QUEEN'S UNIVERSITY BELFAST

Semester 1 Report

Group 59

Table of Contents

Peer Assessment

Evaluation

Evaluation	Group Number: 59			
Name	Contribution to team-working and motivation[1]	Contribution to PDF Report 1 [1,2]	Contribution to Interim Demo[1,2]	**Peer Score** (Range 85 – 115)
Dean Logan	4	5	3	114
Conor Nugent	3	4	3	108
John Higgins	4	3	5	110
Scott McDonald	3	3	3	105

[1]Values for contribution: 1 = Minimal Contribution; 2 = Reasonable Contribution; 3 = Good Contribution; 4 = Very Good Contribution;
5 = Excellent Contribution

Declaration

Declaration

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

Name	Date	Confirmation *(use the words shown in the example below!)*
Dean Logan	02/12/2021	I agree to the terms of the declaration
Conor Nugent	02/12/2021	I agree to the terms of the declaration
John Higgins	02/12/2021	I agree to the terms of the declaration
Scott McDonald	02/12/2021	I agree to the terms of the declaration

Personal Statements

Personal statement of	*Dean Logan*
The following were my most significant contributions to the Semester 1 Deliverable (100 words or less):	
I was responsible for the creation of some of the use case descriptions and helping the team create the use case diagram, class diagram, board layout and forming the pdf report. I also created the gantt chart and all the use case realisations. All of work I did for the deliverable has been labelled with my initials within the final PDF document.	

Personal statement	*Conor Nugent*
The following were my most significant contributions to the Semester 1 Deliverable (100 words or less):	
I was responsible for the creation of some use case descriptions, some use cases in the use case diagram, some classes in the UML class diagram and designed the game board including naming the board locations. All of work I did for the deliverable has been labelled with my initials within the final PDF document.	

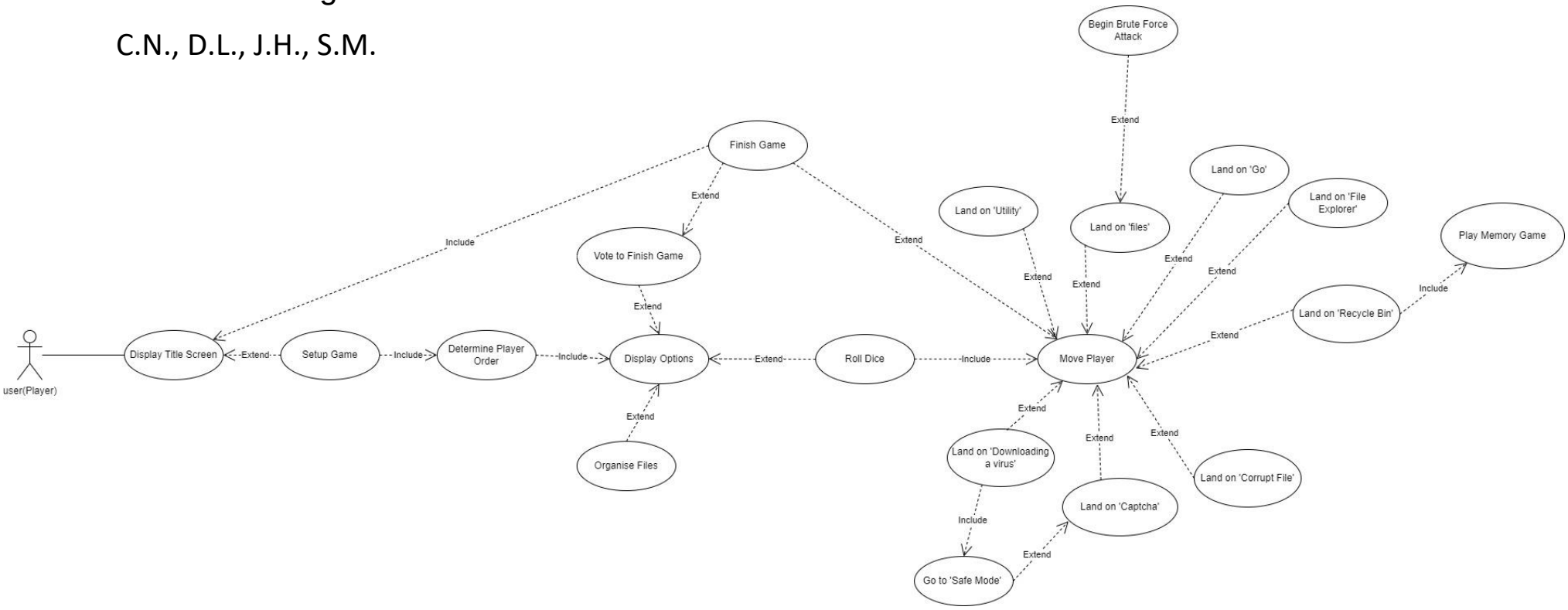
Personal statement of	*John Higgins*
The following were my most significant contributions to the Semester 1 Deliverable (100 words or less):	
Like all members of the team, I was involved in the discussions surrounding, and creation of the use case diagram and the class diagram. I was also solely responsible for creating some of the use case descriptions as noted in the PDF document by my initials. I was also the lead developer on the demo programme and recorded and edited the video presentation based on it.	

Personal statement of	*Scott McDonald*
The following were my most significant contributions to the Semester 1 Deliverable (100 words or less):	
I wrote the weekly minutes, I created several of the use case descriptions relating to the options and edited many more, I aided the design of the board layout, I aided with the creation of the use case diagram, I concocted the idea behind the file explorer function, I created some of the class diagrams and aided with the use case realisations. I created several of the ideas behind how the user would interact with the files and other tiles on the board including but not limited to, buying a property, the brute force attack, the recycle bin, and more.	

Use Case Requirements Specification and Planning

Use Case Diagram

C.N., D.L., J.H., S.M.



Use Case Description

Flow of Events for the Land on 'Downloading a Virus' use-case – D.L	
Objective	**To punish the player for landing on this location. It will put the player into "Safe Mode" for a certain amount of go's depending on the value of "severity of virus" which is incremented every time a user lands on this location.**
Precondition	*The player has rolled the dice to land on this tile*
Main Flow	1. Increment the severity of the virus. 2. Get the severity of the virus and trigger the 'Go To Safe Mode' use case. The length the user is in 'Safe Mode' depends on the severity of the virus.
Alternative Flows	N/A
Post-Condition	*The player can now take their turn*

Flow of Events for Go To 'Safe Mode' – D.L	
Objective	**Send the player from whatever tile they are currently on to the 'Safe Mode' tile as a punishment**
Precondition	*The player has landed on the 'Downloading a virus' tile*
Main Flow	1. Output text telling the user that they are being moved to the 'Safe Mode' file. 2. The player rolls a double to get out of 'Safe Mode'
Alternative Flows	At 2, if the player does not roll a double within 3 goes, they must use 50 bits and move forward by the number of spaces rolled on the dice At 2, the player decides to pay out
Post-Condition	*The next player can take their turn*

Flow of Events for Land on 'Captcha'	
Objective	**To give the player a reward or punishment depending on their performance**
Precondition	*The player has rolled the dice to land on this tile*

Main Flow	1. A random mini game is selected 2. The player completes the mini game successfully 3. The player then receives a certain number of bits depending on the mini game
Alternative Flows	At 2, the player does not complete the mini game successfully 3. Then the “Go To Safe Mode” use case is triggered
Post-Condition	*The player can now take their turn*

Flow of Events for Land on ‘Corrupt File’ – D.L	
Objective	**Used to punish the player for landing on this location. It will take a random file from the user as it has become “corrupt” and then add this file to the recycling bin. If the player has no files they will lose bits**
Precondition	*The player has rolled the dice to land on this location*
Main Flow	1. The system will check if the player owns any files 2. If the player owns files then a random file will be selected and moved to the recycling bin.
Alternative Flows	At 2, If the user does not have any files then nothing will be taken off the player
Post-Condition	*The next player now can take their turn*

Flow of Events for Land on ‘File’ – D.L	
Objective	**These will act similarly to properties in monopoly**
Precondition	*The player has landed on one of the ‘file’ tiles*
Main Flow	1. A check is completed to see if a player owns a file or not 2. The player has decided to if they want to download the file 3. The player will then “use up” the determined amount of their storage space to download the file
Alternative Flows	At 2, Another player is already downloading the file 3. The current player will then begin to play the “Brute Force Attack” mini game At 2, The current player does not own the file and decides not to download it
Post-Condition	*The next player can take their turn*

Flow of Events for Brute Force Attack Mini Game – D.L	
Objective	**Gives the player a chance to not have to give up bits by answering a quiz question**
Precondition	*A player has landed on a file which another player owns*
Main Flow	1. The player will be presented with a random question 2. If the player answers the question correctly they will not have to give the other player any bits
Alternative Flows	At 2, The player does not answer the question correctly and is has to give the other player bits depending on the file
Post-Condition	*The next player can take their turn*

Flow of Events for the Land on 'Go' use-case – D.L	
Objective	**This is the starting location for the game, the user will get 200 bits whenever they pass go**
Precondition	*The game has started*
Main Flow	1. The user has passed over the 'Go' location 2. The user will receive 200 bits
Alternative Flows	N/A
Post-Condition	*The next player can now take their turn*

Flow of Events for the Move Player use-case – D.L	
Objective	**This will be responsible for moving the player around the board after a dice roll**
Precondition	*The player has rolled their dice*
Main Flow	1. Check the players bits to see if they are out of the game 2. The number generated by roll dice will be added onto the players current position 3. Then the use case for the corresponding location which they have landed on will trigger. 4. Then current player will be updated to the next player
Alternative Flows	At 2, The current player has no bits so they are out of the game 3. If only one player has bits left this will trigger the finish game use case

Post-Condition	*The next player can now take their turn*

Flow of Events for the Land On Utility use-case - C.N	
Objective	**To land on a 'Utility'**
Precondition	*The player rolls the dice to land on the 'Utility'*
Main Flow	1. The player lands on the 'Utility'. 2. The player gets a pass to get a full amount of go bits when using search engines. 3. The next player then has their turn.
Alternative Flows	At 2, the player may already have a pass using the search engines. Therefore, they get a second pass once landed on the 'Utility'.
Post-condition	*The player can use the pass to move across the board using the 'Search Engine' and get the full amount of bits passing go.*

Flow of Events for the Land On File Explorer use-case - C.N	
Objective	**To land on a 'File Explorer'**
Precondition	*The player rolls the dice to land on the 'File Explorer'*
Main Flow	1. The player lands on the 'File Explorer'. 2. The 'files' that the current player owns are checked. 3. The player can move to any 'File' they own. 4. The player pays a fare out of their bits they get when passing go. 5. If they pass go, they can use the 'Utility' pass to get the full amount of bits. 6. The next player then has their turn.
Alternative Flows	At 3, the player may choose not to jump files to but allow the next player to move. At 3, the player may not own any files. At 4, the player may not have a 'Utility' pass and will not get the full amount passing go.
Post-condition	*When the current player passes go, they either get the full amount of bits or not.*

