and upload to teams.

Other tasks that need completed - To be reviewed on Saturday 6/4/24 and planned

- **Domain Model**
- **Group report**
- **Group video**
- **Individual reports**

Action Plan (with allocations to each team member):
- **Becca to change use cases to tabular format and upload to teams.**
- **Gerard, Paddy and Becca to start sprint 2 with a view to completing by Sunday 7th of April.**
- **Josh to review notes and lecture on sequence diagrams and complete one for review with Janak**
- **Chris to do the same**

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location: 06/04/24 – 11am – Graduate Building

Meeting Objectives:

Attendees:
Patrick Duggan, Gerard Gargan, Rebecca Gregory, Christopher Gillen

Apologies: Joshua McCann

Agenda:

- Review Sprint 1 and discuss any problems
- Finish Sprint 2
- Discuss Sequence Diagrams for our Use Cases

Discussions (challenges, and progress by each team member on previous tasks):

- Watched Youtube video on creating sequence diagrams – link shared with team
- Watched Youtube video on UAT
- Created sequence diagram for UC01
- Attempted to create sequence diagram for UC02, decided to speak with Janak first to make sure we are on the right track
- Rebecca completed UC07 and US13 to complete Sprint 2

Action Plan (with allocations to each team member):

- Complete Sprint 3 by Monday
- Show Janak UML
- Ask about Domain Diagram – just an empty UML diagram?
- Show Janak Sequence Diagram for UC01
- Ask Janak:
  - What level of detail do we need for the sequence diagrams
  - Should we include a Player class for its connection with the Game class
  - Can we just pick a scenario for each use case as opposed to mapping out every possibility
  - Can we merge several use cases into one sequence diagram
- Ask about UAT and Unit Testing – how exhaustive
- Should UAT be in the video?
- For group video – should this be one person doing it or should everyone be involved?

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location: 10/04/24 – MS Teams
Meeting Objectives: Agree Sequence Diagrams and User Acceptance Testing with Janak
Attendees: Christopher Gillen (CG), Gerard Gargan (GG), Joshua McCann, Janak
Apologies: Patrick Duggan, Rebeccas Gregory

Agenda:
- Review previous minutes
- Meet with Janak

Discussions (challenges, and progress by each team member on previous tasks):
Talk before Janak
- Question planning
- UML question
  - Looks logical
  - Missing – name of the associations or relationships
    - Eg - Is a / has a
  - Missing – direction of relationship? – abstract
    - Add arrows to show relationship direction
  - What about domain diagram
    - Don't do the domain diagram
- Sequence diagrams
  - Gerards example – name
  - For the loop as well, need a frame
  - GG: Is it OK to mix and match method calls and other stuff? – JA: its mostly methods
  - GG: What about methods within methods? – JA: Methods are most important! And should be the focus. So the main method is the key but the helper methods can be abstracted out
  - Group: Go with first one with more details, although the activation period with the method call.
  - Group: We will include frames for alt and loops, but diff activation boxes
  - GG: What about use cases with multiple different paths? – JA: Include all scenarios
  - GG: If overlapping with other use cases? – JA: You can but not advisable
  - JM: So if the use case has preconditions? JA: You can refer to previous sequence diagrams by referring to it at the start then building upon it?
  - GG: Do we need to instantiate player class in UC1? JA: No it doesn't refer to player class within this use case
- Junit and User Acceptance testing
  - CG: Can we focus on specific testing? – JA: Yes
  - CG: UAT – more involved? Is it more functional testing? – JA: Functional testing, no performance. It's all functionality.
  - CG: How do we showcase that? JA: Perhaps its more guided by the user stories.
  - CG: User Interface testing is minimal? – Yeah
  - CG: Functional testing? JA: Guided by gameplay flow from beginning to end. – More high-level black box testing. The game plays correctly.

Janak additional comments:
- Don't write Junit for each class – only one or two and submit
- UAT is more guided by user stories
- GG: Group demo video, does everyone need to be involved? – JA: Everybody needs to be involved in it.
- GG: Is it better on Teams or in person? JA: Either, and 7 mins is slightly flexible – upto 8.5mins
- GG: What is the content, is it game functionality? JA: Yeah, there is a summary and implementation etc, its mostly game flow
- GG: Do you want nitty gritty coding? JA: Major decisions, technical challenges etc

Action Plan (with allocations to each team member):
- Sequence diagram updates – Gerard UC01 onwards, Josh UC10 backwards for Sat 13th
- Junit for UC01 – Chris to complete for Sat 13th.
- Start UAT on Sat 13th
- Video planning on Sat 13th– plan out what needs covered and to what extent.

