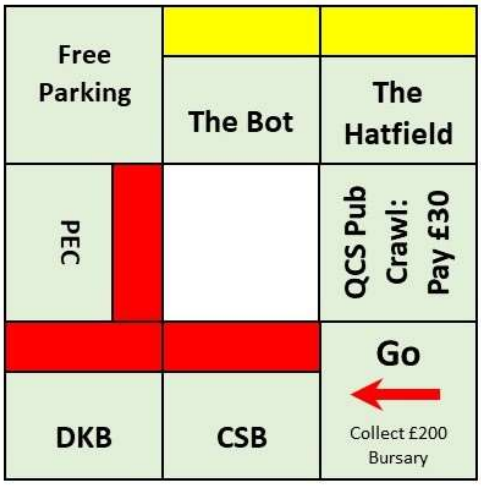


Monoversity: An Undergraduate's Experience at Queen's University, Belfast

Project Design and Implementation Summary Report

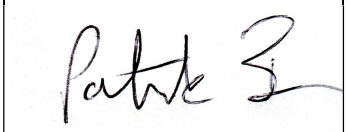
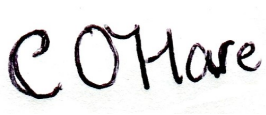
Prepared by Queen's University, Belfast
Group G19: Compiler
March 2018

For Module Ref.: CSC7053 Software Engineering

Document Control

Document:	Project Design and Implementation Summary Report
Project:	Monoversity: An Undergraduate's Experience at QUB Monopoly Game
Module:	CSC7053 Software Engineering

Revision:	For Issue
Date:	08 March 2018

Report prepared, reviewed and approved by: *			
Signature:	**Name and Student No.:**	**Signature:**	**Name and Student No.:**
	Paddy Burns (PB) 18596037		Colm McGoldrick (CmcG) 40006755
	Conor Leonard (CL) 40106357		Justin Kelly (JK) 40214842
	Catherine O'Hare (COH) 40108703		

* Initials next to each section/subsection indicate team member responsible for delivery and/or contributor to output

Contents

Appendices

Appendix A	Project Delivery Programme
Appendix B	Property Values and Rent Costs
Appendix C	Meeting Minutes
Appendix D	Static Testing Outputs

1.0 Introduction

PB

1.1 Project Overview

The project team has been commissioned to develop a monopoly-based game with a theme that is connected to Queen's University, Belfast (QUB). This document provides an overview of the processes implemented in the production of the required monopoly-based game, and the outputs from such during the development process.

1.2 Scope of Work

The overarching scope of work has been derived from the Project Specification which outlined a number of key phases of the development process that are required to be followed, and subsequently demonstrated in the Summary Report. This specifically involves the following:

- Requirements Analysis: incorporating a game guide, and subsequent Use Case Diagram and associated Use Case Descriptions based on Unified Modelling Language (UML) principles;
- System Realisation: conceptualised using UML Sequence Diagrams;
- System Design: idealised through a UML Class Diagram;
- Design Process: providing details on the software testing process implemented, and a record of team meetings;
- Game Development: essentially involves the delivery of a working system that meets the specified requirements.

The aforementioned is subsequently detailed in the following sections of this report.

1.3 Project Delivery Programme

A programme for the delivery of the project was developed following the initial kick-off meeting and conceptualised using MS Project. A copy of the programme is presented in Appendix A, which details the key tasks and workflows deemed necessary to meet the project obligations in full. Further details regarding the project delivery process are provided in Section 5.

2.0 Requirements Analysis

PB

In undertaking an analysis of the core project requirements, the following key sequences were carried out:

- Initial project kick-off meeting to conduct a high-level discussion on the Monoversity Project Specification, agree on a game theme, and initial project progression;
- Analysis of the customer requirements as detailed in the Monoversity Project Specification to determine core system functionality requirements;
- Development of a "Game Guide" that included a conceptualised board layout and game rules;
- Production of a Use Case Diagram and associated Use Case Descriptions.

2.1 Project Kick-Off

The key outcome from the project kick-off meeting with respect to the Requirements Analysis process was the agreement among the team on a theme for the game. The agreed approach was to develop a light-hearted game with the core theme of a student's experience at attending QUB, including the social aspects of the experience. The working title "Monoversity: An Undergraduate's Experience at QUB.

2.2 Determination of Core System Functionality Requirements

The Project Specification was analysed in order to identify the core requirements in terms of the overall function of the system that would subsequently inform the system development process. Outlined in Table 2.1, these core requirements would also form the basis against which the operational system would be benchmarked against during the testing phase(s) of the development process.

Table 2.1: Summary of Core System Functionality Requirements

Ref.	Requirement
R1	The game has two players, who start with the same amount of money. Their names should be entered.
R2	The players take turns.
R3	They throw a virtual die.
R4	There is a start square, which players land on or pass to pick up their salary/grant money/funding.
R5	There are two property groups, one comprising two properties and another comprising three.
R6	The properties in the two-property group are more expensive than the ones in the three-property group.
R7	Before you can develop a group by building on it, you must own the *whole* group,

Ref.	Requirement
R8	The game will have a university- and specifically a Queen's-related theme.
R9	Three 'houses'/*labs* are needed before you can build or establish a 'hotel' (or whatever you decide it is).
R10	Players taking a turn are told where they have landed and what their obligations or opportunities are.
R11	Where appropriate, they may indicate their choice of action.
R12	If a player's finances have changed, the system indicates the reason for the change and announces the player's new bank balance.
R13	When one player goes bankrupt, the other is declared the winner. If one player no longer wants to play, the game ends. In both cases, the players' bank balance is given.

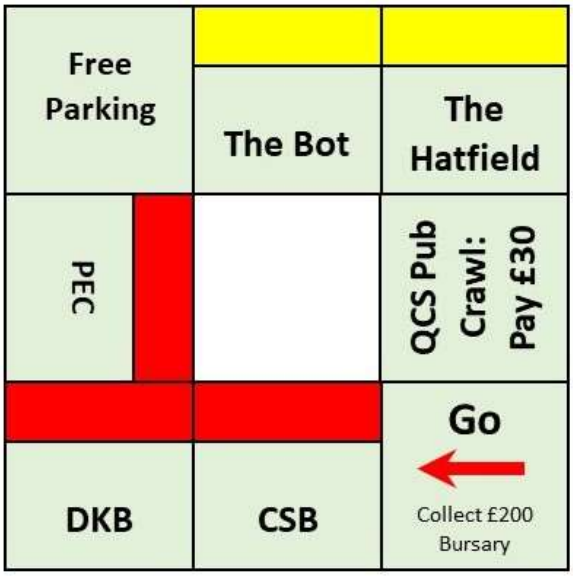
2.3 Game Guide

PB, JK, COH, CmcG, CL

2.3.1 Board Layout

The development of a board layout and associated game guide/rules was carried out in context of the key project requirements outlined in Table 2. 1, with the board layout presented in Inset 2.1 (below):

Inset 2.1: Conceptualised Game Board Layout



The project specification required the provision of a Go Square and two distinct property groups (one comprising two property squares, the other three property squares). The QUB theme is enforced when complying with this requirement through the inclusion of QUB property squares (Red squares), namely the Computer Science Building (CSB), David Keir Building (DKB) and the Physical Education Centre

(PEC). The more expensive yellow property group represents bars, the social aspect of attending university.

A 'Free Parking' and 'Queens Computing Society (QCS) Pub Crawl' square (replacing the traditional 'Income Tax' square in Monopoly) were included, under the terms of the project specification (i.e. "*Your game will have many of the features that players of the original Monopoly would expect, but in a much simpler form")* to give a more rounded game experience.

2.3.2 Basic Rules

CmcG

- The aim of the game of the game is to become the wealthiest player through buying, renting and selling property.
- The game is for two players, with each player prompted to enter their name before the start of the game.
- Each player starts off with £500 in the bank.
- Before the game starts both players roll the virtual dice. The player who rolls the highest number gets to move first and the game is started.
- The players then take turns to roll the virtual dice. One dice is rolled, and the moves of the players are made by the system.
- The game is over either when one player is bankrupt (can't pay rent or a fine) or when one player decides to terminate the game when they are asked if they wish to roll the dice when it's their turn.

2.3.3 Gameplay

CmcG

Landing on Squares (General)

- When the player lands on a square the system will display the name of that square and any functions/details associated with it.
- Both players start the game on the 'Go' square.
- Each time a player lands on or passes 'Go' following the start they receive £200.00.
- When a player lands on the 'pub crawl' square they have to pay £30.00 to the bank.
- When a player lands on the Free Parking square no changes are made to the players balance or status.

Additional functions associated with Property Squares

- When a player lands on a property square the system will confirm whether the property is owned or not.
- If a property square is not already owned, the player will have the option to buy that property square if they have the required amount of funds.
- If the property square is already owned, the system will display whether there is any development on the site (i.e. extra floors or extensions), the amount of rent due, the player's current balance and the balance after the rent has been paid.

- If a player has insufficient funds, the system will declare that player bankrupt.
- Development is only allowed on properties when the player owns all the properties in that group.
- Extensions can only be added when three floors have been added to that property square.
- Costs of properties and rent values can be found in Appendix B.

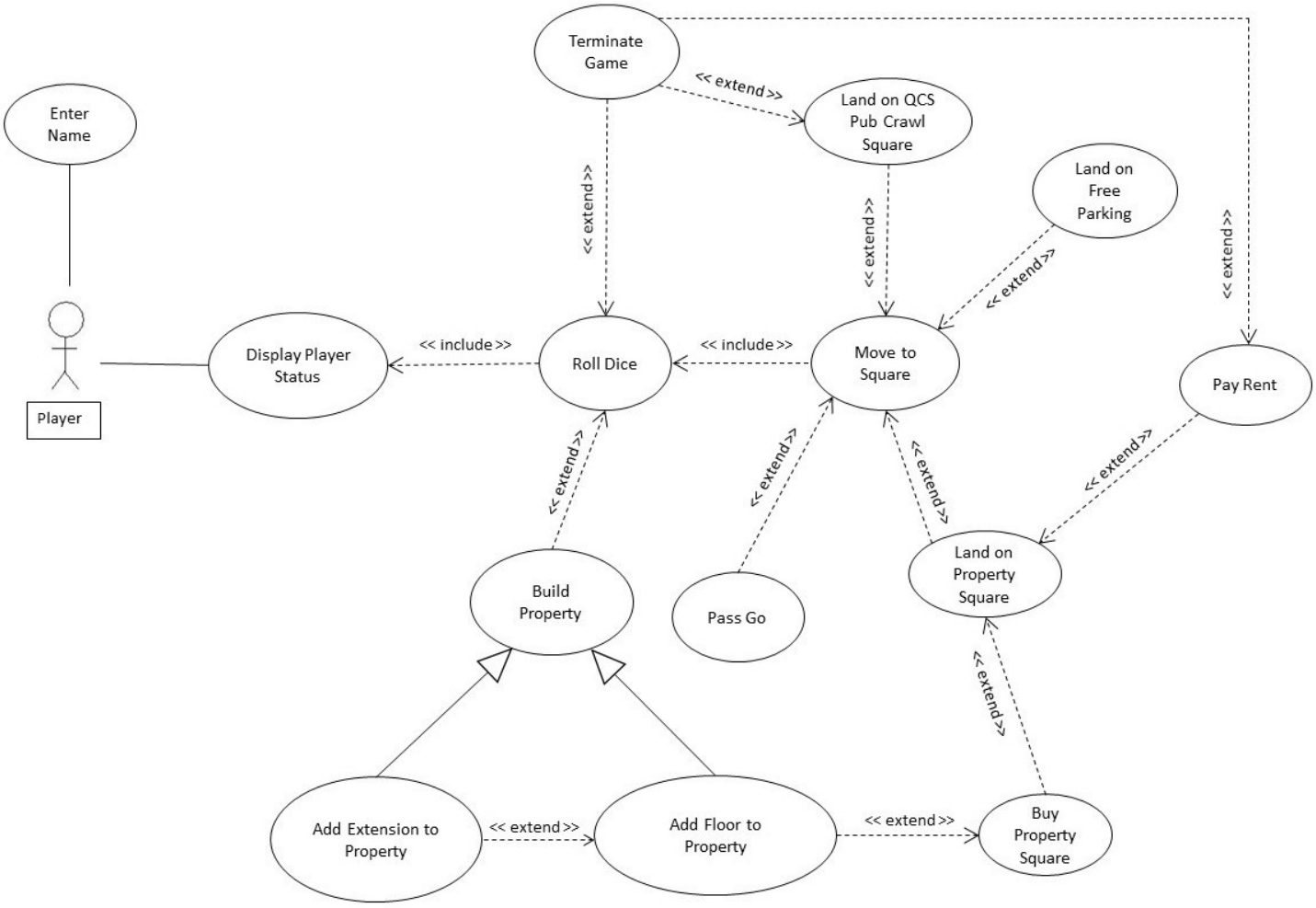
2.4 Use Case Diagram and Descriptions

PB, JK, COH, CmcG, CL (whole of Section 2.4)

2.4.1 Use Case Diagram

The Use Case Diagram that outlines the system pathways which reflect the core requirements is outlined in Inset 2.2 overleaf. The associated Use Case Descriptions which detail the specific actions intended to be implemented for each use case are presented in Section 2.4.2.