Flow of Events for the Display Title Screen use-case - C.N	
Objective	**To display the 'Title' screen.**
Precondition	*The user runs the program to display the 'Title' screen.*
Main Flow	1. The 'Title' screen is displayed. 2. The user selects the 'Setup Game' option.
Alternative Flows	At 2, the user selects the 'Quit Game' option which kills the program.
Post-condition	*The user is directed to the 'Setup Game' screen.*

Flow of Events for the Set-Up Game use-case - C.N	
Objective	**To set up the game on the 'Setup Game' screen.**
Precondition	*The 'Setup Game' option is selected on the 'Title' screen.*
Main Flow	1. The 'Setup Game' screen is displayed. 2. The player names are entered by the user.
Alternative Flows	At 2, the user decides to direct back to the 'Title' screen instead of setting up the game.
Post-condition	*The user has finished setting up the game and moves on to determining the player order.*

Flow of Events for the Finish Game use-case - C.N	
Objective	**To finish the game.**
Precondition	*All the players, apart from one, have lost all of their bits.*
Main Flow	1. The game is finished. 2. The final game results are displayed.
Alternative Flows	N/A
Post-condition	*The game title screen is displayed.*

Flow of events for the 'Roll Dice' Use Case - J.H.	
Objective	**To Obtain A Randomly Generated Number of Places for the Player to Move.**
Pre-condition	*The Players Have Been Rotated*
Main Flow	1. The Player tells the interface to 'roll the dice'. 2. Two dice objects obtain random numbers between 1 and 6. 3. These numbers are added together.
Alternative Flows	At 1, the player may have been required to miss a turn, meaning they cannot roll the dice and the players should rotate again.
Post-Condition	*Control is passed to the Move Player use case.*

Flow of events for the 'Land on Recycle Bin' Use Case - J.H.	
Objective	**To check for 'files' previously marked as being 'corrupt'.**
Pre-condition	*The player's position has been moved and they have landed on 'Recycle Bin'*
Main Flow	1. The system checks for any 'files' that have been previously marked as 'corrupt' and loads them into an array that is **not displayed to the player.** 2. The 'Memory Game' use case is executed.
Alternative Flows	At 2, the system may not have found any files marked as corrupt, in which case the players rotate and the game returns to the 'Display Menu' use case.
Post-Condition	*The 'Memory Game' use case executes.*

Flow of events for the 'Memory Game' Use Case - J.H.	
Objective	To allow the player to play a 'Memory Game' (framed as a 'Search Recycle Bin Function) that will allow them to win some of the 'corrupted files' based on how many they can remember.
Pre-condition	*The game has checked for corrupt files and has called the memory game use case.*

Main Flow	1. The corrupt file array is obtained. 2. The player is asked to enter the names of any files that they can remember have been marked as 'corrupt' throughout the game. 3. This input request will execute until a termination case is entered. 4. Every input is checked against the array of files. 5. If an input matches a file in an array the file is removed from the array, the 'recycling bin' and assigned to the player. 6. When the termination case is entered, control is returned to 'Roll Dice'.
Alternative Flows	At 1, if there is no match in the array for the player's input, a message is displayed to the console stating no match was found. The use case will continue to execute until the termination case is entered.
Post-Condition	*The current player has gained all files that they were able to remember. Other files remain in the 'Recycling Bin'.*

Flow of events for the 'Determine Player Order' Use Case - J.H.	
Objective	**To determine an order for the players to take their turns in.**
Pre-condition	*The number of players and their names have already been entered. A terminating case (e.g., '-1') was entered while accepting inputs for players.*
Main Flow	1. The system will iterate through the players. 2. For each player the dice is rolled. 3. After every player has rolled the dice, the players are re-ordered based on who rolled the highest value. 4. The resulting order is displayed to the console. 5. Display Options' is called.
Alternative Flows	At 1, if only a single player was declared beforehand, a message is displayed to the console stating that more than one player is required. Control is returned to the 'Setup Game' use case.
Post-Condition	*The players have been placed into an order that will be referred to throughout the game as this is the order in which they will take their turns.*

Flow of Events for the Display Menu Use Case - S.M.	
Objective	**Present the user with a list of options relevant to playing the game**
Precondition	*The game has started, and the player order has been determined*
Main Flow	1. User enters in an option that corresponds with one of the options on the list. 2. The “Roll Dice” option is then selected and the two dice are “rolled”.
Alternative Flows	At 1, the user may select to display the game rules. In this case the game instead displays a new list clarifying the rules of the game. At 1, the user may select to display the board layout. In this case the game instead displays a new list describing the board and listing the names, prices, if there is a player present and how far away the property is. At 1, the user may select to start a vote to return to the menu. At 1, the user may select to organise their files. At 1, the system may run a check and find that the user
Post-Condition	*The player takes their turn*

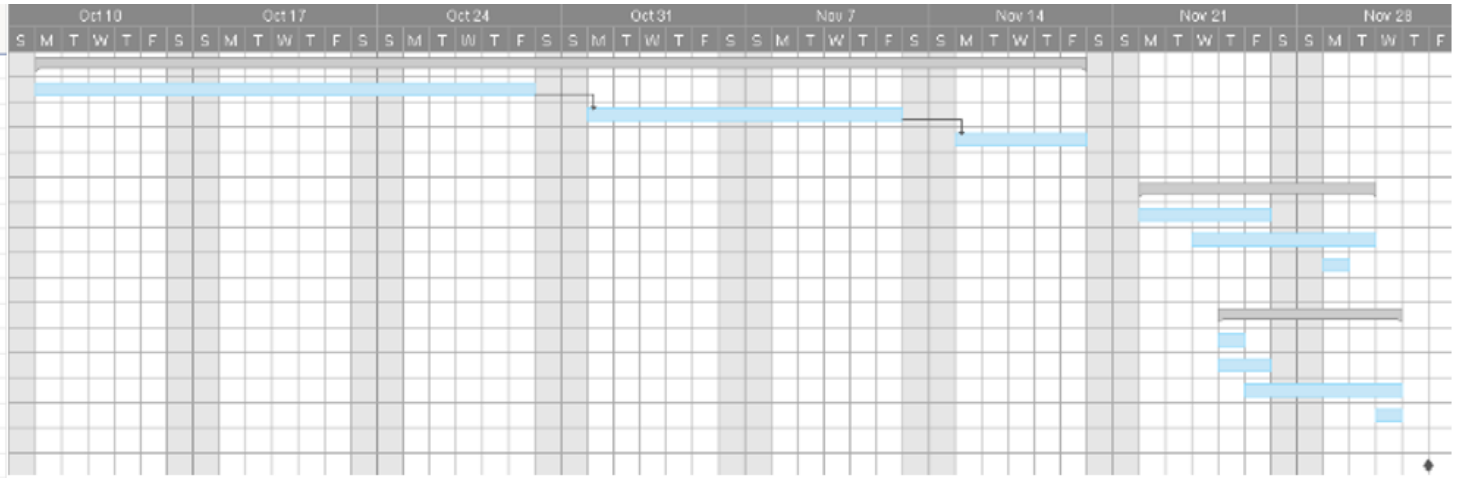
Flow of Events for the Organise File Use Case - S.M.	
Objective	**Present the user with a list of all viable files that can be organised.**
Pre-Condition	*The user has selected this option at the start of their turn*
Main Flow	1. User enters in a number that corresponds with one of the options on the list 2. The corresponding file is then selected 3. The user begins setup 4. The user then returns to the file list 5. The user returns to the option list
Alternative Flows	At 3, the user may attempt to begin installation if they have already setup 3 times before this. At 4, the user may choose to continue with setups At 5, the user may choose to begin install. At 5, the user may choose to setup again.
Post-Condition	*The user files are organised and now have a higher penalty for players that land on them.*

Flow of Events for the Vote to Finish Game Use Case - S.M.	
Objective	**Present the users with two options. Quit or Keep Playing.**
Precondition	*The user has selected to start a vote to return to the main menu*
Main Flow	1. A user is asked if they wish to keep playing the game 2. Their vote is taken and stored 3. The next player in order is asked if they wish to keep playing 4. A majority vote is reached in favour of quitting. 5. The game ends.
Alternative Flows	At 1, there may already be a majority vote in favour of leaving the game and so no vote will be taken, and the game will be ended. At 1, there may already be a majority vote in favour of continuing the game and so no vote will be taken by the player and the game will continue. At 2, a majority vote may have been reached with their vote and so the next player will not be asked. At 4, a majority vote may not have been reached (for example a stalemate). In this case a vote is taken again and again until a majority is reached. At 4, a majority vote may be in favour of continuing the game and so the game does not end, and the option list is displayed again.
Post-Condition	*Their results are displayed and then they return to the main menu.*

Gantt Charts

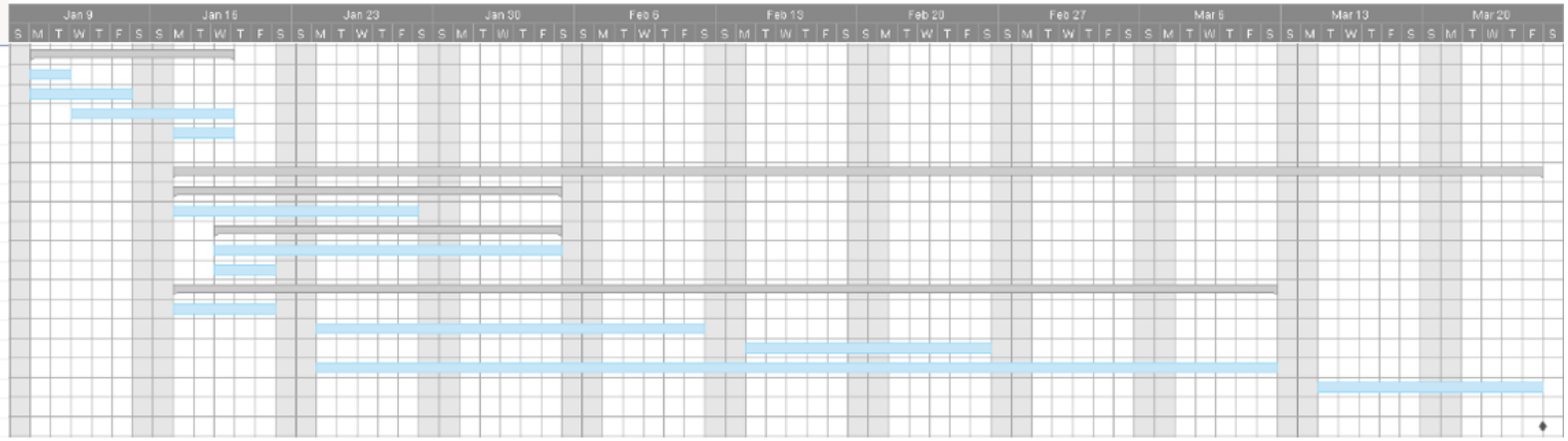
Semester 1 D.L

Task Name	Duration	Start	Finish
Use Case Requirements Specification and Planning	30d	10/11/21	11/19/21
Get Project Ideas and Plan Game	15d	10/11/21	10/29/21
Use Case Diagram	10d	11/01/21	11/12/21
Use Case Descriptions	5d	11/15/21	11/19/21
System Analysis	7d	11/22/21	11/30/21
Class Diagram	5d	11/22/21	11/26/21
Use Case Realisations	5d	11/24/21	11/30/21
Draft Game Layout	1d	11/29/21	11/29/21
Interim Demo	5d	11/25/21	12/01/21
Create Plan for the Video	1d	11/25/21	11/25/21
Write a Script	2d	11/25/21	11/26/21
Code a Demo	4d	11/26/21	12/01/21
Film Video	1d	12/01/21	12/01/21
Submission of PDF report and Demo	0	12/03/21	12/03/21