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location:
13/4/24 Graduate Buildina

Meeting Objectives: Review sequence diagrams, review user acceptance tests, review junit

Attendees: Gerard Gargan, Rebecca Gregory, Christopher Gillen, Patrick Duggan

Apologies: Joshua McCann

Agenda:
- **Review sequence diagrams**
- **Review user acceptance tests**
- **Review junit testing**

Discussions (challenges, and progress by each team member on previous tasks):
RG – User acceptance tests for user stories 1-10 completed - reviewed during the meeting.
PD – User acceptance tests for user stories 11 – 16 completed
CG – User acceptance tests for user stories 17 – 20 completed, CG to complete test for US 21 prior to next meeting

Junit testing reviewed and approved, no changes needed.
CG – Found one bug during testing, Player was exited but the full fine is still paid to the field owner and players balance was not reduced to 0.

Sequence diagrams reviewed – Were very good. Some feedback/things for Josh to check in his below:

Action Plan (with allocations to each team member):
- GG to fix bug in the code for player fine/exiting as noted above
- Chris to finish UAT for US 21
- Rebecca to standardise all UAT's into one formatted document
- Becca, Patrick and Chris to take evidence screenshots for each test to reference in the report
- Josh to review feedback for sequence diagrams

Meeting Minutes

CSC7083 Course Work (Group Component)

Date and Location:
17/4/24 Microsoft Teams 16:00

Meeting Objectives: Meeting with Janak to go over report questions.

Attendees: Gerard Gargan, Rebecca Gregory, Christopher Gillen, Joshua McCann, Janak Adhikari

Apologies: Patrick Duggan

Agenda:

1. Go over questions with Janak.
2. Review report and video plans.

Janak Questions:
GG - What is allowed in appendix vs report?
JA – Technically no limit to appendix, JA hopes that it's not too much, but it will be marked.

JM - Can just reference to Jira and how to access the backlog instead of having 20 pages of exported data in appendix?
JA – Yes, as I have access to it, that is OK.

CG – For the UAT and Junit, is it OK to put the screenshot evidence in the appendix?
JA - Yeah that is OK to include them.
CG – The Testing approach was basic Junit, exhaustive UAT as agreed previously.
JA – That sounds like more than enough. Although for Junit, it would need to be done for a sample CG - For the sample Junit, is testing for for one constructor and one method suitable?
JA – That is sufficient.

GG – The criteria mentions 20% marks for working system? Does this need to be a separate section of the report or does the video cover this requirement?
JA – no separate documentation, it is the entire report and code etc. It's actually worth 10%, with additional 10% for other things such as UX and how game runs etc. No specific section is required for working system.

GG – Can we make sure it's the correct Jira chart to use for the report? Issue count per sprint was shown on screenshare to JA.
JA – That's the sprint burndown, I want the whole project one.
JM – I don't think Jira allows for whole project burndown report.
JA - He will confirm if possible and contact Gerard. If not sprint burndown will do.

GG - How to submit the video? Link for YouTube or different?
JA – Upto to you guys, YouTube works well for him. Length – target 7mins, slightly over is OK, 10min is hard cap.

PD Questions?
What to show for agile iterations?
JA: Selection of backlog per sprint and why. Decisions taken during sprint. What was the product increment at the end of it. Sprint retrospective and what was used to feed into next sprint.
Sprint planning is:

- How did we prioritise the backlog? Basis for prioritisation of product backlog. What needs developed first etc. What is exactly the basis for the selection of priority. We can link back to user story map.

Sprint Retrospective is:

- What did you learn and what will you try to improve next sprint.

GitLab commit history, is this total or per sprint?
JA: Don't need separated out per sprint, the git log should be sufficient. Again, like Jira, we can reference GitLab which he has access to reduce report appendix.

GG - Within system design it mentions User Story mapping, Screenshare of the user journey map template was shown to JA and asked if it was required to fulfil this requirement for the report?
JA: Do not do the user journey mapping, just the user story mapping.

Actions:

- Report draft sections for Friday 5pm for Rebecca to review and format on Sat (file in teams under 'Written Report')
- Draft video for Friday for Gerard to clip together for Sat (Save video clips in 'Group Video' folder) ****Try to keep under 50s for overall video length****