Inset 2.2: Use Case Diagram



2.4.2 Use Case Descriptions

Flow of Events for Enter Name use case	
Objective	**All players to enter valid player details**
Precondition	The system/program has been initiated (ran) by a player.
Main Flow	1. A dialog box is displayed, and the players are prompted to enter their name. 2. The system requires the player to confirm if the name they entered is correct. 3. If the player indicates it was incorrect, the system requests the player to enter their name again.
Alternative Flows	None
Post-Condition	The game proceeds, with the players offered the option to roll the dice.

Flow of events for the Display Player Status to Square use-case	
Objective	**To output the current status of the player**
Precondition	It is the player's turn.
Main Flow	1. The system displays the player's current bank balance. 2. The system lists any properties the player currently owns.
Alternative Flows	None
Post-condition	The system proceeds to the Roll Dice use case.

Flow of events for the Terminate Game use-case	
Objective	**Electing to terminate the game**
Precondition	The player has started the game.
Main Flow	1. The player has decided to end the game at the beginning of their turn by selecting 'N' when asked if they wish to roll the dice. 2. The system displays a message asking if the player is sure they would not like to continue playing. 3. Player input required: Y for yes, they are sure, or N for no, they are not sure. 4. On selecting Y, the game terminates, with the other player declared the winner. 5. On selecting N, the game resumes.
Alternative Flows	1. A player is declared bankrupt where they do not have sufficient funds to either pay rent or pay the QCS Pub Crawl fee. 2. The game is terminated, with the other player declared the winner.
Post-condition	The Game is over.

Flow of events for the Roll Dice use-case	
Objective	**To roll the dice**
Precondition	It is the player's turn
Main Flow	1. The system displays the player's current status (Display Player Status use case). 2. The system asks the player if they would like to roll the dice. 3. Player input required: Y for Yes, N for No. 4. Player indicates they would like to roll by entering Y.
Alternative Flows	At 2, if the player indicates that they would not like to roll the dice, proceed to the Terminate Game use case description.
Post-condition	The random number generated (dice roll) will be displayed.

Flow of events for the Move to Square use-case	
Objective	**To move around the board**
Precondition	The player has rolled the dice
Main Flow	3. The system calculates the new board position of the player based on their previous location and the dice number rolled. 4. The system displays the new board location and any pertinent details about the square landed on.
Alternative Flows	None
Post-condition	The player has changed position and moved to a new square.

Flow of Events for Land on Free Parking use case	
Objective	**Land on the Free Parking square**
Precondition	After rolling the dice, the system calculates the player's location as the Free Parking square.
Main Flow	1. The system display informs the player they have landed on the Free Parking square.
Alternative Flows	None
Post-Condition	The player's turn is over.

Flow of Events for the Land on the QCS Pub Crawl Square use case	
Objective	**Land on a the QCS Pub Crawl Square**
Precondition	After rolling the dice, the system calculates the player's location as the QCS Pub Crawl square.
Main Flow	1. A prompt from the system indicates a QCS Pub Crawl fee of £30 is due. 2. The required fee of £30 is automatically decremented from the player's bank balance. 3. The new bank balance is displayed by the system.
Alternative Flows	At point 2, if the player does not have sufficient funds to cover the QCS Pub Crawl fee, proceed to the Terminate Game use case description (Alternative Flow).
Post-Condition	1. The player's balance has been decremented. 2. The player's turn is over.

Flow of Events for the Land on Property Square use case	
Objective	**Land on a Property Square**
Precondition	After rolling the dice, the system calculates the player's location as one of the Property squares (i.e. the PEC, DKB, CSB, Bot or Hatfield).
Main Flow	1. The system confirms the player's position on the board and any details regarding the property square. 2. If the square is not owned by any player, proceed to the Buy Property use case. 3. If the square is owned by the other player, proceed to the Pay Rent use case. 4. If the player owns the square and the remaining property squares in that Property Group, proceed to the Add Floor to a Property use case. 5. If the player owns the square, but not the remaining property squares in that Property Group, the system will display a message confirming the player currently owns that property square.
Alternative Flows	None.
Post-Condition	The player's turn is over.

Flow of Events for the Pass Go use case	
Objective	**Player to Pass Go**
Precondition	After rolling the dice, the system calculates the player's location at a square which takes the player past the Go square.
Main Flow	1. The system increments the player's bank balance by £200.00. 2. The system displays a message informing the player they have passed Go and have had £200 added to their bank balance, with the updated balance also displayed.
Alternative Flows	The player lands on Go. Flow of events then follow those of the Main Flow.
Post-Condition	The system initiates the flow of events for the square the player ultimately landed on.

Flow of Events for the Buy Property use case	
Objective	**To buy a property square**
Precondition	1. The player lands on a property square which is not owned by any player. 2. The Player has the necessary funds to cover the cost of the property square.
Main Flow	1. The system display confirms the square the player has landed on and that it is not owned by any player. 2. The system displays the property square details (cost, base rent) and asks if the player wishes to buy the property. 3. Player input required: Y for Yes, N for No. 4. The Player indicates they wish to buy the property by entering Y. 5. The system indicates that the player has purchased the property. 6. The updated bank balance is displayed.
Alternative Flows	1. At 2, if the player does not have sufficient funds to buy the property, the system confirms this (message on screen) and the player's turn is over. 2. At 3, if the player selects N, the player's turn is over.
Post-Condition	1. The player owns the property. 2. The player's turn is over.

Flow of Events for the Pay Rent use case	
Objective	**To pay the required rent**
Precondition	The player has landed on a property square owned by the other player.
Main Flow	1. The system display confirms the square the player has landed on and that it is owned by the other player therefore rent is due. 2. The system displays the amount of rent to be paid and decrements the players account with this value if sufficient funds are available. 3. The system displays the player's updated bank balance. 4. The system displays the updated bank balance of the player who owns the property to confirm their account has been incremented with the rent value.
Alternative Flows	At point 2, if the player does not have sufficient funds to pay the rental amount, proceed to the Terminate Game use case.
Post-Condition	1. The player has paid rent. 2. The player's turn is over.

Flow of Events for the Build Property use case	
Objective	**To add a floor or extension to a property square**
Precondition	1. The player has landed on a property square that they own. 2. The player owns all the properties in that particular Property Group.
Main Flow	1. If there are fewer than three floors on the property, proceed to the Add Floor to Property use case. 2. If there are more than three floors on the property, proceed to the Add Extension to Property use case.
Alternative Flows	None
Post-Condition	1. The property has had an extension added and increased in rental value. 2. The player's turn is over.

Flow of Events for the Add Floor to Property use case	
Objective	**To add a floor to a property square**
Precondition	1. The player has landed on a property square that they own. 2. The player owns all the properties in that particular Property Group. 3. There are fewer than three floors on the property square.
Main Flow	1. The system will ask if the Player wishes to add a floor to the property square and the cost of doing so. 2. Player input required: Y for Yes, N for No. 3. The system will display a message confirming the floor has been added, how many floors are now on the property, and the updated rental value. 4. The system will display the player's updated bank balance.
Alternative Flows	None
Post-Condition	1. The property has had a floor added and increased in rental value. 2. The player's turn is over.

Flow of Events for the Adding an Extension to Property use case	
Objective	**To add an extension to a property square**
Precondition	1. The player has landed on a property square that they own. 2. The player owns all the properties in that particular Property Group. 3. There are three floors on the property square.
Main Flow	1. The system will ask if the Player wishes to add an extension to the property square and the cost of doing so. 2. Player input required: Y for Yes, N for No. 3. The system will display a message confirming the extension has been added, how many extensions are now on the property, and the updated rental value. 4. The system will display the player's updated bank balance.
Alternative Flows	None
Post-Condition	1. The property has had an extension added and increased in rental value. 2. The player's turn is over.

3.0 System Realisation

PB

A Sequence Diagram is used to model the temporal interactions between objects identified from the Use Case process (i.e. during runtime). In order to facilitate the identification of classes and objects that would be included in the final, working system that meets the requirements of the specification, sequence diagrams were developed for key processes that are felt to represent the full sequence of events which would be experienced during gameplay, specifically the following sequence diagrams:

- Addition of player names and termination of game;
- Moving to a square on the board (except a property square), including passing Go (Inset 3.1);
- Moving to a property square, including purchase of a property, adding a floor and adding an extension (Inset 3.2);
- Landing on a property square and either paying rent or unable to do so (Inset 3.3).

The sequence diagrams are presented after the following descriptions of such.

3.1 Discussion of Sequence Diagrams Developed

Addition of Player Names and Termination of Game

Once a player initiates the game (i.e. runs the program) the system will initially create two player objects, and then request the players enter valid names. The first player enters a name and, if it meets the validation criteria in terms of character length and content, the system asks the player if this is correct. If the player selects No, the system will loop and ask the player again to enter details. If the player confirms (enters Y for yes) then the system requests the second player to enter their name, again with the same process. The system then requests each player to roll the virtual dice, with the first turn set to the player rolling the highest number, who is then given the option to roll the dice again (Yes or No to roll dice).

Specific pathways of actions are provided based on the player's input. If 'Yes' is entered, the system rolls the virtual dice, then proceeds to a sequence of events relative to the number rolled (and subsequent board position), as detailed on the remaining sequence diagrams. If 'No' is entered, the system will revert to the game termination sequence, declaring the other player the winner and displaying the bank balance for both players.

Moving to a Square (except a Property Square)

The initial aspects of the sequence diagram map the planned process which allows the player to roll a virtual dice; use of the number rolled (passed from the RollDice method) by the system to calculate the

player's new board position based on their current location (where each square on the board has a corresponding numerical reference number); then carrying out a specific sequence of actions or display outputs based on the new board location (i.e. ref. number).

This diagram focussed on the sequence of actions when the player lands on the Go, QCS Pub Crawl and Free Parking squares, and the actions to implement if they pass Go on the way to these new board locations. Functions associated with landing on a property square are considered in separate sequence diagrams due to the complexities associated with such.

With respect to the Go square functionality, alternative actions were anticipated where the calculated new board location (the dice number + current board location) would be greater than 8 (the number of squares on the board) to represent passing Go; and where the new board location would be 1 [current location – (8 – the dice number) = 1] to represent landing on Go. In both instances the current player's balance should be incremented by £200 as per the game guide rules.

Similarly, when the player lands on the QCS Pub Crawl square (calculated board location = 8), their account should be automatically decremented by the required amount (£30).

When landing on the Free Parking Square (calculated board location = 5) no specific player actions are required

With all of the above situations, an instance of the respective Square class will be interacted with to output the details of that specific square to screen, and the actions carried out by the system as a consequence.

Moving to a Property Square (including purchase of a property, adding a floor and adding an extension)

As with the sequence diagram for the 'Moving to a Square' diagram, the initial aspects of the process capture the sequence of events that take place in order to calculate the player's new board location based on a random number returned from the RollDice method/class. One minor deviation will ensure the details of the property (price, rental value etc.) will be displayed. Alternative sequences associated with purchasing a property square, adding floors and extensions to the square have been mapped in the diagram, all which occur after a check on the ownership status of the square whether the player has the necessary funds to buy the square (by an integrated method) has been carried out.

Where the square is not owned by any player and the current player has the necessary funds to buy the square (i.e. bank balance > property price) two optional sequences are provided to capture the events that occur when the player is asked whether they wish to buy the property. On selecting No, a message confirming this will be displayed. On selecting Yes, a sequence of actions to firstly get the

property cost, decrement the player's account with the cost of the property and set the ownership status of the property (including displaying confirmation of such) will be carried out.