Semester 2 D.L

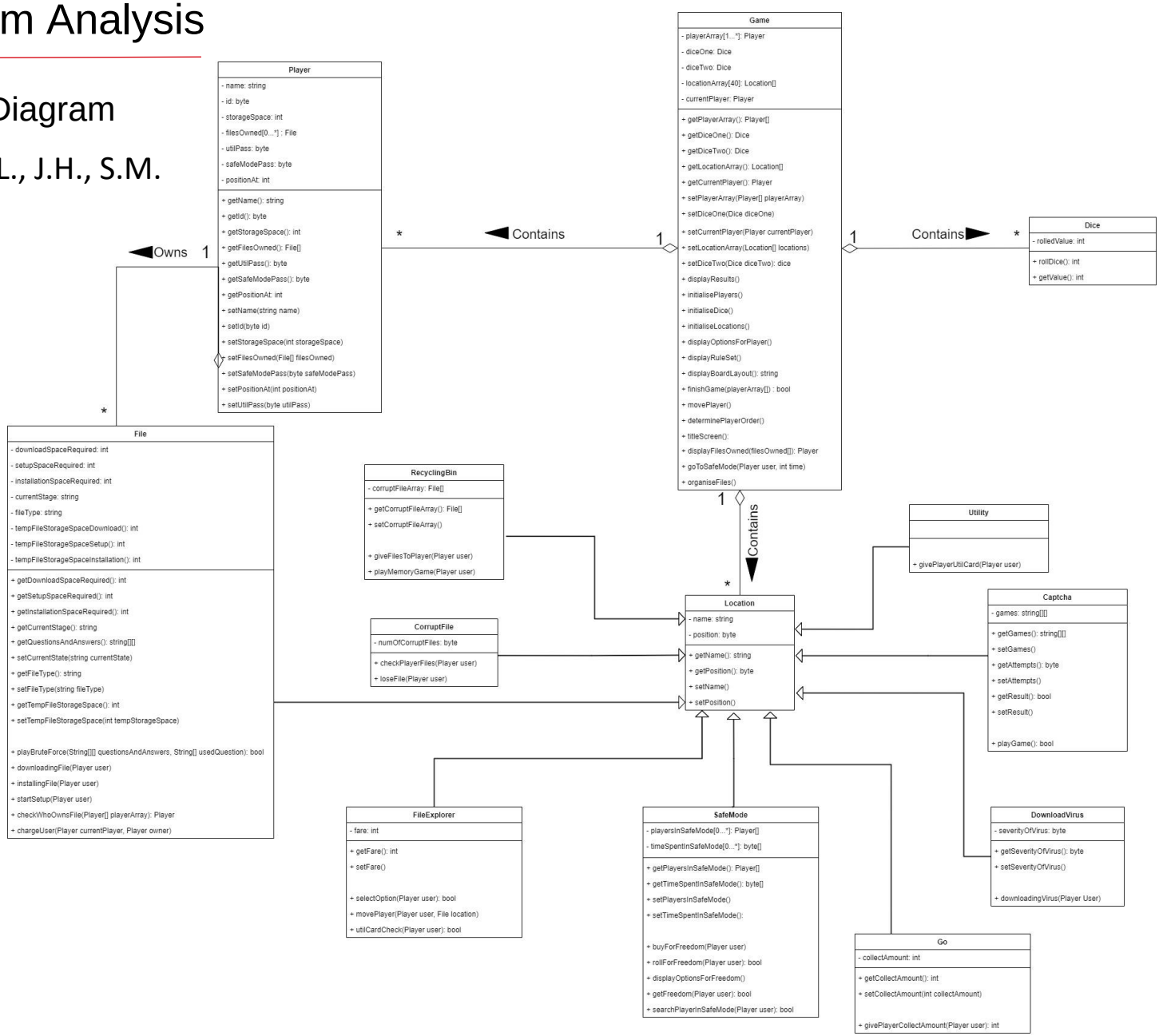
Task Name	Duration	Start	Finish
Design Documentation	8d	01/10/22	01/19/22
Text User Interface Design	2d	01/10/22	01/11/22
Class Relationship Model	5d	01/10/22	01/14/22
Sequence Diagram	6d	01/12/22	01/19/22
Final Game Layout	3d	01/17/22	01/19/22
Development of Solution	49d	01/17/22	03/24/22
Code Development	15d	01/17/22	02/04/22
Create The Console	10d	01/17/22	01/28/22
Create Classes	13d	01/19/22	02/04/22
Define Methods	13d	01/19/22	02/04/22
Define Fields	3d	01/19/22	01/21/22
Testing	40d	01/17/22	03/11/22
Create Test Plan	5d	01/17/22	01/21/22
Unit Test Classes	15d	01/24/22	02/11/22
Integration Testing	10d	02/14/22	02/25/22
Bug Fixing	35d	01/24/22	03/11/22
Code Review	9d	03/14/22	03/24/22
Project End	0	03/25/22	03/25/22



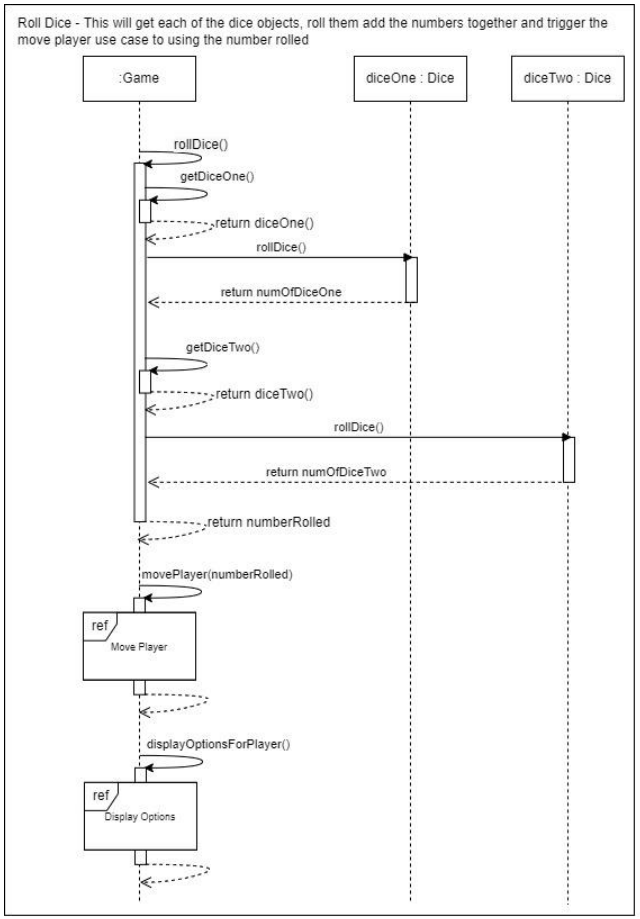
System Analysis

Class Diagram

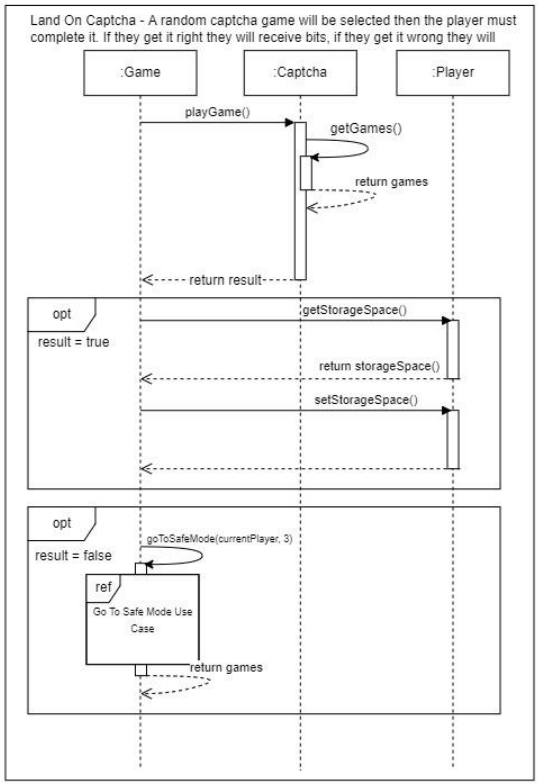
C.N., D.L., J.H., S.M.



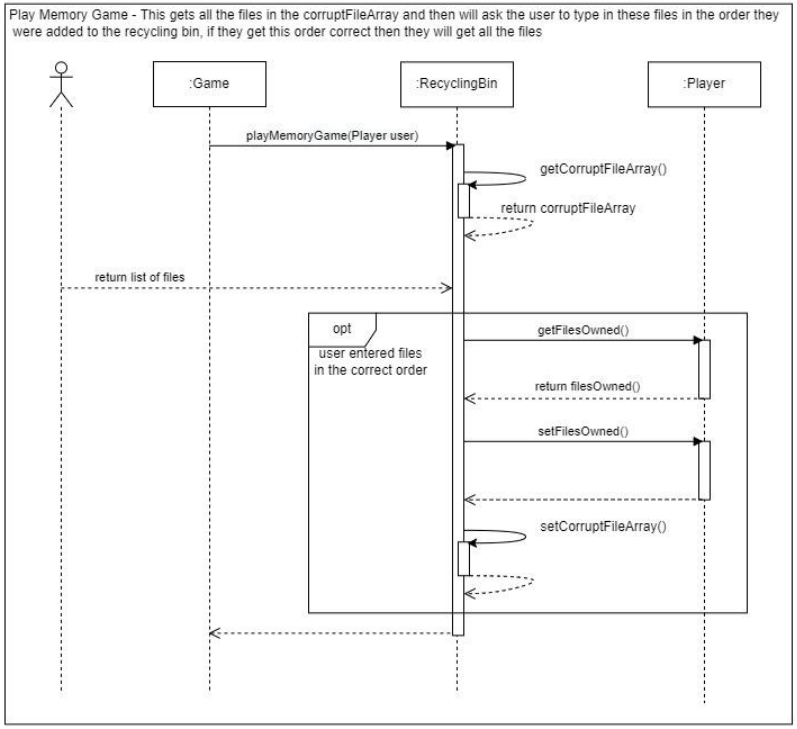
Use Case Realisation



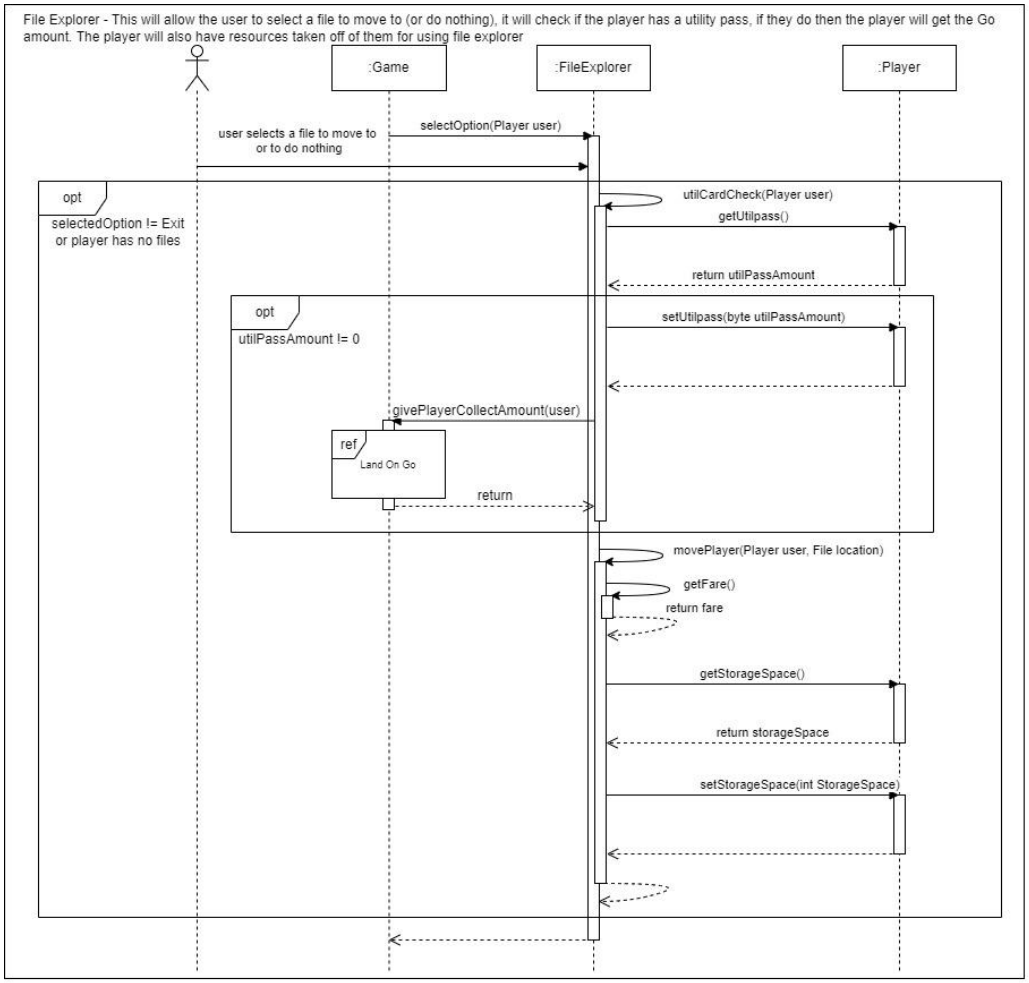
D.L, S.M



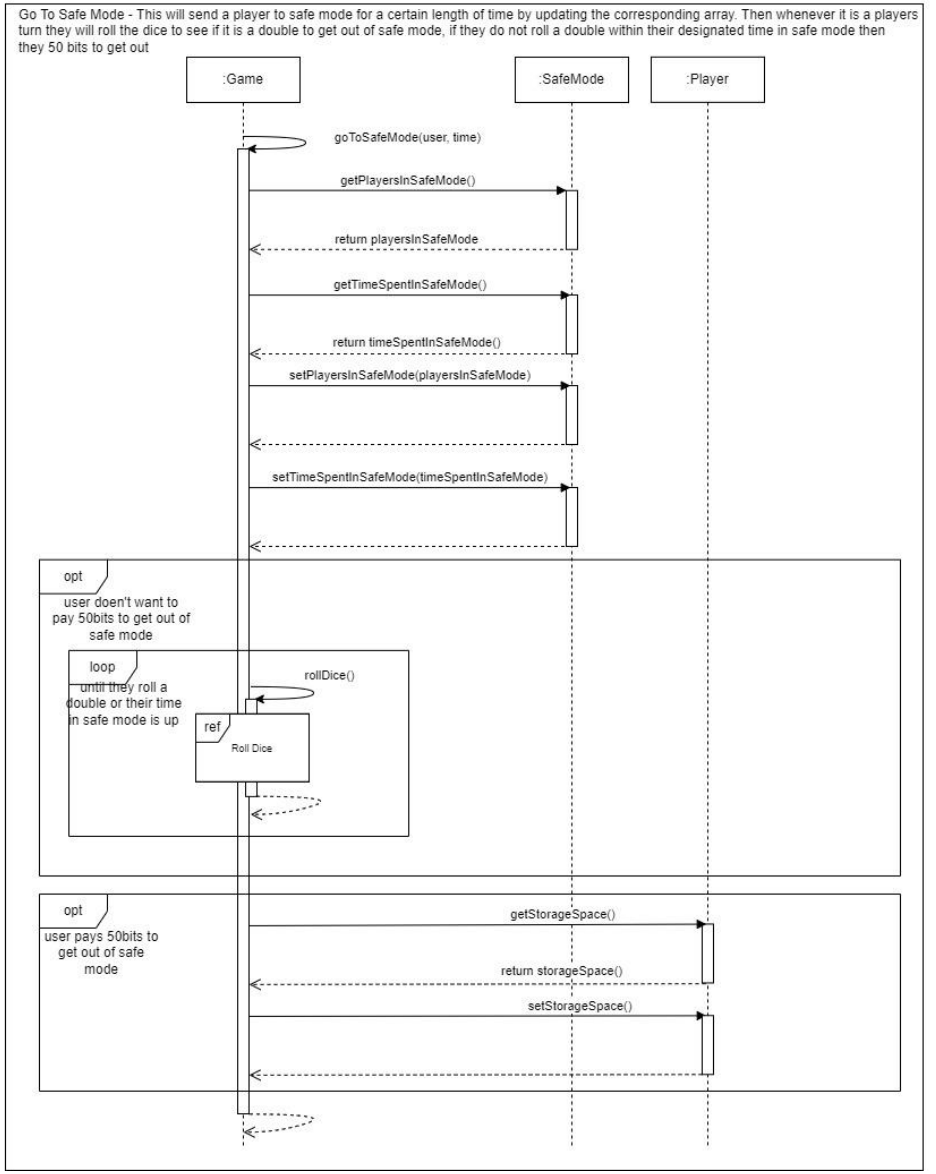
D.L



D.L, J.B

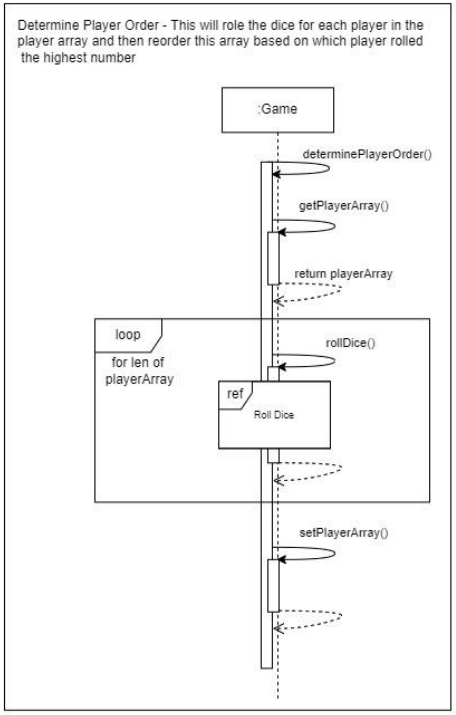

File Explorer - This will allow the user to select a file to move to (or do nothing), it will check if the player has a utility pass, if they do then the player will get the Go amount. The player will also have resources taken off of them for using file explorer
:Game
:FileExplorer
:Player
user selects a file to move to or to do nothing
selectOption(Player user)
opt
selectedOption != Exit or player has no files
utilCardCheck(Player user)
getUtilpass()
return utilPassAmount
opt
utilPassAmount != 0
setUtilpass(byte utilPassAmount)
givePlayerCollectAmount(user)
ref
Land On Go
return
movePlayer(Player user, File location)
getFare()
return fare
getStorageSpace()
return storageSpace
setStorageSpace(int StorageSpace)

D.L, C.N

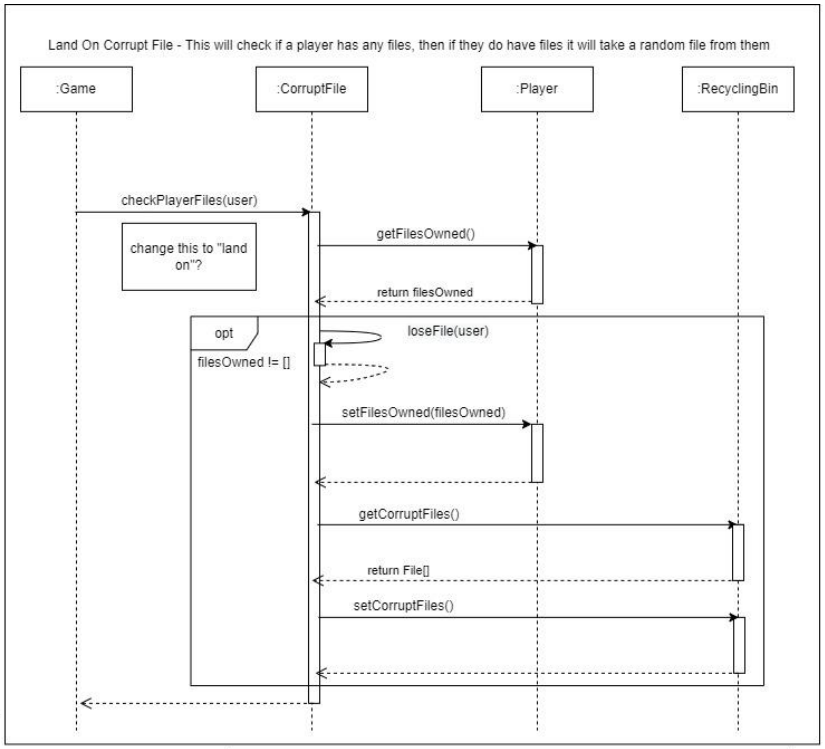

Go To Safe Mode - This will send a player to safe mode for a certain length of time by updating the corresponding array. Then whenever it is a players turn they will roll the dice to see if it is a double to get out of safe mode, if they do not roll a double within their designated time in safe mode then they 50 bits to get out
:Game
:SafeMode
:Player
goToSafeMode(user, time)
getPlayersInSafeMode()
return playersInSafeMode
getTimeSpentInSafeMode()
return timeSpentInSafeMode()
setPlayersInSafeMode(playersInSafeMode)
setTimeSpentInSafeMode(timeSpentInSafeMode)
opt
user doen't want to pay 50bits to get out of safe mode
loop
until they roll a double or their time in safe mode is up
rollDice()
ref
Roll Dice
opt
user pays 50bits to get out of safe mode
getStorageSpace()
return storageSpace()
setStorageSpace()