Where the player lands on a square which they own (as determined by the system), has fewer than 3 floors developed on it, and the cost of a new floor is less than their bank balance, a sequence of events similar to the previous (purchasing a property square) will be carried out, again based on how the player responds when asked if they wish to add a floor to the property (i.e. Yes or No). The 'No' response will illicit the same sequence of actions, and similarly a 'Yes' response will initially get the cost of a floor, require the respective adjustment of the player's bank account and setting the new number of floors. A similar process will also be followed for the sequence of events surrounding the addition of a new extension to the property (controlling conditions of the player owns the property, number of floors > 3, and the cost of a floor is less than the player's bank balance).

On exiting any of these sequences, an action has been included to ensure the player's turn is set to the other player.

Landing on a Property Square and Paying Rent

The mapped initial sequence of actions for when a player lands on a property square and the payment of rent is considered mirrors that outlined for the previous sequence diagram (moving to a property square and considering purchase of such) in that the system needs to determine the current state of ownership for that square. Once ownership is identified, a number of alternative sequences are provided for where the current player does not own the property square, but the other player does (i.e. when rent is due). An initial sequence of actions is carried out in order to determine the amount of rent due, based on the square details (base rent), and the level of development on such (i.e. any floors or extensions added) in context of the current player's bank balance (i.e. can they pay the rent). An additional sequence of actions is then modelled based on the ability of the current player to pay. If the rent due is less than the current player's balance, the system will decrement the current player's balance with the rent amount and increment the other player's balance with same, with both then displayed to confirm the transaction. If the rent due exceeds the current player's balance, the system will revert to the game termination sequence, declaring the current player bankrupt, the other player the winner and displaying the bank balance for both to confirm the final game standings.

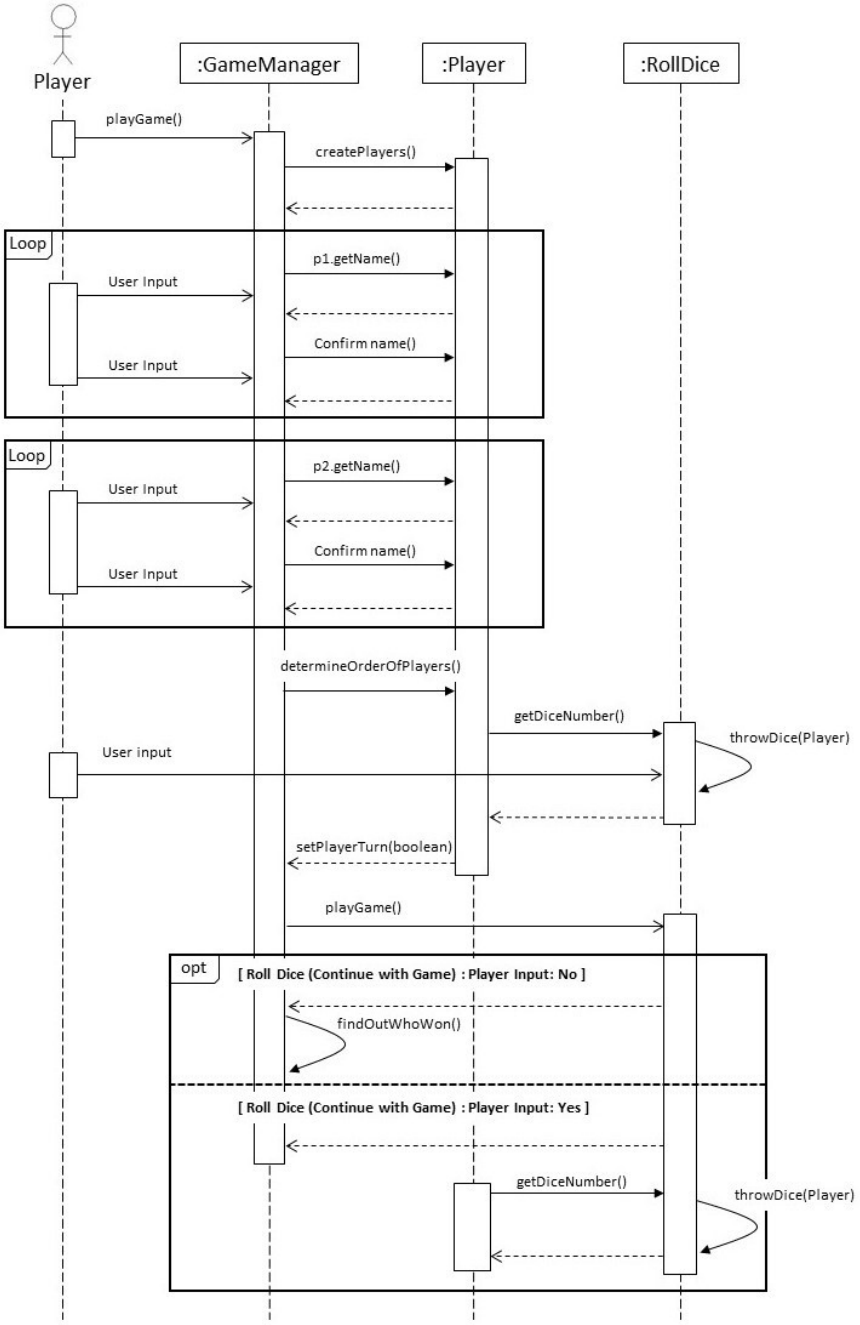
3.2 Developed Sequence Diagrams

PB, JK

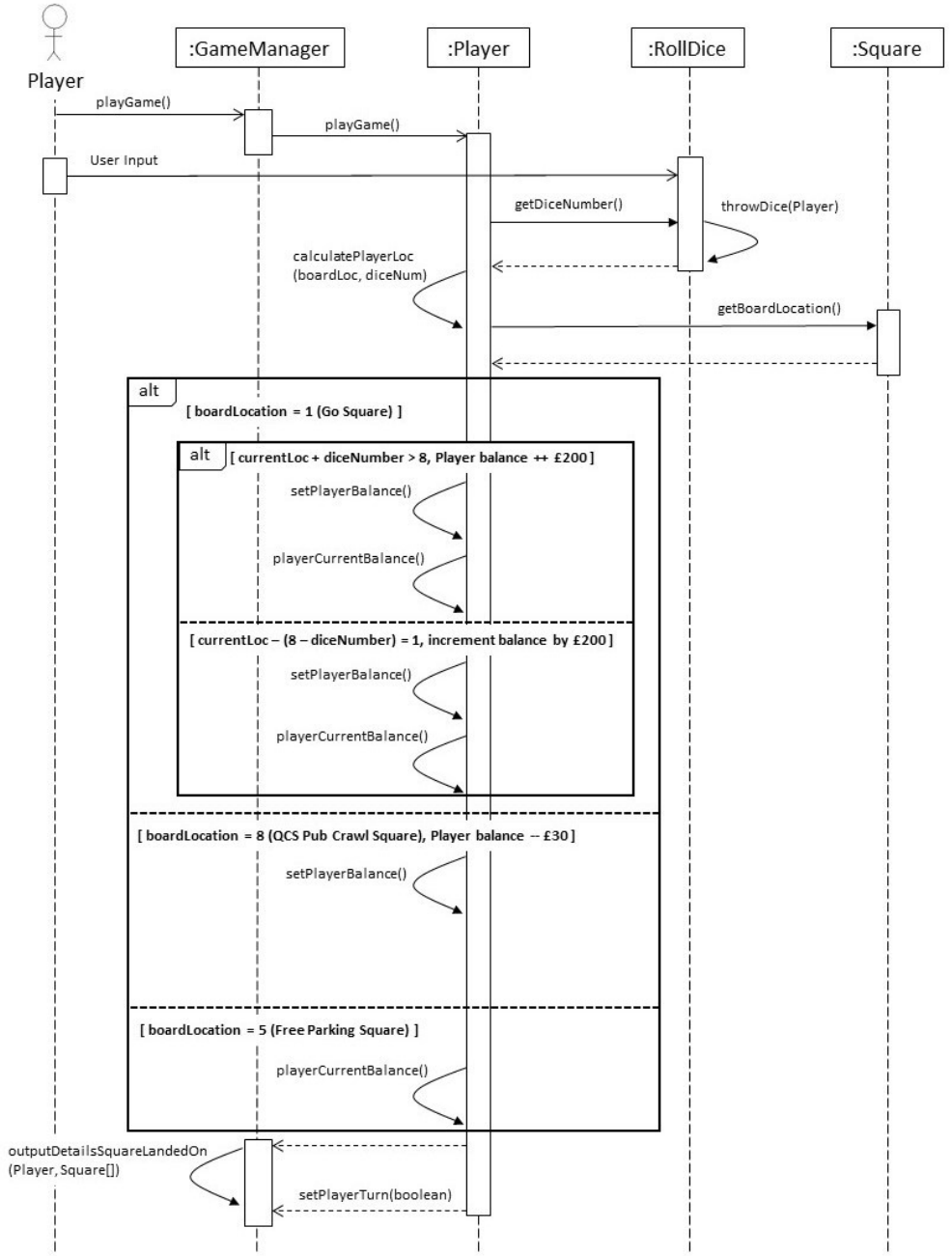
The sequence diagrams previously discussed are presented as follows (overleaf):

PB, JK

Inset 3.1: Addition of Player Names and Termination of Game

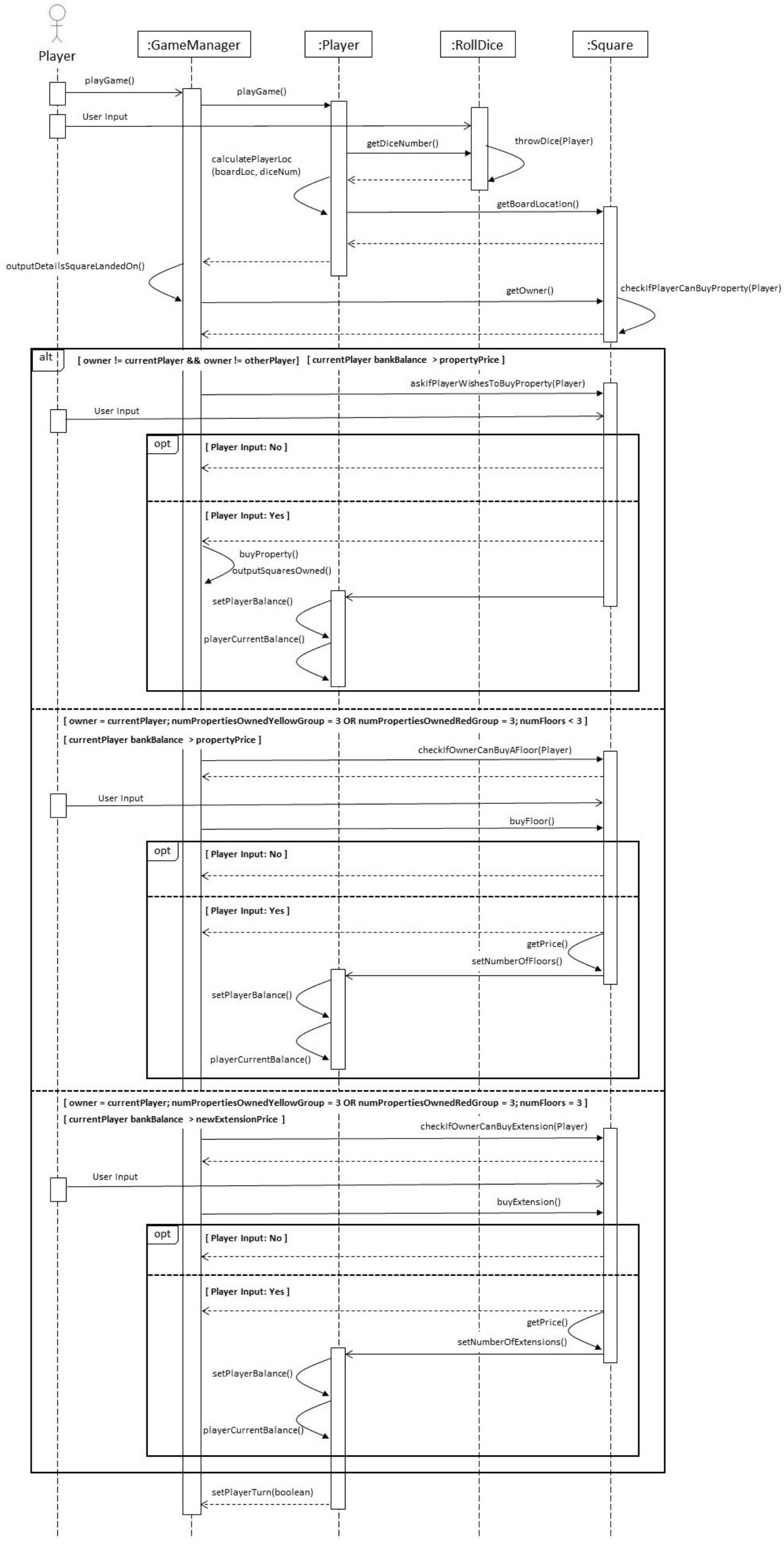


Inset 3.2: Moving to a Square (except a Property Square)