D.L

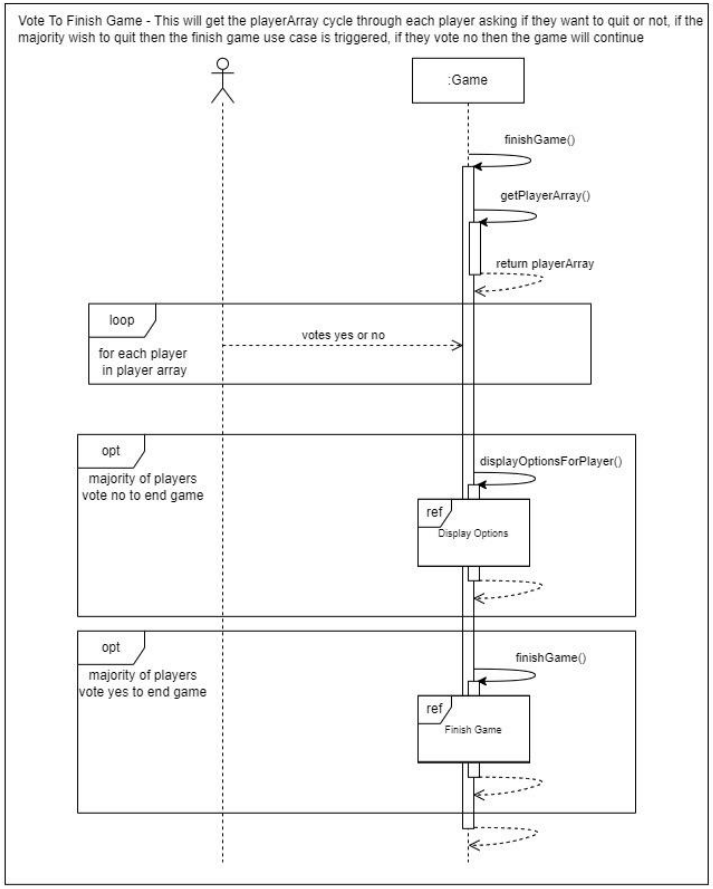
D.L, J.B

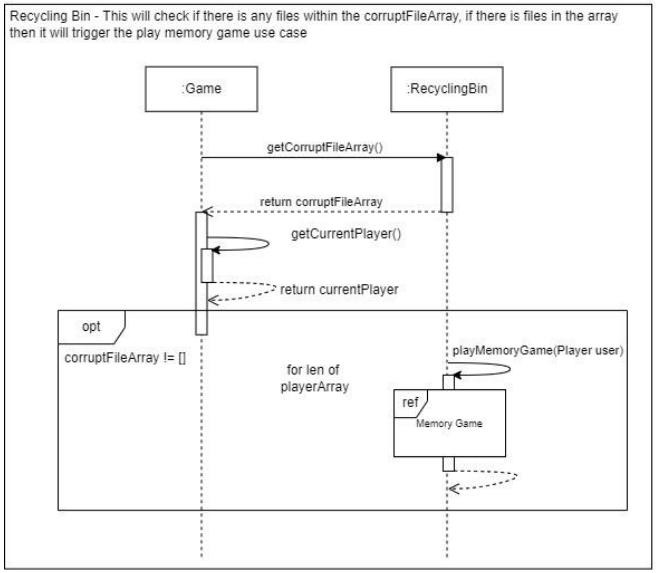


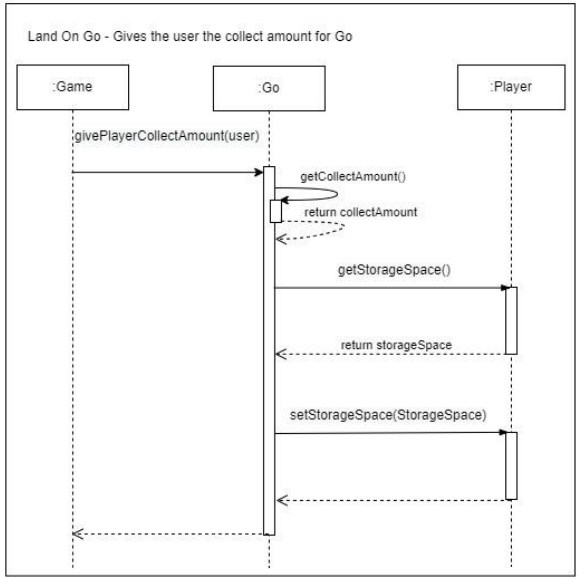
D.L



D.L

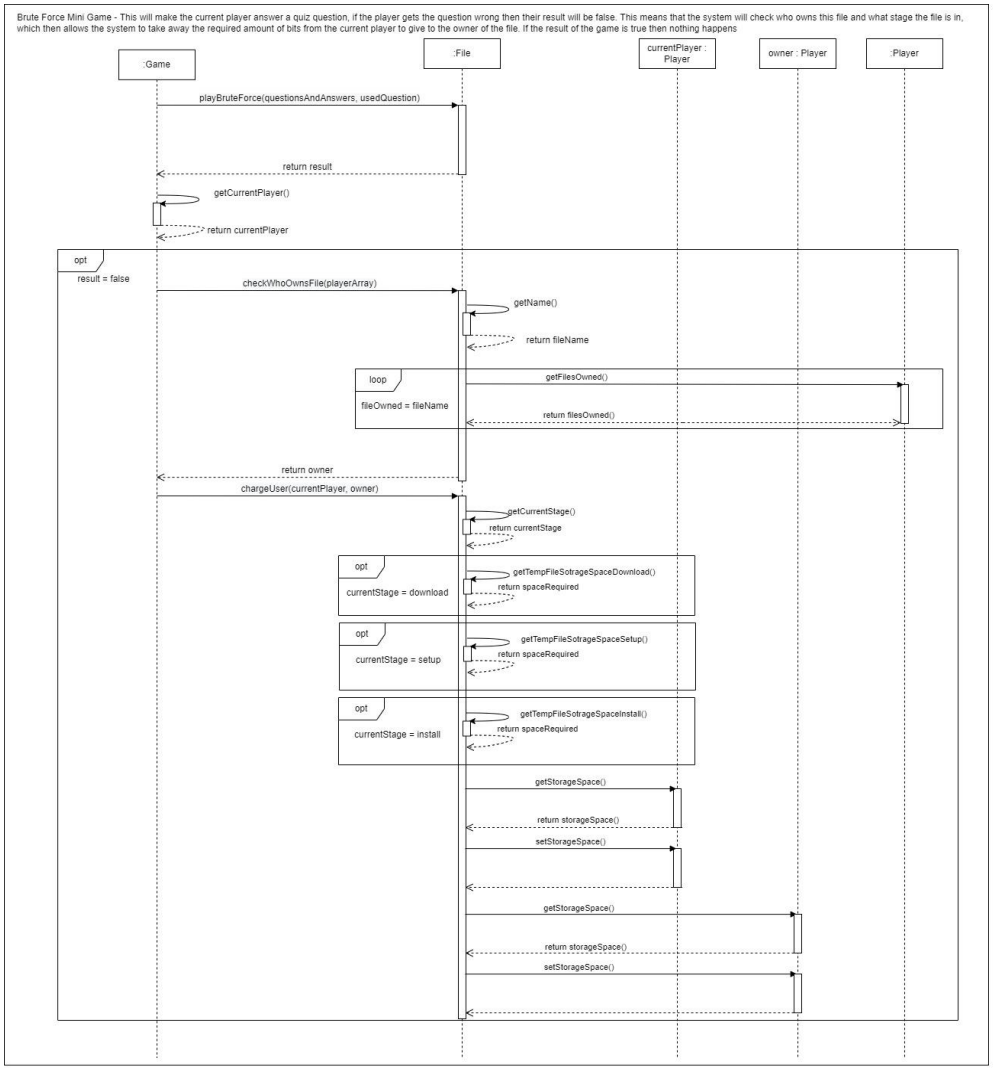




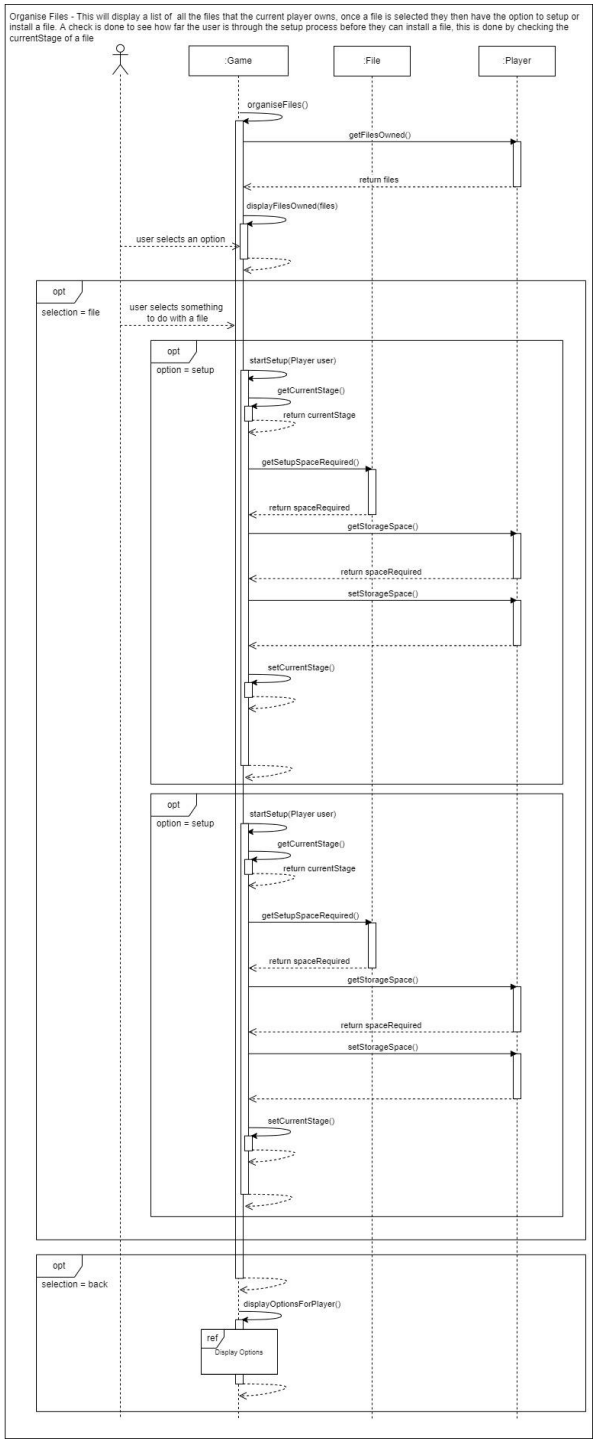


D.L, J.B

D.L

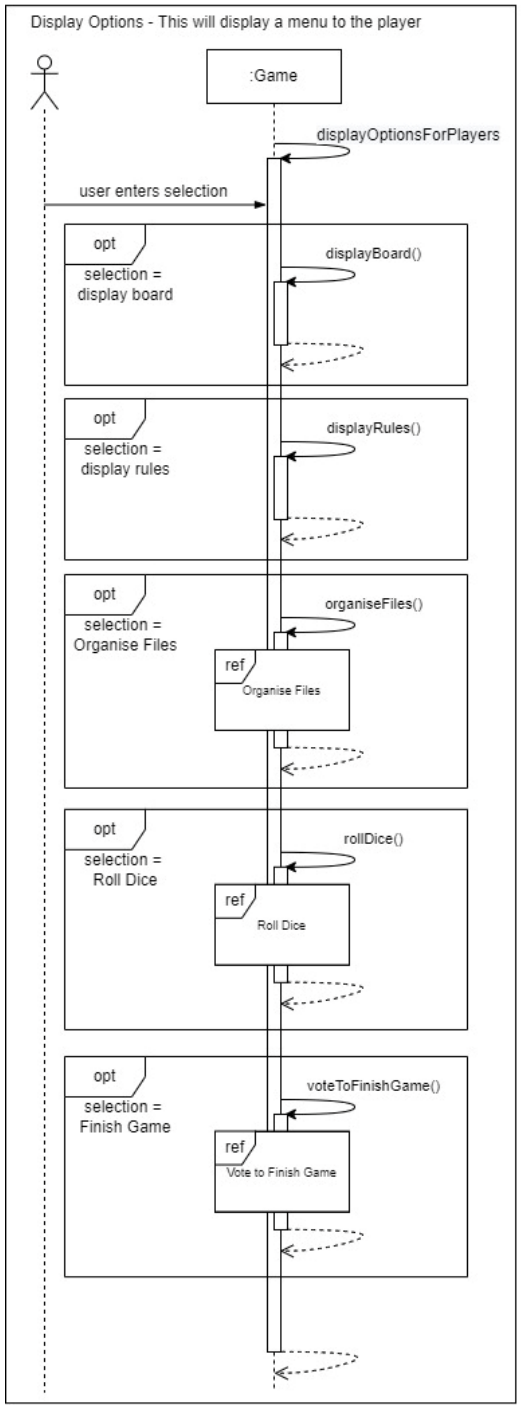

Brute Force Mini Game - This will make the current player answer a quiz question, if the player gets the question wrong then their result will be false. This means that the system will check who owns this file and what stage the file is in, which then allows the system to take away the required amount of bits from the current player to give to the owner of the file. If the result of the game is true then nothing happens
:Game
:File
currentPlayer : Player
owner : Player
:Player
playBruteForce(questionsAndAnswers, usedQuestion)
return result
getCurrentPlayer()
return currentPlayer
opt
result = false
checkWhoOwnsFile(playerArray)
getName()
return fileName
loop
fileOwned = fileName
getFilesOwned()
return filesOwned()
return owner
chargeUser(currentPlayer, owner)
getCurrentStage()
return currentStage
opt
currentStage = download
getTempFileSotrageSpaceDownload()
return spaceRequired
opt
currentStage = setup
getTempFileSotrageSpaceSetup()
return spaceRequired
opt
currentStage = install
getTempFileSotrageSpaceInstall()
return spaceRequired
getStorageSpace()
return storageSpace()
setStorageSpace()
getStorageSpace()
return storageSpace()
setStorageSpace()

D.L

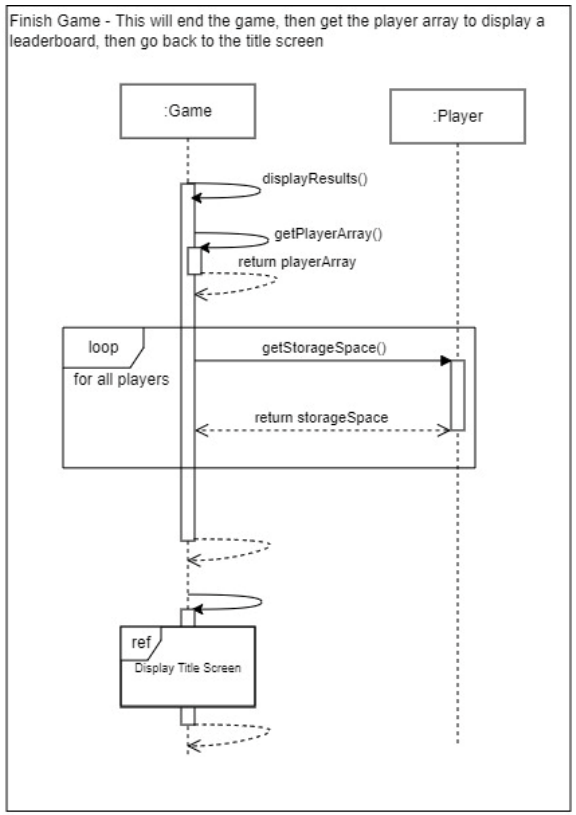

Organise Files - This will display a list of all the files that the current player owns, once a file is selected they then have the option to setup or install a file. A check is done to see how far the user is through the setup process before they can install a file, this is done by checking the currentStage of a file
:Game
:File
:Player
organiseFiles()
getFilesOwned()
return files
displayFilesOwned(files)
user selects an option
opt
selection = file
user selects something to do with a file
opt
option = setup
startSetup(Player user)
getCurrentStage()
return currentStage
getSetupSpaceRequired()
return spaceRequired
getStorageSpace()
return spaceRequired
setStorageSpace()
setCurrentStage()
opt
option = setup
startSetup(Player user)
getCurrentStage()
return currentStage
getSetupSpaceRequired()
return spaceRequired
getStorageSpace()
return spaceRequired
setStorageSpace()
setCurrentStage()
opt
selection = back
displayOptionsForPlayer()
ref
Display Options

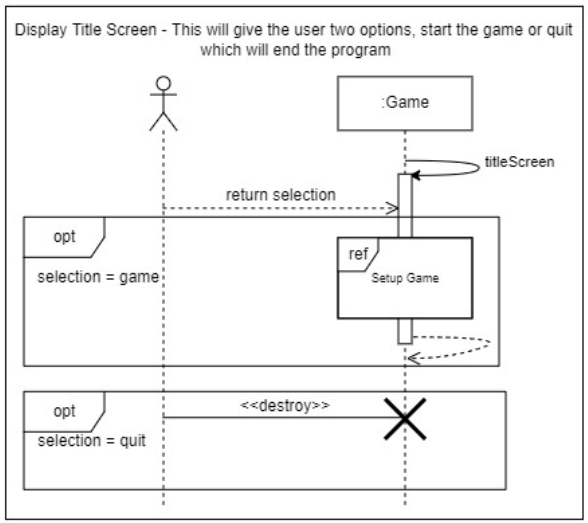
D.L

D.L


Display Options - This will display a menu to the player
:Game
displayOptionsForPlayers
user enters selection
opt
selection = display board
displayBoard()
opt
selection = display rules
displayRules()
opt
selection = Organise Files
organiseFiles()
ref
Organise Files
opt
selection = Roll Dice
rollDice()
ref
Roll Dice
opt
selection = Finish Game
voteToFinishGame()
ref
Vote to Finish Game

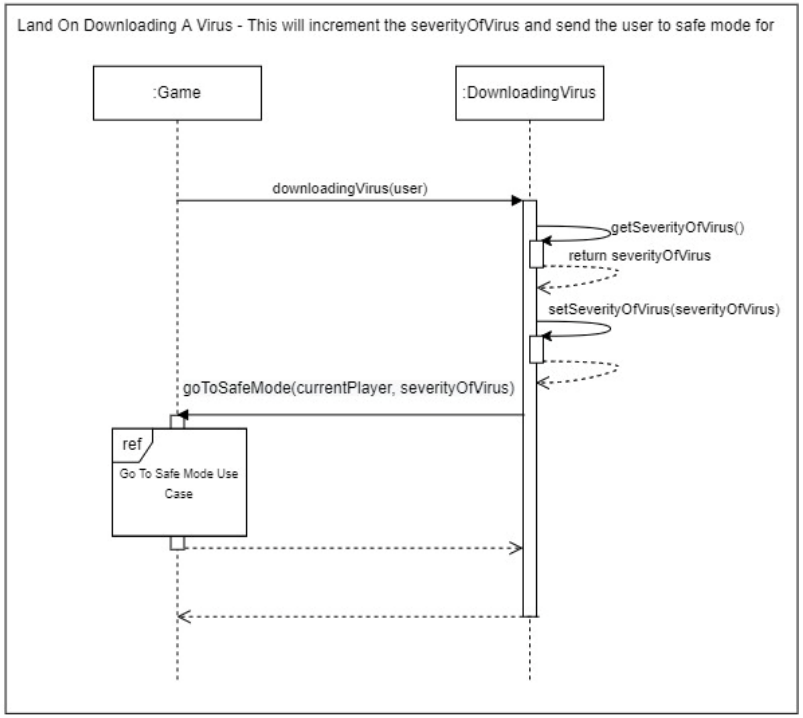
D.L

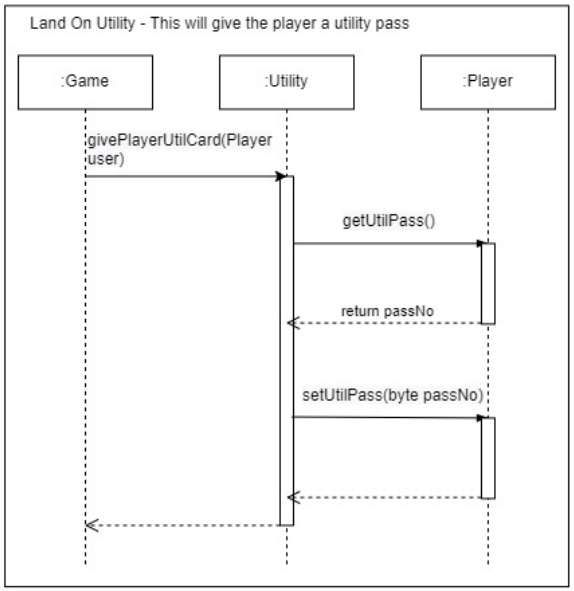

Finish Game - This will end the game, then get the player array to display a leaderboard, then go back to the title screen
:Game
:Player
displayResults()
getPlayerArray()
return playerArray
loop
for all players
getStorageSpace()
return storageSpace
ref
Display Title Screen