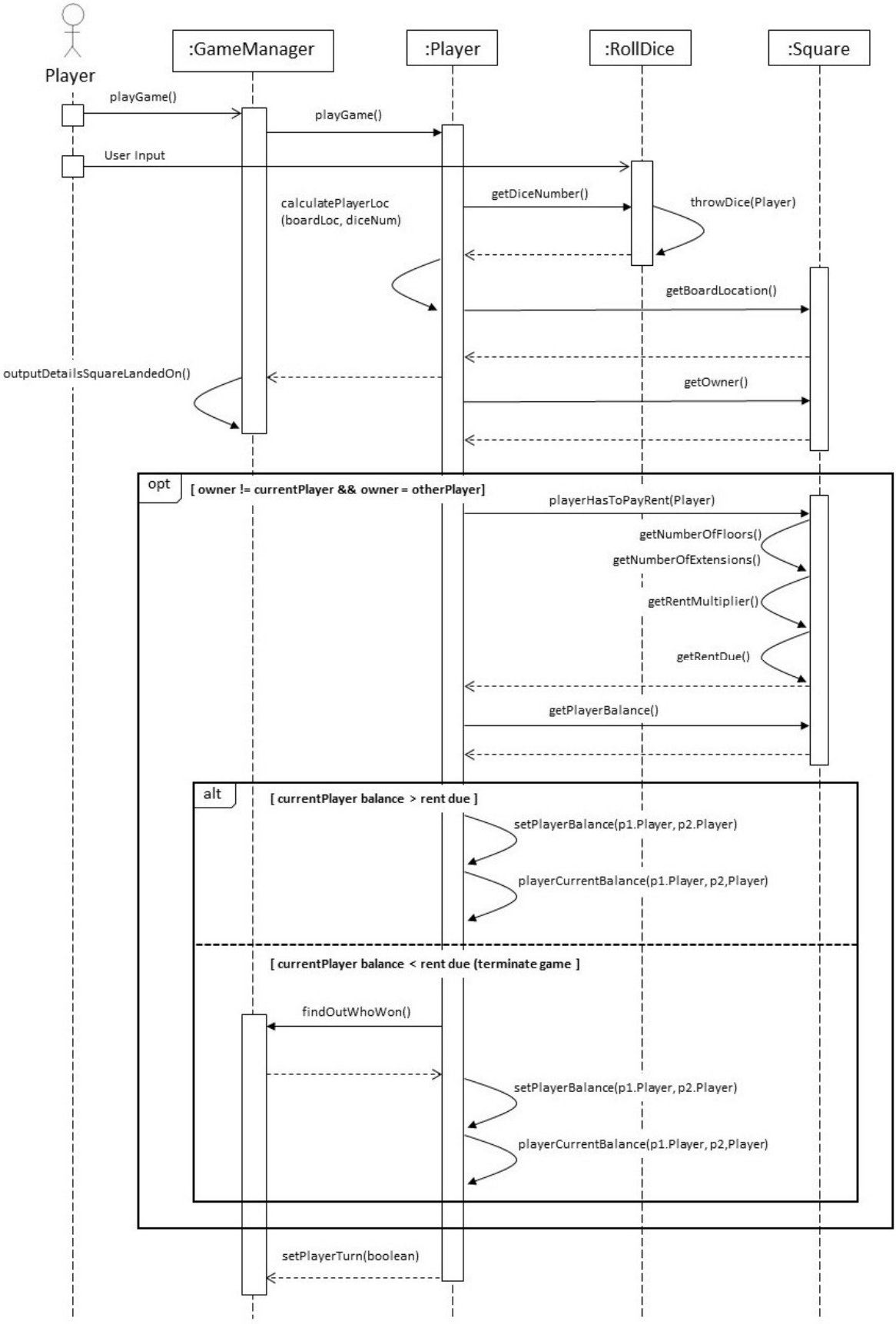
PB, JK

Inset 3.3: Moving to a Property Square (including purchase of a property, adding a floor and adding an extension)



Inset 3.4: Landing on a Property Square and Paying Rent

PB, JK



4.0 System Design – UML Class Diagram

PB

The sequence diagrams discussed in the previous section and specifically the interaction between the objects modelled were translated into a Class Diagram, the structure of such adhering to the UML principles. The final design of the Class Diagram is presented in Inset 4.1 (overleaf).

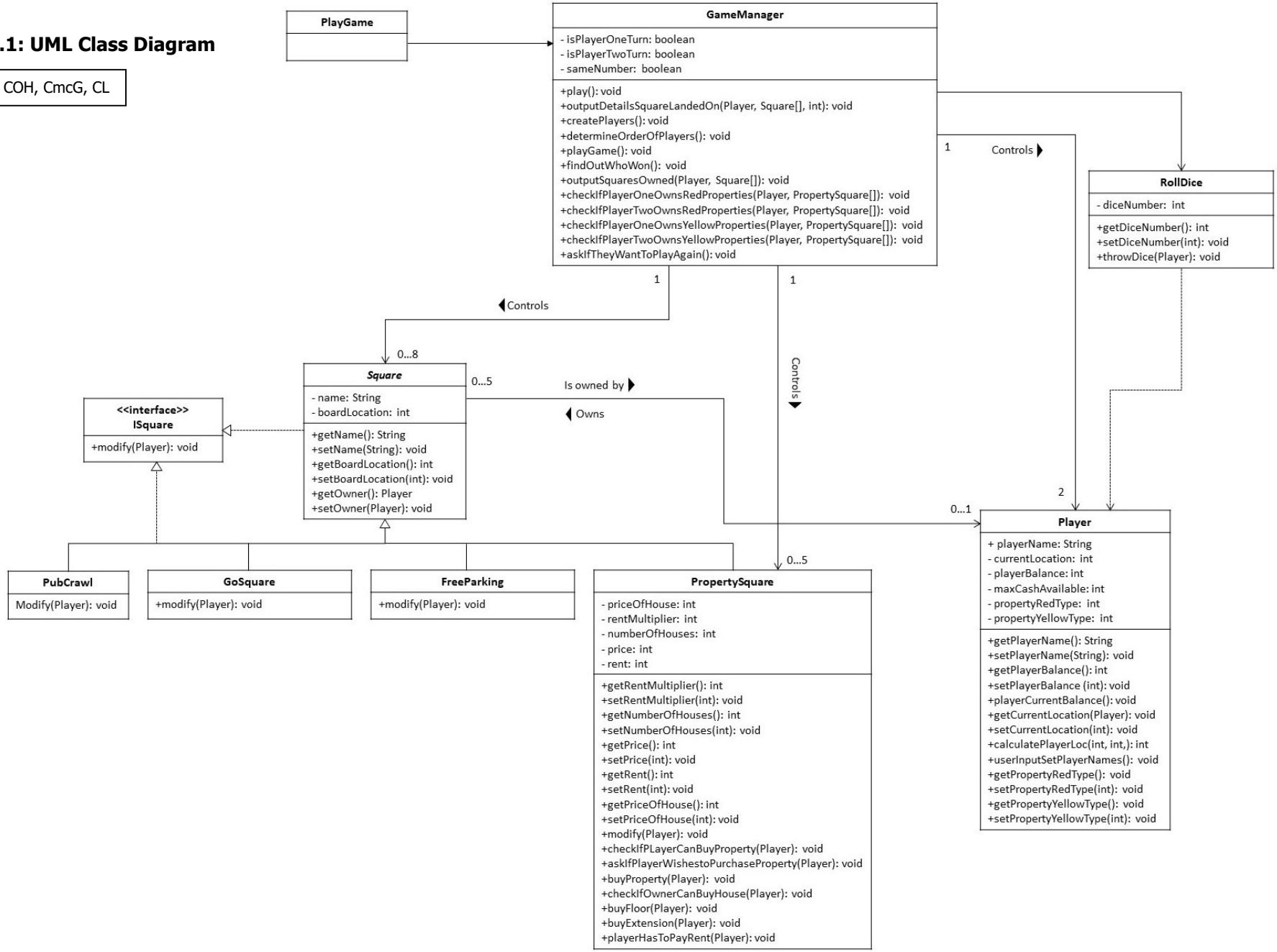
The objects (instances of classes) and even methods identified in the sequence diagrams were transposed and integrated into a range of various Java classes which the development team felt were necessary to provide the necessary system functionality as required by the Project Specification. An initial basic structure was developed and, as the team progressed through the development lifecycle (as discussed further in Section 5.0), this structure evolved until the optimal structure was arrived at. This development phase included the identification of the attributes and methods that were contained within appropriate class types (i.e. abstract, interface, super classes and associated sub-classes), and the subsequent definition of the relationships between such.

Where appropriate, the relationship between classes was annotated to clearly describe the multiplicity arrangements between such, with a relationship descriptor (verb descriptor) along with an arrow to identify the direction in which that particular relationship descriptor should be read. For example, the relationship between the Square and Player classes was annotated to ensure the diagram clearly identified that 0 to 5 squares could be owned by 0 or 1 players, as demonstrated in the simplified diagram below:

This subsequently provided the foundation on which the development of the system code was carried out, where the classes, methods and attributes (instance variables) outlined in the Class Diagram were used as the framework around which the Java code was written/developed. The clear advantage of using the Class Diagram as a reference point was that it maintained consistency in approach by the development team, specifically where the relationships between the various classes were maintained and enhanced. This approach also enforced consistency in the naming conventions for the methods, ensuring any methods referenced by another developer (i.e. called in a different class) should effectively function when the system was integrated and initiated.

Inset 4.1: UML Class Diagram

PB, JK, COH, CmcG, CL



5.0 Project Delivery Process

5.1 Project Management

PB

As previously discussed in Section 1.3, the project was progressed in accordance with the development programme (Appendix A). Regular team meetings (the frequency of which were increased as the deadline approached) were held to maintain a high degree of communication between the team, whilst informal communication outside of formal meetings was also encouraged and carried out. A comprehensive set of minutes for each meeting is provided in Appendix C.

Mapping out the key phases of the project at the earliest possible stage facilitated compliance with the project specification within the required timeframe. Deadlines for key milestones were planned to ensure a sufficient amount of time (plus some padding) was provided for each task. The 'padding' incorporated into each task essentially meant the programme was inherently flexible, where extra time for a specific task (before the deadline) would likely be found if required. Importantly, benchmarking actual progress against the planned progress during both the formal meetings and through the informal inter-team communication platforms (meeting members during the normal daily routine, and using a group messaging service) was key to ensure the delivery of the key objectives remained as planned. Consequently, every milestone and deadline prescribed in the project programme (Appendix A) was met, ensuring the requirements, realisation, design and development processes (including testing) proceeded in a timely manner, negating the need for any final 'sprint' or compressed periods of working to complete tasks.

5.2 Software Development Process

PB, CMcG

A sequential approach to the initial design tasks (Requirements Analysis, Realisation and Design) was adopted, with an iterative incremental development methodology (IID) approach applied to the system development phase (programming) that provided greater flexibility in the process. In applying the IID methodology, the team identified three key components of the process (system coding, unit testing and functionality testing) within which the completion of a working system would be developed around. Our IID approach enabled each developer to code and test each system component (class) concurrently, rather than completing all the coding and then testing at the end of the process (i.e. a 'Big Bang' at the end of an incremental 'Waterfall' approach). This approach, in conjunction with robust project delivery planning, allowed the team to effectively meet the requirements of the project without implementing a final 'Sprint' as the deadline approached.

5.3 Test Plan

PB
COH

Testing is the systematic attempt to find faults in a planned way in the implemented software (Bruegge & Dutoit, 2009[1]). It is also critical that test planning starts early in the development process, therefore we undertook test planning and implementation as soon as our use case model was stable to reduce impact of any defects identified, provide sufficient time to address such, and to allow as much testing to be carried out within the defined, restrictive timescale. Similarly, as exhaustive testing is not possible, a planned and targeted approach focussing on identified risk areas will yield the best results within a defined timeframe. The testing regime the team applied to the process incorporated developer led component testing, static testing, system testing and a recorded system demonstration.

Component testing was conducted by the individual developers during the coding stages where functionality was tested as they developed and using the Integrated Development Environment (IDE) to ensure correct use of syntax was maintained.