Display Title Screen - This will give the user two options, start the game or quit which will end the program
:Game
titleScreen
return selection
opt
selection = game
ref
Setup Game
opt
selection = quit
<<destroy>>

D.L

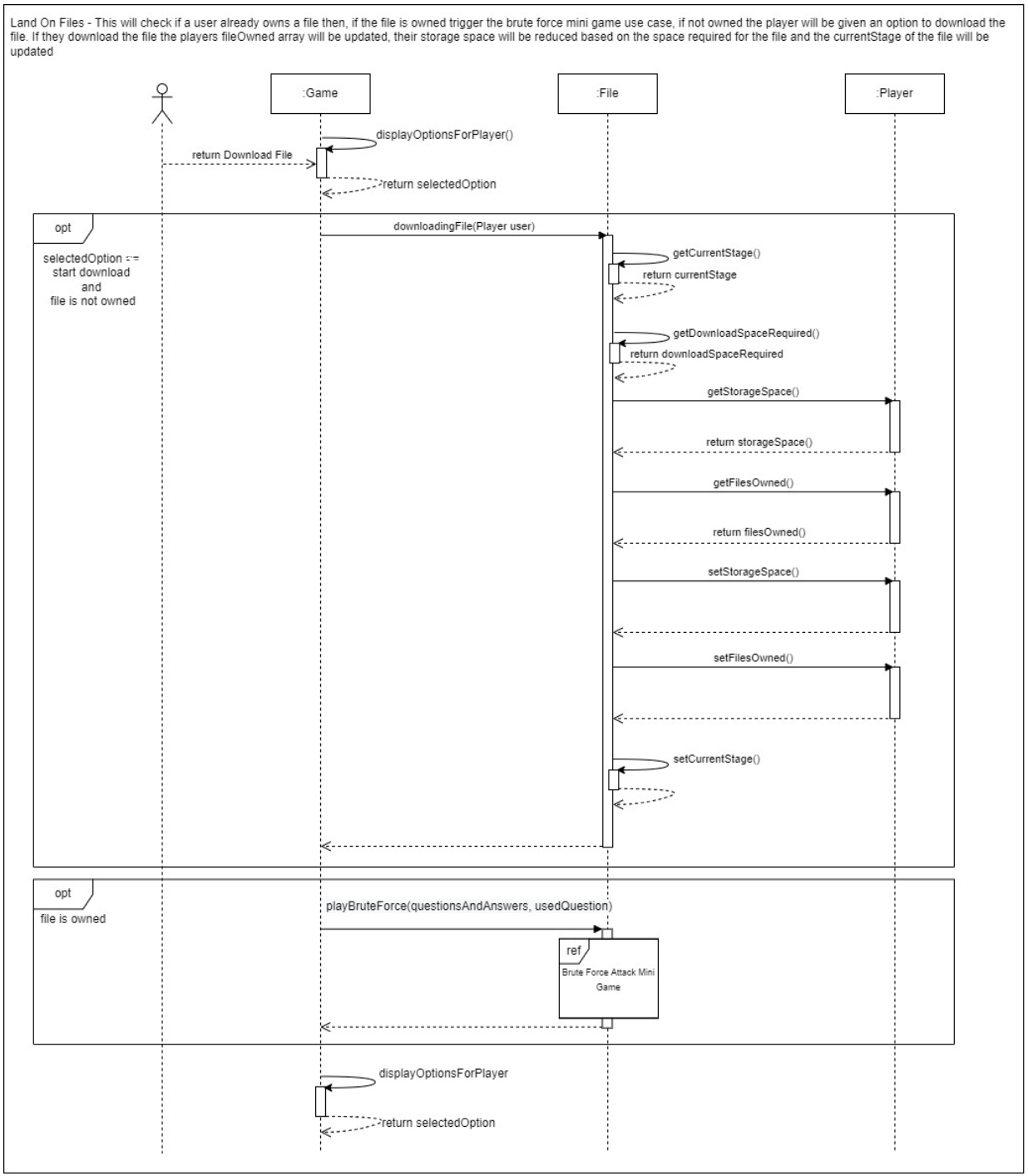
D.L


Land On Downloading A Virus - This will increment the severityOfVirus and send the user to safe mode for
:Game
:DownloadingVirus
downloadingVirus(user)
getSeverityOfVirus()
return severityOfVirus
setSeverityOfVirus(severityOfVirus)
goToSafeMode(currentPlayer, severityOfVirus)
ref
Go To Safe Mode Use Case

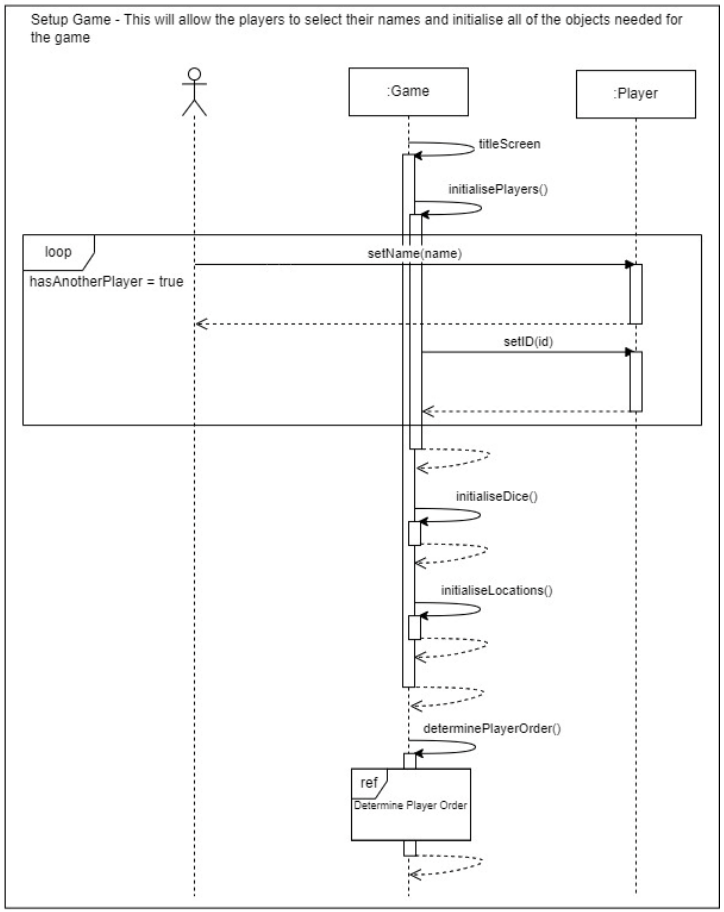

Land On Utility - This will give the player a utility pass
:Game
:Utility
:Player
givePlayerUtilCard(Player user)
getUtilPass()
return passNo
setUtilPass(byte passNo)

D.L

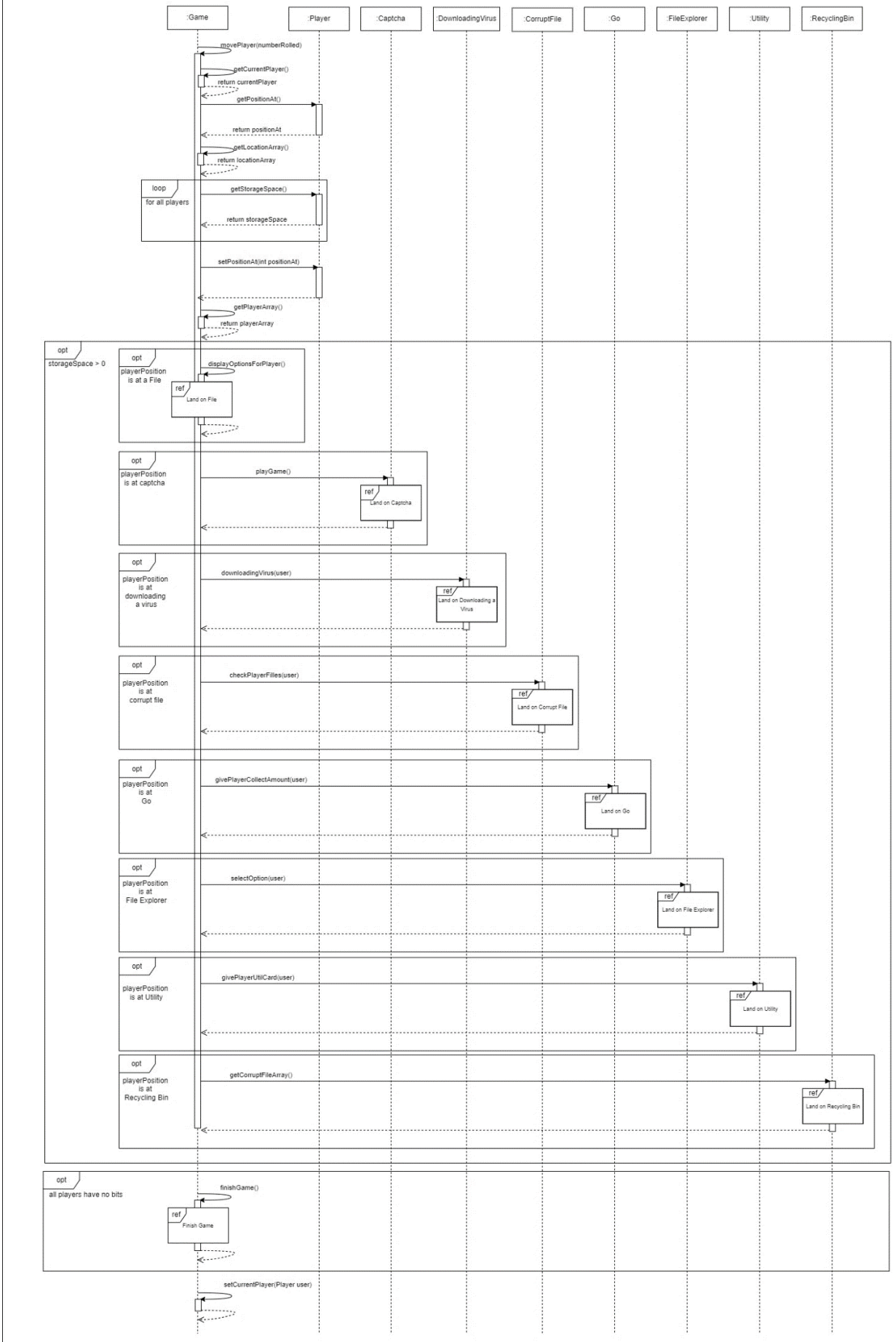
D.L


Land On Files - This will check if a user already owns a file then, if the file is owned trigger the brute force mini game use case, if not owned the player will be given an option to download the file. If they download the file the players fileOwned array will be updated, their storage space will be reduced based on the space required for the file and the currentStage of the file will be updated
:Game
:File
:Player
displayOptionsForPlayer()
return Download File
return selectedOption
opt
selectedOption == start download and file is not owned
downloadingFile(Player user)
getCurrentStage()
return currentStage
getDownloadSpaceRequired()
return downloadSpaceRequired
getStorageSpace()
return storageSpace()
getFilesOwned()
return filesOwned()
setStorageSpace()
setFilesOwned()
setCurrentStage()
opt
file is owned
playBruteForce(questionsAndAnswers, usedQuestion)
ref
Brute Force Attack Mini Game
displayOptionsForPlayer
return selectedOption

D.L


Setup Game - This will allow the players to select their names and initialise all of the objects needed for the game
:Game
:Player
titleScreen
initialisePlayers()
loop
hasAnotherPlayer = true
setName(name)
setID(id)
initialiseDice()
initialiseLocations()
determinePlayerOrder()
ref
Determine Player Order

Move Player - This gets all of the players storage space checks if they are 0, if a players storage space is 0 then that players position will become -1, if only 1 player has a position which is not -1 then the finish game use case is triggered. Assuming the current players storage space is not 0 then there position will be updated and depending on the location they land on the corresponding use case will be triggered, the currentPlayer will be updated to the next element in the playerArray



D.L

Board Layout – C.N, S.M

The Diagram to the right is the draft board layout of our game.

You can also see a list of our different file extensions and their corresponding colour. The .one file extension will consume the most storage space to download while .mpp will consume the least.

There is an arrow indicating the flow of the game.

Each location (square) will be stored within an array called locationArray, it will be stored in the order of the flow of the game. E.g GO is at index 0, then Old Kent Road.mpp is index 1, Corrupt File is index 2, etc.

The files (squares with a coloured bar on them) can be downloaded by a player only whenever they are on that location. Then on any turn the player can start the setup process, before the user can start installing the file they must go through 3 different stages of setup.

The file explorer locations will allow the user to move directly to any of the files that they are currently downloading at the expensive of some storage space.

The Util Pass location means that the player will be able to collect the Go amount if they are moving using file explorer.

Corrupt file will take a random file from the user and add this file to recycling bin.

Recycling Bin will allow the user to get all the files in it if they can remember the order, they were added to recycling bin in.

Downloading a virus will send the player to Safe Mode

Safe Mode is where the player will be until they roll a double or they have reached the amount of time they have to spend in quarantine.

PDF

DOCX

XLSX

PPTX

ONE

MPP

MP4

SVG

Appendix

Weekly Minutes

Minutes for CSC2058 Project 59, Week commencing 11-Oct Date of this minute 13-Oct

The following team members were present on Teams when minutes were discussed:

Name (printed/typed)	Signature (agreed bitmap or initials)
Scott McDonald	S.McD.
Dean Logan	D.L.
Conor Nugent	C.N.
John B Higgins	J.B.H.

Task Reporting (Briefly list the progress for each team member in the last week.*)

Name (1): Scott McDonald

- Proposed game design/idea
- Proposed game in which the user was an up and coming industrialist in America during the Victorian era.
- Allowed the user to build trains on properties which, they could then ride.

Name (2): Dean Logan

- Proposed Game design/idea
- Proposed an adventure board game.
- Users could travel both sides of the board and would complete challenges, quizzes and stories to earn money or take over property.
- Players could also attack another players property.

Name (3): Conor Nugent

- Proposed game design/idea
- Proposed game in which users would travel around space.

Name (4): John B Higgins

- Proposed game design/idea
- Proposed game which used servers as properties and players travelled around the web (board)..

Actions Planned (Briefly list the actions required of each team member for the next week.)

Name (1): Scott McDonald

- Further expand game idea
- Plan out potential use cases

Name (2): Dean Logan

- Further expand game idea
- Plan out potential use cases

Name (3): Conor Nugent

- Further expand game idea
- Plan out potential use cases

Name (4): John B Higgins

- Further expand game idea
- Plan out potential use cases

Minutes for CSC2058 Project 59, Week commencing 18-Oct Date of this minute 20-Oct

The following team members were present on Teams when minutes were discussed:

Name (printed/typed)	Signature (agreed bitmap or initials)
Scott McDonald	S.McD.
Dean Logan	D.L.
Conor Nugent	C.N.
John B Higgins	J.B.H.

Task Reporting (Briefly list the progress for each team member in the last week.*)

Name (1): Scott McDonald

- Design was expanded.
- Relevant names were assigned to property
- How the game could be coded was planned

Name (2): Dean Logan

- Design was expanded
- Game was abandoned as didn't fit with requirements
- Assisted SMcD and JBH with game design

Name (3): Conor Nugent

- Design was expanded
- Game was abandoned
- Assisted SMcD and JBH with game design

Name (4): John B Higgins

- Design was expanded
- Design was changed from servers to a hard disk drive – players would be the arms.
- How the game could be coded was planned

Actions Planned (Briefly list the actions required of each team member for the next week.)

Name (1): Scott McDonald

- Take useful elements from other designs and see how it could be used to improve others.
- Decide between Railway and Storage proposal.

Name (2): Dean Logan

- Take useful elements from other designs and see how it could be used to improve others.
- Decide between Railway and Storage proposal.

Name (3): Conor Nugent

- Take useful elements from other designs and see how it could be used to improve others.
- Decide between Railway and Storage proposal.

Name (4): John B Higgins

- Take useful elements from other designs and see how it could be used to improve others.
- Decide between Railway and Storage proposal.

Minutes for CSC2058 Project 59, Week commencing 25-Oct Date of this minute 27-Oct

The following team members were present on Teams when minutes were discussed:

Name (printed/typed)	Signature (agreed bitmap or initials)
Scott McDonald	S.McD.
Dean Logan	D.L.
Conor Nugent	C.N.
John B Higgins	J.B.H.

Task Reporting (Briefly list the progress for each team member in the last week.*)

Name (1): Scott McDonald

- Elements of game design was expanded
- Users could create links/”tracks” between properties that they owned that they could then travel on.

- Figured out a way on how to incorporate this into

Name (2): Dean Logan

- Elements of game design were expanded
- Users could complete a quiz or challenge when landing on a property – if they fail they pay an increased tax, pass then a decreased one
- Users could complete a quiz or challenge as a part of the community or chance cards.

Name (3): Conor Nugent

- Expanded Hard Disk game elements

Name (4): John B Higgins

- Expanded Game Design

Actions Planned (Briefly list the actions required of each team member for the next week.)

Name (1): Scott McDonald

- Plan use case
- Expand game elements

Name (2): Dean Logan

- Plan use case
- Expand game elements

Name (3): Conor Nugent

- Plan use case
- Expand game elements

Name (4): John B Higgins

- Plan use case
- Expand game elements

Minutes for CSC2058 Project 59, Week commencing 15-Nov Date of this minute 19-Nov

The following team members were present on Teams when minutes were discussed:

Name (printed/typed)	Signature (agreed bitmap or initials)
Scott McDonald	S.McD.
Dean Logan	D.L.
Conor Nugent	C.N.
John B Higgins	J.B.H.

Task Reporting (Briefly list the progress for each team member in the last week.*)

Name (1): Scott McDonald

- Finished all assigned use cases
- Defined basic board layout
- Began work on class diagrams of the use cases they wrote descriptions for
- Reworked use case diagram
- Finished class diagrams
- Began working on realisations

Name (2): Dean Logan

- Finished all assigned use cases
- Defined basic board layout
- Began work on class diagrams of the use cases they wrote descriptions for
- Reworked use case diagram
- Merged completed use case descriptions
- Finished class diagrams
- Began working on realisations

Name (3): Conor Nugent

- Finished all assigned use cases
- Defined basic board layout
- Began work on class diagrams of the use cases they wrote descriptions for
- Finished class diagrams
- Began working on realisations

Name (4): John B Higgins

- Finished all assigned use cases
- Defined basic board layout
- Began work on class diagrams of the use cases they wrote descriptions for
- Reworked use case diagram
- Finished class diagrams
- Began working on realisations

Actions Planned (Briefly list the actions required of each team member for the next week.)

Name (1): Scott McDonald

- Continue working on realisations

Name (2): Dean Logan

- Continue working on realisations

Name (3): Conor Nugent

- Continue working on realisations

Name (4): John B Higgins

- Continue working on realisations

Minutes for CSC2058 Project 59, Week commencing 22-Nov Date of this minute 26-Nov

The following team members were present on Teams when minutes were discussed:

Name (printed/typed)	Signature (agreed bitmap or initials)
Scott McDonald	S.McD.
Dean Logan	D.L.
Conor Nugent	C.N.
John B Higgins	J.B.H.

Task Reporting (Briefly list the progress for each team member in the last week.*)

Name (1): Scott McDonald

- Finished first set of assigned use case realisations
- Began new ones
- Finished final use case realisations

Name (2): Dean Logan

- Rework all use case diagrams
- Finished first set of assigned use case realisations
- Began new ones
- Finished final use case realisations
- Merged completed realisations

Name (3): Conor Nugent

- Finished first set of assigned use case realisations
- Began new ones
- Finished final use case realisations

Name (4): John B Higgins

- Finished first set of assigned use case realisations
- Began new ones
- Finished final use case realisations
- Began work on demo

Actions Planned (Briefly list the actions required of each team member for the next week.)

Name (1): Scott McDonald

- Go over realisations

- Go over use case descriptions
- Go over use case diagram

Name (2): Dean Logan

- Go over realisations
- Go over use case descriptions
- Go over use case diagram

Name (3): Conor Nugent

- Go over realisations
- Go over use case descriptions
- Go over use case diagram

Name (4): John B Higgins

- Go over realisations
- Go over use case descriptions
- Go over use case diagram

Minutes for CSC2058 Project 59, Week commencing 29-Nov Date of this minute 1-Dec

The following team members were present on Teams when minutes were discussed:

Name (printed/typed)	Signature (agreed bitmap or initials)
Scott McDonald	S.McD.
Dean Logan	D.L.
Conor Nugent	C.N.
John B Higgins	J.B.H.

Task Reporting (Briefly list the progress for each team member in the last week.*)

Name (1): Scott McDonald

- Finished minutes
- Went over use case realisations
- Reworked use case descriptions
- Completed PDF
- Completed Peer Review
- Completed Personal Statement

Name (2): Dean Logan

- Went over demo
- Completed gantt chart
- Reworked use case realisations
- Completed PDF
- Completed Peer Review

- Completed Personal Statement

Name (3): Conor Nugent

- Went over demo
- Went over use case realisations
- Reworked use case descriptions
- Reworked use case diagram
- Completed PDF
- Completed Peer Review
- Completed Personal Statement

Name (4): John B Higgins

- Finished demo
- Went over use case realisations
- Reworked use case descriptions
- Completed PDF
- Completed Peer Review
- Completed Personal Statement