PB

Once a functional system was completed, the testing team conducted Static Testing, where a suite of Test Conditions, Cases and Procedures were developed and implemented based on the core Requirements as outlined in Table 2.1. The outputs from this testing phase are presented in Appendix D. During the initial static testing phase, a total of 27 test conditions were developed, from which 41 test cases and associated test procedures were developed. Of the 41 test cases, 8 failed the initial testing phase (9.5%), resulting in the identification of 5 out of 13 requirements (38.5%) not being fully met. These test failures (defects) were duly recorded and passed back to the development team with sufficient information to allow them to be addressed. Once the developer(s) felt they were addressed, only the failures/defects were tested (due to time constraints), where it was subsequently confirmed all passed. This allowed the defects to be formally closed off, recorded as such and the team to declare that 100% of the core system requirements were met.

COH
JK

Testing using JUnit functionality in the IDE was carried out confirm that each subsystem is correctly coded and carries out the intended functionality. This primarily focused on high priority areas such as the getters and setters and any validation due to times pressures. For example, testing validation added to ensure a player's location could only be within specific prescribed limits. This process was developer led, where any issues identified were addressed (such as amendments to validation to ensure business rules were complied with) and re-tested by the developer to ensure the issue was closed off. During this process, only a small number of issues were identified, successfully addressed by the developer and re-tested to confirm system integrity and compliance with business rules was maintained.

CMcG
CL

Finally, to further confirm compliance with the system requirements the team recorded a system test demonstrating the implementation of each core requirement in the working system.

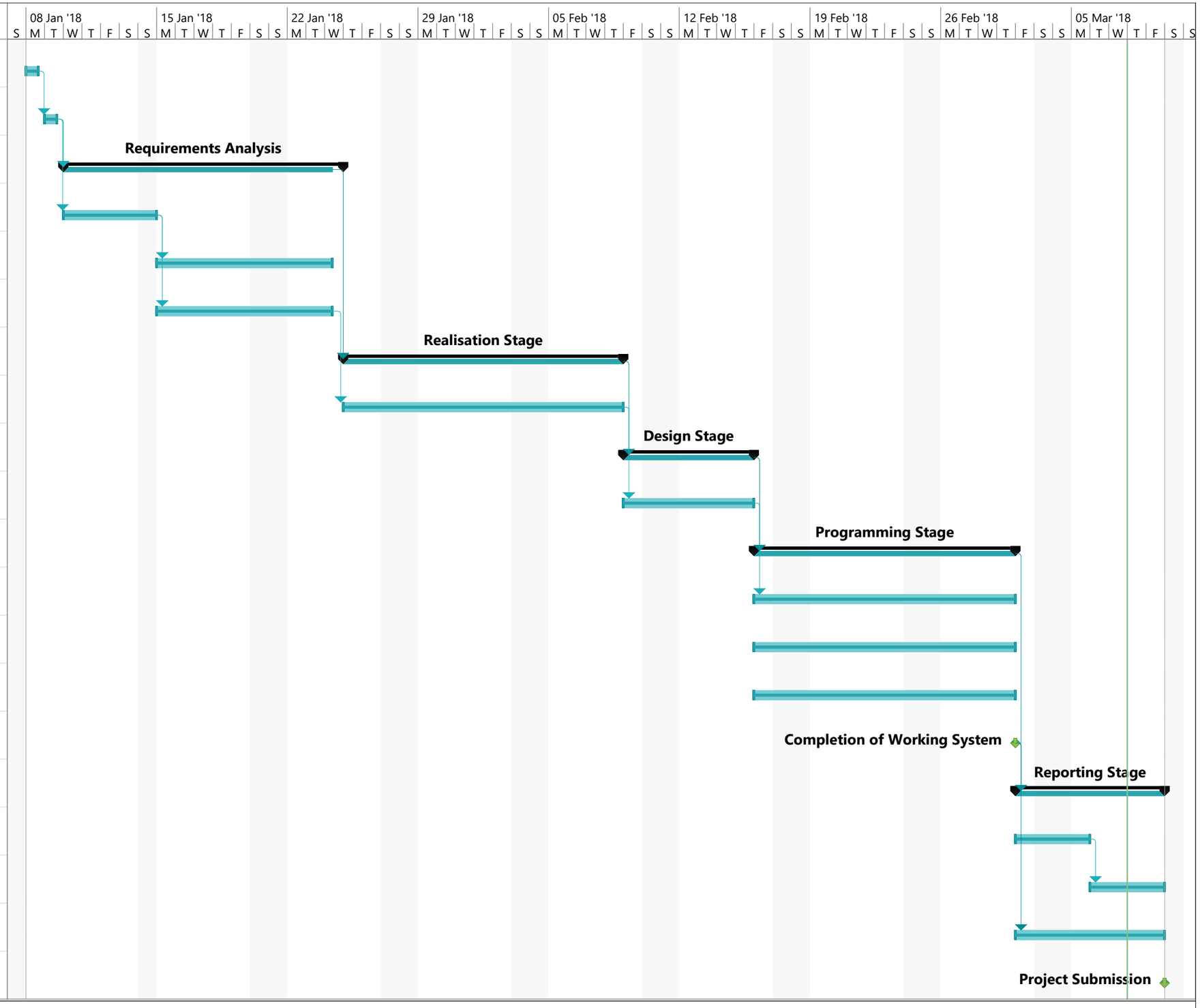
[1] Bruegge & Dutoit. Object-oriented Software Engineering. Pearson, 2009.

APPENDICES

PB

Appendix A: Project Delivery Programme

ID	Task Name	Start	Finish
1	Project Issue	Mon 08/01/18	Mon 08/01/18
2	Project kick-off meeting	Tue 09/01/18	Tue 09/01/18
3	**Requirements Analysis**	**Wed 10/01/18**	**Wed 24/01/18**
4	Game Board Design + Game Guide	Wed 10/01/18	Sun 14/01/18
5	UML Case Diagram Design	Mon 15/01/18	Wed 24/01/18
6	UML Use Case Descriptions	Mon 15/01/18	Wed 24/01/18
7	**Realisation Stage**	**Wed 24/01/18**	**Thu 08/02/18**
8	UML Sequence Diagrams Design	Wed 24/01/18	Thu 08/02/18
9	**Design Stage**	**Fri 09/02/18**	**Thu 15/02/18**
10	UML Class Diagram	Fri 09/02/18	Thu 15/02/18
11	**Programming Stage**	**Fri 16/02/18**	**Thu 01/03/18**
12	System Coding	Fri 16/02/18	Thu 01/03/18
13	Unit Testing	Fri 16/02/18	Thu 01/03/18
14	Functionality Testing	Fri 16/02/18	Thu 01/03/18
15	Completion of Working System	Thu 01/03/18	Thu 01/03/18
16	**Reporting Stage**	**Fri 02/03/18**	**Fri 09/03/18**
17	Draft Project Report	Fri 02/03/18	Mon 05/03/18
18	Finalise Report for Issue	Mon 05/03/18	Fri 09/03/18
19	Recording Demonstration Video	Fri 02/03/18	Fri 09/03/18
20	**Project Submission**	Fri 09/03/18	Fri 09/03/18



Project: Smickopoly
Date: Thu 08/03/18

Task	Project Summary	Manual Task	Start-only	Deadline
Split	Inactive Task	Duration-only	Finish-only	Progress
Milestone	Inactive Milestone	Manual Summary Rollup	External Tasks	Manual Progress
Summary	Inactive Summary	Manual Summary	External Milestone

Appendix B: Property Values and Rent Costs

Table B.1: Property Square Costs and Rent Values

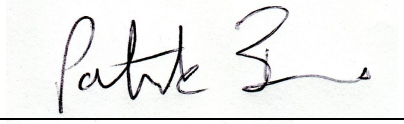
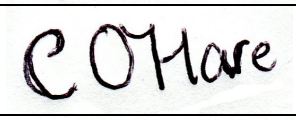
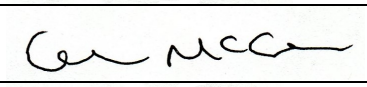
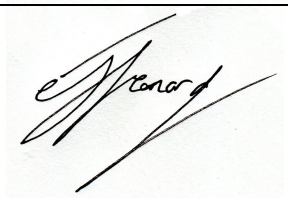
Site	Square Cost (£)	Development Costs		Rent Costs				
		Per Floor (£)	Extension (£)	No Floors (£)	1 Floor (£)	2 Floors (£)	3 Floors (£)	Extension (£)
CSB	120	60	120	24	48	72	96	144
DKB	130	65	130	26	52	78	104	156
PEC	140	70	140	28	56	84	112	168
The Bot	190	95	190	38	76	114	152	228
The Hatfield	200	100	200	40	80	120	160	240

CL

Appendix C: Meeting Minutes

Minutes for Team G19___ Week commencing 2_____ Date of this minute 17/1/18_____

The following team members were present

Name (printed/typed)	Signature
Patrick Burns	
Catherine O'Hare	
Colm McGoldrick	
Conor Leonard	
Justin Kelly	

Task Reporting:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Use Case Modelling.
- Use Case Description, Author.
-
-

Name & Role (2):Catherine O'Hare, Team Member.

- Use Case Modelling.
- Use Case Descriptions, Author.
-
-

Name & Role (3): Colm Mc Goldrick, Team Member.

- Use Case Modelling.
- Use Case Descriptions, Author.
-
-

Name & Role (4):Conor Leonard, Team Member.

- Use Case Modelling.
- Use Case Descriptions, Author.
-
-

Name & Role (5):Justin Kelly, Team Member.

- Use Case Modelling.
- Use Case Descriptions, Author
-
-

*Printouts giving an overview of interim deliverables may be added as a supplement to these minutes.

Actions Planned:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Evolve their prescribed Use Cases.
- Use Cases: Buy Property, Add Premises to property, Add entertainment complex to property, Buy QUB facility.
-
-

Name & Role (2): Catherine O'Hare, Team Member.

- Evolve their prescribed Use Cases.
- Use cases: Move, Roll Dice, View Player status.
-
-

Name & Role (3): Colm Mc Goldrick, Team Member.

- Evolve their prescribed Use Cases.
- Use cases: Get out of Jail, Visit Jail, Enter Name, Pass Go.
-
-

Name & Role (4): Conor Leonard, Team Member.

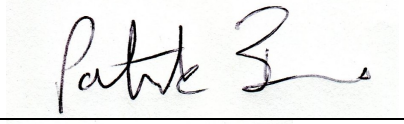
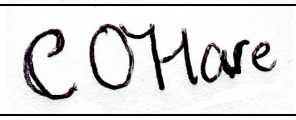
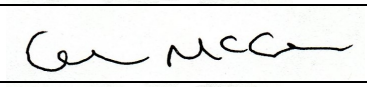
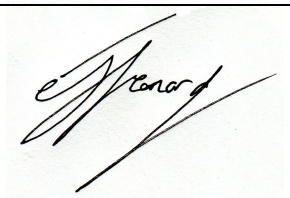
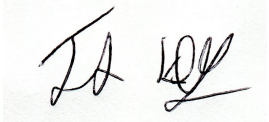
- Evolve their prescribed Use Cases.
- Use Cases: Go to Jail, Go to QUB facility, Go to Free Parking, Go to Property.
-
-

Name & Role (5): Justin Kelly, Team Member.

- Evolve their prescribed Use Cases.
- Use Cases: Go to Chance, Take a chance Card, pay rent.
-
-

Minutes for Team G19___ Week commencing 3_____ Date of this minute 24/1/18_____

The following team members were present

Name (printed/typed)	Signature
Patrick Burns	
Catherine O'Hare	
Colm McGoldrick	
Conor Leonard	
Justin Kelly	

Task Reporting:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Use case diagrams completed based on initial design.
-
-
-

Name & Role (2):Catherine O'Hare, Team Member.

- Use case diagrams completed based on initial design.
-
-
-

Name & Role (3): Colm McGoldrick, Team Member.

- Use case diagrams completed based on initial design.
-
-
-

Name & Role (4):Conor Leonard, Team Member.

- Use case diagrams completed based on initial design.
-
-
-

Name & Role (5):Justin Kelly, Team Member.

- Use case diagrams completed based on initial design.
-
-
-

*Printouts giving an overview of interim deliverables may be added as a supplement to these minutes.

Actions Planned:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Edit the Use Cases prescribed to better suit the specification.
- Sequence diagrams to be drawn based on the new use case diagram design
-
-

Name & Role (2): Catherine O'Hare, Team Member.

- Edit the Use Cases prescribed to better suit the specification.
- Sequence diagrams to be drawn based on the new use case diagram design
-
-

Name & Role (3): Colm McGoldrick, Team Member.

- Edit the Use Cases prescribed to better suit the specification.
-
-
-

Name & Role (4): Conor Leonard, Team Member.

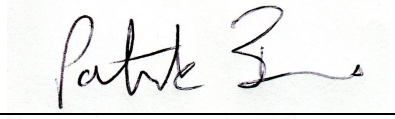
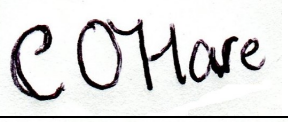
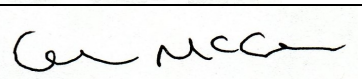
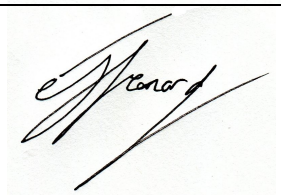
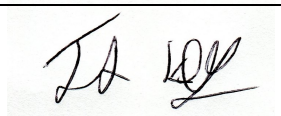
- Edit the Use Cases prescribed to better suit the specification.
- Sequence diagrams to be drawn based on the new use case diagram design
-
-

Name & Role (5): Justin Kelly, Team Member.

- Edit the Use Cases prescribed to better suit the specification.
- Sequence diagrams to be drawn based on the new use case diagram design
-
-

Minutes for Team G19___ Week commencing 4_____ Date of this minute 30/1/18_____

The following team members were present

Name (printed/typed)	Signature
Patrick Burns	
Catherine O'Hare	
Colm McGoldrick	
Conor Leonard	
Justin Kelly	

Task Reporting:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Finalised use cases from previous week
- Completed individual sequence diagrams
-
-

Name & Role (2):Catherine O'Hare, Team Member.

- Finalised use cases from previous week
- Completed individual sequence diagrams
-
-

Name & Role (3): Colm McGoldrick, Team Member.

- Finalised use cases from previous week
- Completed individual sequence diagrams
-
-

Name & Role (4):Conor Leonard , Team Member.

- Finalised use cases from previous week
- Completed individual sequence diagrams
-
-

Name & Role (5):Justin Kelly , Team Member.

- Finalised use cases from previous week
- Completed individual sequence diagrams
-
-

*Printouts giving an overview of interim deliverables may be added as a supplement to these minutes.

Actions Planned:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Agreed to look at implementing a class diagram
-
-
-

Name & Role (2): Catherine O Hare, Team Member.

- Agreed to look at implementing a class diagram
-
-
-

Name & Role (3): Colm McGoldrick, Team Member.

- Agreed to look at implementing a class diagram
-
-
-

Name & Role (4): Conor Leonard, Team Member.

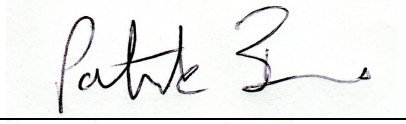
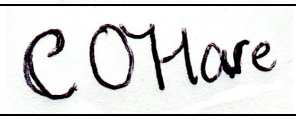
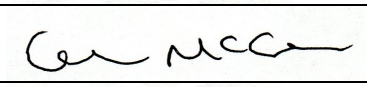
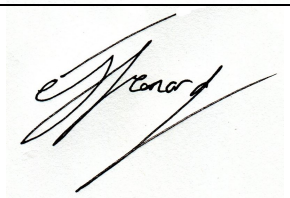
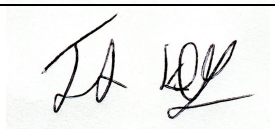
- Agreed to look at implementing a class diagram
-
-
-

Name & Role (5): Justin Kelly, Team Member.

- Agreed to look at implementing a class diagram
-
-
-

Minutes for Team G19___ Week commencing 5_____ Date of this minute 9/2/18_____

The following team members were present

Name (printed/typed)	Signature
Patrick Burns	
Catherine O'Hare	
Colm McGoldrick	
Conor Leonard	
Justin Kelly	

Task Reporting:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Brought ideas for class diragram
- Helped to implement and complete the class diagram
-
-

Name & Role (2):Catherine O'Hare, Team Member.

- Brought ideas for class diragram
- Helped to implement and complete the class diagram
-
-

Name & Role (3): Colm McGoldrick, Team Member.

- Brought ideas for class diragram
- Helped to implement and complete the class diagram
-
-

Name & Role (4):Conor Leonard, Team Member.

- Brought ideas for class diragram
- Helped to implement and complete the class diagram
-
-

Name & Role (5):Justin Kelly, Team Member.

- Brought ideas for class diragram
- Helped to implement and complete the class diagram
-
-

- *Printouts giving an overview of interim deliverables may be added as a supplement to these

Actions Planned:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Divided up the specification
- Look into completing system tests
-

Name & Role (2): Catherine O'Hare, Team Member.

- Divided up the specification
- Look into completing system tests
-
-

Name & Role (3): Colm McGoldrick, Team Member.

- Divided up the specification
- Look into completing system tests
-
-

Name & Role (4): Conor Leonard, Team Member.

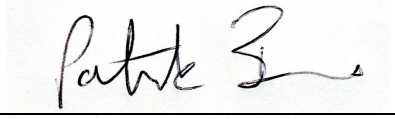
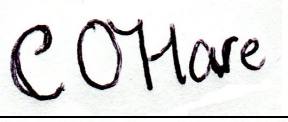
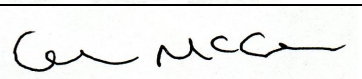
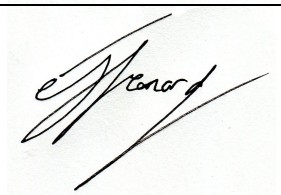
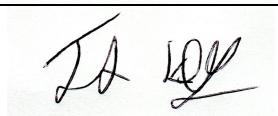
- Divided up the specification
- Look into completing system tests
-
-

Name & Role (5): Justin Kelly, Team Member.

- Divided up the specification
- Look into completing system tests
-
-

Minutes for Team G19___ Week commencing 6_____ Date of this minute 16/2/18_____

The following team members were present

Name (printed/typed)	Signature
Patrick Burns	
Catherine O'Hare	
Colm McGoldrick	
Conor Leonard	
Justin Kelly	

Task Reporting:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Confirm the system tests against the specification
- Ensure the tests are properly written.
-
-

Name & Role (2):Catherine O'Hare, Team Member.

- Confirm the system tests written against the specification.
- Ensure the tests are properly written
-
-

Name & Role (3): Colm McGoldrick, Team Member.

- Confirm the system tests written against the specification
- Ensure the tests are properly written.
-
-

Name & Role (4):Conor Leonard, Team Member.

- Confirm the system tests written written against the specification.
- Ensure the tests are properly written.
-
-

Name & Role (5):Justin Kelly, Team Member.

- Confirm the system tests written written against the specification.
- Ensure the tests are properly written.
-
-

*Printouts giving an overview of interim deliverables may be added as a supplement to these minutes.

Actions Planned:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Classes divided up from the class diagram
- Create the code for Class: Player Status, Player Details.
- Create Unit tests for the prescribed code.
-

Name & Role (2): Catherine O'Hare, Team Member.

- Classes divided up from the class diagram
- Create the code for Class: Player Move.
- Create Unit tests for the prescribed code.
-

Name & Role (3): Colm McGoldrick, Team Member.

- Classes divided up from the class diagram
- Create the code for Class: DevelopingProperty, PropertyStatus.
- Create Unit tests for the prescribed code.
-

Name & Role (4): Conor Leonard, Team Member.

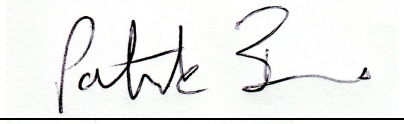
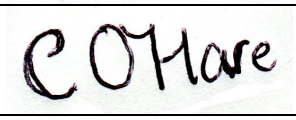
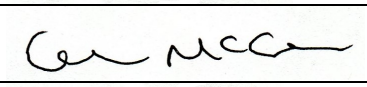
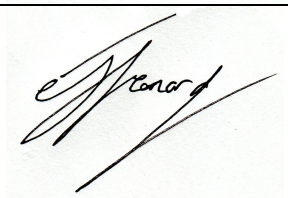
- Classes divided up from the class diagram
- Create the code for Class: Dice.
- Create Unit tests for the prescribed code.
-

Name & Role (5): Justin Kelly, Team Member.

- Classes divided up from the class diagram
- Create the code for Class: Square, PropertySquare, PubCrawlSquare,FreeParkingSquare,GoSquare.
- Create Unit tests for the prescribed code.
-

Minutes for Team G19___ Week commencing 7_____ Date of this minute 23/2/18_____

The following team members were present

Name (printed/typed)	Signature
Patrick Burns	
Catherine O'Hare	
Colm McGoldrick	
Conor Leonard	
Justin Kelly	

Task Reporting:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Collate the code created from the week previous.
- System test.
-
-

Name & Role (2):Catherine O'Hare, Team Member.

- Collate the code created from the week previous.
- System test.
-

Name & Role (3): Colm McGoldrick, Team Member.

- Begin Documentation and writing the report.
-
-
-

Name & Role (4):Conor Leonard, Team Member.

- Begin Documentation and writing the report
-
-
-

Name & Role (5):Justin Kelly, Team Member.

- Collate the code created from the week previous.
- System test.
-
-

*Printouts giving an overview of interim deliverables may be added as a supplement to these minutes.

Actions Planned:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Review Class diagram, sequence diagrams,ensuring consistency in documentation.
-
-
-

Name & Role (2): Catherine O'Hare, Team Member.

- Review Class diagram, sequence diagrams,ensuring consistency in documentation.
-
-
-

Name & Role (3): Colm McGoldrick, Team Member.

- Review Class diagram, sequence diagrams,ensuring consistency in documentation.
- Beging to formulate the requirements analysis and realisation section in the report
-
-

Name & Role (4): Conor Leonard, Team Member.

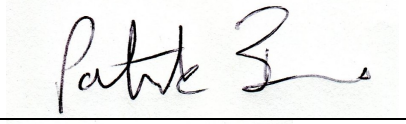
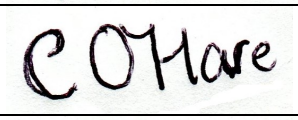
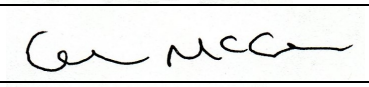
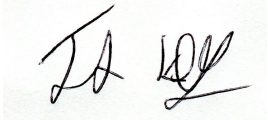
- Review Class diagram, sequence diagrams,ensuring consistency in documentation.
- Begin to formulate the requirements analysis and realisation section in the report.
-
-

Name & Role (5): Justin Kelly, Team Member.

- Review Class diagram, sequence diagrams,ensuring consistency in documentation.
-
-
-

Minutes for Team G19___ Week commencing 8_____ Date of this minute 1/3/18_____

The following team members were present

Name (printed/typed)	Signature
Patrick Burns	
Catherine O Hare	
Colm Mc Golrick	
Justin Kelly	

Task Reporting:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Class diagrams and sequence diagrams reviewed
- Review the coding of the Monoversity project
-
-

Name & Role (2):Catherine O'Hare, Team Member.

- Class diagrams and sequence diagrams reviewed
- Review the coding of the Monoversity project
-

Name & Role (3): Colm McGoldrick, Team Member.

- Class diagrams and sequence diagrams reviewed
- Requiremnts analysis and realisation stage of the report documented.
-
-

Name & Role (4):Conor Leonard, Team Member.

- Absent
-
-
-

Name & Role (5):Justin Kelly, Team Member.

- Class diagram and sequence diagrams reviewed
- Review the coding of the Monoveristy project.
-
-

*Printouts giving an overview of interim deliverables may be added as a supplement to these minutes.

Actions Planned:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Work on the report
- Review the use case descriptions
- Review the sequence diagrams
-

Name & Role (2): Catherine O'Hare, Team Member.

- Work on the report
- Review the class diagram
- Prepare static testing document
-

Name & Role (3): Colm McGoldrick, Team Member.

- Work on the report
- Review the sequence diagrams
-
-

Name & Role (4): Conor Leonard, Team Member.

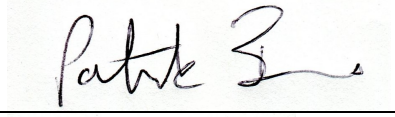
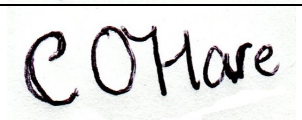
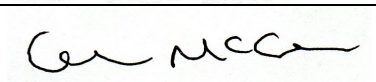
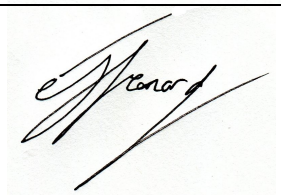
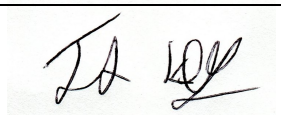
- Review the use case description
- Work on the report
-
-

Name & Role (5): Justin Kelly, Team Member.

- Work on the report
- Review the game code to ensure consistency
-
-

Minutes for Team G19___ Week commencing 9_____ Date of this minute 8/03/18_____

The following team members were present

Name (printed/typed)	Signature
Patrick Burns	
Catherine O'Hare	
Colm McGoldrick	
Conor Leonard	
Justin Kelly	

Task Reporting:

Name & Role (1): Patrick Burns, Moderator, Team Lead.

- Finalise use cases
- Finalise sequence diagram
- Review and complete report
-

Name & Role (2): Catherine O'Hare, Team Member.

- Finalise class diagram
- Finalise static testing document
- Review and complete report
-

Name & Role (3): Colm McGoldrick, Team Member.

- Finalise sequence diagrams
- Review and complete report
-
-

Name & Role (4): Conor Leonard, Team Member.

- Finalise use case descriptions
- Review and complete report
-
-

Name & Role (5): Justin Kelly, Team Member.

- Finalise code
- Review and complete report
-
-

*Printouts giving an overview of interim deliverables may be added as a supplement to these minutes.

PB

Appendix D: Static Testing Outputs

Project Requirements

Req	Description
R1	The game has two players, who start with the same amount of money. Their names should be entered.
R2	The players take turns.
R3	They throw a virtual die.
R4	There is a start square, which players land on or pass to pick up their salary/grant money/funding .
R5	There are two property groups, one comprising two properties and another comprising three.
R6	The properties in the two-property group are more expensive than the ones in the three-property group.
R7	Before you can develop a group by building on it, you must own the *whole* group,
R8	The game will have a university- and specifically a Queen's-related theme.
R9	Three 'houses'/*labs* are needed before you can build or establish a 'hotel' (or whatever you decide it is).
R10	Players taking a turn are told where they have landed and what their obligations or opportunities are.
R11	Where appropriate, they may indicate their choice of action.
R12	If a player's finances have changed, the system indicates the reason for the change and announces the player's new bank balance.
R13	When one player goes bankrupt, the other is declared the winner. If one player no longer wants to play, the game ends. In both cases, the players' bank balance is given.

Test Conditions

Test Condition ID	Description	Source	Priority
TCon_1	To show that the game has two players.	R1	Medium
TCon_2	To show that each player starts with the same amount of money.	R1	Medium
TCon_3	To show that each player is required to enter their name.	R1	Medium
TCon_4	To show that the two players take a turn each in a sequential manner.	R2	Medium
TCon_5	To show that each player throws a virtual die.	R3	Medium
TCon_6	To show that there is a Start/Go square.	R4	Medium
TCon_7	To show that each player is allocated a set amount when they land on the Start/Go square.	R4	Medium
TCon_8	To show that each player is allocated a set amount when they pass the Start/Go square.	R4	Medium
TCon_9	To show that there are two distinct property groups.	R5	Medium
TCon_10	To show that one property group consists of two property squares.	R5	Medium
TCon_11	To show that one property group consists of three property squares.	R5	Medium
TCon_12	To show that the properties in the two-property group are more expensive than those in the three-property group.	R6	Medium
TCon_13	To show that a player cannot develop a property square in (i.e. add a house or similar depending on the game choices) that belongs to the two-property group unless they own all property squares in that group.	R7	Medium
TCon_14	To show that a player cannot develop a property square (i.e. add a house or similar depending on the game choices) that belongs to the three-property group unless they own all property squares in that group.	R7	Medium
TCon_15	To show that the game has a Queen's University theme.	R8	Medium
TCon_16	To show that the a player cannot build a hotel (or similar depending on the game choices) on a property belonging to the two-property group unless each square in that group already has three houses (or similar) built on it.	R9	Medium
TCon_17	To show that the a player cannot build a hotel (or similar depending on the game choices)on a property belonging to the two-property group unless each square in that group already has three houses (or similar) built on it.	R9	Medium
TCon_18	To show that each player is told what square they have landed on.	R10	Medium
TCon_19	To show that each player is informed of any required actions (obligations and/or opportunities) when they land on a square.	R10	Medium
TCon_20	To show that, when a player lands on a square and is presented with an opportunity, they may indicate their choice of action.	R11	Medium
TCon_21	To show that the players current bank balance is displayed following any increments or decrements resulting from board and/or player actions.	R12	Medium
TCon_22	To show that the reason for any increment or decrement to a player's bank balance is clearly provided.	R12	Medium
TCon_23	To show that once one player is declared bankrupt, the other player is declared the winner.	R13	Medium
TCon_24	To show that players are given the opportunity to terminate the game if desired.	R13	Medium
TCon_25	To show that, in the event a player elects to terminate the game, the other player is declared the winner.	R13	Medium
TCon_26	To show that, upon game termination due to bankruptcy, the bank balance for both players is displayed.	R13	Medium
TCon_27	To show that, if a player elects to terminate the game, the bank balance for both players is displayed.	R13	Medium

Test Cases

Test case ID	Objective	Preconditions	Test data	Expected Result	Test condition(s)	Priority	Test completion date	Status	Tester	Defect ID	Defect Status	Defect Severity	Open Date	Close Date	Description / Comment	Priority
TCase_1	Check that the game has two players.	In Eclipse, in the monoversity package and class, running the PlayGame class, with player names entered.	N/A	The console will require input/responses for two players only (Player One and Player Two).	TCon_1 TCon_3	Medium	02/03/2018	Passed	PBurns							
TCase_2	Check that each player starts with the same amount of money.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	At the start of the game (once names have been entered) the balance for Player One and Player Two will both indicate they each have £500.	TCon_2	Medium	02/03/2018	Passed	PBurns							
TCase_3	To show that each player is required to enter their name.	In Eclipse, in the monoversity package and class, running the PlayGame class.	Player one name: TestName1; Player two name: TestName2;	The console will require the users to input details for two players only.	TCon_3	Medium	02/03/2018	Passed	PBurns							
TCase_4	Check that the two players take a turn each in a sequential manner.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	Player One will be offered a turn firstly, and once that turn is completed Player Two will be prompted to take their turn. Once Player Two has finished their turn, the cycle will repeat.	TCon_4	Medium	02/03/2018	Passed	PBurns							
TCase_5	Check that each player throws a virtual die.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	The system will prompt Player One and Two to roll a virtual die, and then output a random number.	TCon_5	Medium	02/03/2018	Passed	PBurns							
TCase_6	Check that there is a Start/Go square.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	If the Player rolls a dice number that, when calculating board position based on the game guide, is anticipated to take them past the start/Go square, The system will inform the Players that they have passed a Start/Go square.	TCon_6	Medium	02/03/2018	Passed	PBurns							
TCase_7	Check that each player is allocated a set amount when they land on the Start/Go square.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	If the Player rolls a dice number that, when calculating board position based on the game guide, they are anticipated to land on the Start/Go square, the system increment their bank balance by £200 and will output their balance to screen reflecting such.	TCon_7	Medium	02/03/2018	Failed	PBurns	DEF_1	Closed	Med	21/02/2018	26/02/2018	The player's balance is not incremented by £200 when they land on the Go square.	High
TCase_8	Check that each player is allocated a set amount when they pass the Start/Go square.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	If the Player rolls a dice number that, when calculating board position based on the game guide, is anticipated to take them past the start/Go square, the system increment their bank balance by £200 and will output their balance to screen reflecting such.	TCon_8	Medium	02/03/2018	Passed	PBurns							
TCase_9	Check that one property group consists of two property squares.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	When the Player lands on the Bot property square, the system shall indicate it is part of the Bars property group.	TCon_9 TCon_10	Medium	02/03/2018	Failed	PBurns	DEF_2	Closed	Med	21/02/2018	26/02/2018	When the player lands on a property square the system does not clearly indicate which property group that square belongs to.	High
TCase_10	Check that one property group consists of two property squares.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	When the Player lands on the Hatfield property square, the system shall indicate it is part of the Bars property group.	TCon_9 TCon_10	Medium	02/03/2018	Failed	PBurns	DEF_3	Closed	Med	21/02/2018	26/02/2018	When the player lands on a property square the system does not clearly indicate which property group that square belongs to.	High
TCase_11	Check that one property group consists of three property squares.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	When the Player lands on the PEC property square, the system shall indicate it is part of the QUB Premises property group.	TCon_9 TCon_11	Medium	02/03/2018	Failed	PBurns	DEF_4	Closed	Med	21/02/2018	26/02/2018	When the player lands on a property square the system does not clearly indicate which property group that square belongs to.	High
TCase_12	Check that one property group consists of three property squares.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	When the Player lands on the DKB property square, the system shall indicate it is part of the QUB Premises property group.	TCon_9 TCon_11	Medium	02/03/2018	Failed	PBurns	DEF_5	Closed	Med	21/02/2018	26/02/2018	When the player lands on a property square the system does not clearly indicate which property group that square belongs to.	High
TCase_13	Check that one property group consists of three property squares.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	When the Player lands on the CSB property square, the system shall indicate it is part of the QUB Premises property group.	TCon_9 TCon_11	Medium	02/03/2018	Failed	PBurns	DEF_6	Closed	Med	21/02/2018	28/02/2018	When the player lands on a property square the system does not clearly indicate which property group that square belongs to.	High
TCase_14	Check that there are two distinct property groups.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	When the Player lands on the Start/Go square ,the system shall not indicate that it is part of a property group.	TCon_10 TCon_11	Medium	02/03/2018	Passed	PBurns							
TCase_15	Check that there are two distinct property groups.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	When the Player lands on the Free Parking square, the system shall not indicate that it is part of a property group.	TCon_10 TCon_11	Medium	02/03/2018	Passed	PBurns							
TCase_16	Check that there are two distinct property groups.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	When the Player lands on the Pub Crawl square, the system shall not indicate that it is part of a property group.	TCon_10 TCon_11	Medium	02/03/2018	Passed	PBurns							
TCase_17	Check that the properties in the two-property group are more expensive than those in the three-property group.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	When a Player lands on the Bot or Hatfield squares (the Bar property group) the property prices will be less than those identified when the Player lands on the PEC, DKB and CSB squares (QUB Premises property group).	TCon_12	Medium	02/03/2018	Passed	PBurns							
TCase_18	Check that a player cannot develop a property square (i.e. add a house or similar depending on the game choices) that belongs to the two-property group when they don't own all property squares in that group.	In Eclipse, in the monoversity package and class, running the PlayGame class. Player owns none, and only one of the property squares at one time in the Bar property group.	N/A	The Player will not be offered the option to add a floor to one of the property squares in the Bar property group (the Bot or Hatfield squares) when they don't own any, or only own one of the aforementioned property squares at one time.	TCon_13	Medium	02/03/2018	Passed	PBurns							
TCase_19	Check that a player can develop a property square (i.e. add a house or similar depending on the game choices) that belongs to the two-property group when they own all property squares in that group.	In Eclipse, in the monoversity package and class, running the PlayGame class. Player owns both of the property squares in the Bar property group.	N/A	The player will be offered the option to add a floor to one of the property squares in the Bar property group (the Bot or Hatfield squares) when they own both of them.	TCon_13 TCon_19 TCon_20	Medium	02/03/2018	Passed	PBurns							
TCase_20	Check that a player cannot develop a property square (i.e. add a house or similar depending on the game choices) that belongs to the three-property group when they don't own all property squares in that group.	In Eclipse, in the monoversity package and class, running the PlayGame class. Player owns none, and only one of the property squares at one time in the QUB Premises property group.	N/A	The Player will not be offered the option to add a floor to one of the property squares in the Bar property group (the Bot or Hatfield squares) when they don't own any, or only own one of the aforementioned property squares at one time.	TCon_14	Medium	02/03/2018	Passed	PBurns							
TCase_21	Check that a player can develop a property square (i.e. add a house or similar depending on the game choices) that belongs to the three-property group when they own all property squares in that group.	In Eclipse, in the monoversity package and class, running the PlayGame class. Player owns all three of the property squares in the QUB Premises property group.	N/A	The player will be offered the option to add a floor to one of the property squares in the QUB Premises property group (the CSB, DKB or PEC squares) when they own all three.	TCon_14 TCon_19	Medium	02/03/2018	Passed	PBurns							
TCase_22	Check that the game has a Queen's University theme.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	The game interface and square names will have a Queen's University theme.	TCon_15	Medium	02/03/2018	Passed	PBurns							
TCase_23	Check that the a player cannot build a hotel (or similar depending on the game choices) on a property belonging to the two-property group when each square in that group does not have three houses (or similar) built on it.	In Eclipse, in the monoversity package and class, running the PlayGame class. Player owns both of the property squares in the Bar property group. None of the squares have three houses (or similar) built on them.	N/A	The Player will not be offered the option to add a hotel (or similar) to one of the property squares in the Bar property group (the Bot or Hatfield squares) when both property squares do not each have three houses (or similar) already developed on them.	TCon_16 TCon_19 TCon_20	Medium	02/03/2018	Passed	PBurns							

Test Cases

TCase_24	Check that the a player can build a hotel (or similar depending on the game choices) on a property belonging to the two-property group when each square in that group has three houses (or similar) built on it.	In Eclipse, in the monoversity package and class, running the PlayGame class. Player owns both of the property squares in the Bar property group. Both of the squares have three houses (or similar) built on them.	N/A	The Player will be offered the option to add a hotel (or similar) to one of the property squares in the Bar property group (the Bot or Hatfield squares) when both property squares have three houses (or similar) already developed on each of them.	TCon_16 TCon_19 TCon_20	Medium	02/03/2018	Failed	PBurns	DEF_7	Closed	Med	21/02/2018	28/02/2018	The player is not offered the option to add a hotel (or similar - extension in this case).	High
TCase_25	Check that the a player cannot build a hotel (or similar depending on the game choices) on a property belonging to the three-property group when each square in that group does not have three houses (or similar) built on it.	In Eclipse, in the monoversity package and class, running the PlayGame class. Player owns both of the property squares in the QUB Premises property group. None of the squares have three houses (or similar) built on them.	N/A	The Player will not be offered the option to add a hotel (or similar) to one of the property squares in the Bar property group (the DKB, PEC or CSB squares) when the three property squares do not each have three houses (or similar) already developed on them.	TCon_17	Medium	02/03/2018	Passed	PBurns							
TCase_26	Check that the a player can build a hotel (or similar depending on the game choices) on a property belonging to the three-property group when each square in that group has three houses (or similar) built on it.	In Eclipse, in the monoversity package and class, running the PlayGame class. Player owns both of the property squares in the QUB Premises property group. All of the squares have three houses (or similar) built on them.	N/A	The Player will be offered the option to add a hotel (or similar) to one of the property squares in the QUB Premises property group (the PEC, DKB or CSB squares) when the three property squares have three houses (or similar) already developed on each of them.	TCon_17 TCon_19 TCon_20	Medium	02/03/2018	Failed	PBurns	DEF_8	Closed	Med	21/02/2018	28/02/2018	The player is not offered the option to add a hotel (or similar - extension in this case).	High
TCase_27	Check that each player is told what square they have landed on.	In Eclipse, in the monoversity package and class, running the PlayGame class.	N/A	The system will display (print to screen) the name of the square they have landed on after rolling the dice.	TCon_18	Medium	02/03/2018	Passed	PBurns							
TCase_28	Check that each player is informed of any rent which is required to be paid when they land on a property square which is owned by the other player.	In Eclipse, in the monoversity package and class, running the PlayGame class. Player is on a property square owned by the other player.	N/A	The system will inform the player they are required to pay rent the amount of which determined by the level of development (if any) on that square.	TCon_19	Medium	02/03/2018	Passed	PBurns							
TCase_29	Check that each player is given the option to purchase a property square that they land on and which is not owned by any player.	In Eclipse, in the monoversity package and class, running the PlayGame class. Player is on a property square not owned by any player.	N/A	The system will ask the player if they wish to purchase the square, and request user input to confirm such.	TCon_19 TCon_20	Medium	02/03/2018	Passed	PBurns							
TCase_30	Check that each player informed that they are required to pay a 'fine' when they land on the 'Pub Crawl' square.	In Eclipse, in the monoversity package and class, running the PlayGame class. Player is on the Pub Crawl square.	N/A	The system will inform the player they are required to pay a fine of a specific amount.	TCon_19	Medium	02/03/2018	Passed	PBurns							
TCase_31	Check that the players current bank balance is displayed following an increment due to passing Start/Go.	In Eclipse, in the monoversity package and class, running the PlayGame class. Player has passed the Go square during their turn	N/A	The system will increment the player's balance with £200 and this will be displayed on screen, with the reason for the increment also displayed.	TCon_21 TCon_22	Medium	02/03/2018	Passed	PBurns							
TCase_32	Check that the players current bank balance is displayed following an increment due to the other player paying rent.	In Eclipse, in the monoversity package and class, running the PlayGame class. The other player has landed on a property square owned by the initial player.	N/A	The system will increment the player's balance with the calculated rent (based on the level of development) and this will be displayed on screen, along with the reason for the increment.	TCon_21 TCon_22	Medium	02/03/2018	Passed	PBurns							
TCase_33	Check that the players current bank balance is displayed following a decrement due to the player having to pay rent.	In Eclipse, in the monoversity package and class, running the PlayGame class. The player has landed on a property square owned by the other player.	N/A	The system will decrement the player's balance with the calculated rent (based on the level of development) and this will be displayed on screen, along with the reason for the decrement.	TCon_21 TCon_22	Medium	02/03/2018	Passed	PBurns							
TCase_34	Check that the players current bank balance is displayed following a decrement due to the player electing to purchase a property square.	In Eclipse, in the monoversity package and class, running the PlayGame class. The player has landed on a property square not owned by any other player and the player has opted to purchase it.	N/A	The system will decrement the player's balance with the purchase cost and this will be displayed on screen, along with the reason for the decrement.	TCon_21 TCon_22	Medium	02/03/2018	Passed	PBurns							
TCase_35	Check that the players current bank balance is displayed following a decrement due to the player electing to develop a floor on a square.	In Eclipse, in the monoversity package and class, running the PlayGame class. The player owns all the property squares in a property group and has opted to develop a floor on it.	N/A	The system will decrement the player's balance with the development cost and this will be displayed on screen, along with the reason for the decrement.	TCon_21 TCon_22	Medium	02/03/2018	Passed	PBurns							
TCase_36	Check that the players current bank balance is displayed following a decrement due to the player electing to develop an extension on a square.	In Eclipse, in the monoversity package and class, running the PlayGame class. The player owns all the property squares in a property group and has opted to develop an extension on it.	N/A	The system will decrement the player's balance with the development cost and this will be displayed on screen, along with the reason for the decrement.	TCon_21 TCon_22	Medium	02/03/2018	Passed	PBurns							
TCase_37	Check that once one player is declared bankrupt, the other player is declared the winner.	In Eclipse, in the monoversity package and class, running the PlayGame class. One player does not have sufficient funds to pay a fine or rent.	N/A	The system will inform the player they are bankrupt and declare the other player the winner.	TCon_23	Medium	02/03/2018	Passed	PBurns							
TCase_38	Check that players are given the opportunity to terminate the game if desired.	In Eclipse, in the monoversity package and class, running the PlayGame class.		The system provides the players the option to terminate the game.	TCon_24	Medium	02/03/2018	Passed	PBurns							
TCase_39	Check that, in the event a player elects to terminate the game, the other player is declared the winner.	In Eclipse, in the monoversity package and class, running the PlayGame class. One player has opted to terminate the game.	N/A	The system will inform the player they have terminated the game and declare the other player the winner.	TCon_25	Medium	02/03/2018	Passed	PBurns							
TCase_40	Check that, upon game termination due to bankruptcy, the bank balance for both players is displayed.	In Eclipse, in the monoversity package and class, running the PlayGame class. One player does not have sufficient funds to pay a fine or rent.	N/A	The system will display the current bank balance for Player One and Player Two.	TCon_26	Medium	02/03/2018	Passed	PBurns							
TCase_41	Check that, if a player elects to terminate the game, the bank balance for both players is displayed.	In Eclipse, in the monoversity package and class, running the PlayGame class. One player has opted to terminate the game.	N/A	The system will display the current bank balance for Player One and Player Two.	TCon_27	Medium	02/03/2018	Passed	PBurns							

Test Procedures

Test Procedure ID	Test Case ID	Description	Comments
TProc_1	TCase_1 TCase_3	Open Eclipse, then the monoversity package and class, and run the PlayGame class. Visually check on-screen prompts are provided requesting the users to enter the Player's names. Visually note how many players names are requested. Check on-screen prompts are provided requesting the user roll a virtual dice	Manual test (visual inspection)
TProc_2	TCase_2	After following system prompts to enter player names, and on first roll of the dice for each player, note the bank balance for each before any board moves/actions have occurred. Compare the balances to ensure they are the same.	Manual test (visual inspection)
TProc_3	TCase_4 TCase_5	Open Eclipse, then the monoversity package and class, and run the PlayGame class. Check on-screen prompts are provided requesting the user roll a virtual dice. Follow these prompts and note whether a random number is displayed on-screen for each occasion. Check the prompts are sequentially giving each of the two players a turn.	Manual test (visual inspection)
TProc_4	TCase_6 TCase_7 TCase_8 TCase_22 TCase_31	Open Eclipse, then the monoversity package and class, and run the PlayGame class. Follow the on-screen prompts to navigate around the board a number of times and note all the squares encountered, including whether a Start/Go square is encountered. Note whether a defined fee (£200) is incremented to the current player's bank balance both when they pass the Start/Go square and when they land on the Start/Go square. Take note of the overall theme of the game as you navigate around the board (specifically if it has an overall Queen's University theme).	Manual test (visual inspection)
TProc_5	TCase_9 TCase_10 TCase_11 TCase_12 TCase_13 TCase_14 TCase_15 TCase_16 TCase_17 TCase_27	Open Eclipse, then the monoversity package and class, and run the PlayGame class. Follow the on-screen prompts to navigate around the board a number of times and note all the squares encountered and the on-screen information displayed for each square. When landing on property squares, specifically note the cost of each and whether the system indicates that property squares (and any of the other squares) are part of a particular Property Group. Compare these notes to the game board to determine whether property squares are identified as belonging to a specific Property Group and the cost variance between the distinct Property Groups.	Manual test (visual inspection)
TProc_6	TCase_18 TCase_19 TCase_20 TCase_21 TCase_23 TCase_24 TCase_25 TCase_26 TCase_29 TCase_34 TCase_35 TCase_36	Open Eclipse, then the monoversity package and class, and run the PlayGame class. Follow the on-screen prompts to navigate around the board a number of times and puchase property squares when presented with the opportunity, noting under what circumstances the user is offered the opportunity to purchase the square (i.e the current state of ownership of the square). Develop the squares when possible, noting when the user is presented with the option to develop the property squares (and any other square) with floors (the game's equivalent of houses) and extensions (the game's equivalent of hotels). Note whether the user's bank balance is amended accordingly following the purchase of a property square, in addition to the development of a floor and the development of an extension on a property square.	Manual test (visual inspection)
TProc_7	TCase_28 TCase_30 TCase_32 TCase_33	Open Eclipse, then the monoversity package and class, and run the PlayGame class. Follow the on-screen prompts to navigate around the board a number of times. Note whether the player's bank balance is automatically decremented when they land on any of the squares, specifically if it is decremented when expected (i.e. when the player lands on the Pub Crawl square or on a property that is owned by the other player and the payment of rent is required). Note whether the system informs the player how much is to be decremented, why the decrement was made and the bank balance following such. In the instance a decrement is made to pay rent, note whether the bank balance of the player receiving the rent is accordingly amended and displayed.	Manual test (visual inspection)
TProc_8	TCase_37 TCase_40	Open Eclipse, then the monoversity package and class, and run the PlayGame class. Follow the on-screen prompts to play the game until one player does not have the required funds to pay a fine and pay a required rent amount (i.e. bankruptcy). Pay attention to the on-screen information to determine whether the correct player (specifically the other player) has been declared the winner and whether the current bank balance for the two players has been displayed.	Manual test (visual inspection)
TProc_9	TCase_38 TCase_39 TCase_41	Open Eclipse, then the monoversity package and class, and run the PlayGame class. Follow the on-screen prompts and opt to terminate the game if the option is provided. Pay attention to the on-screen information to determine whether the correct player (specifically the other player) has been declared the winner and whether the current bank balance for the two players has been displayed.	Manual test (visual inspection)