Project Scope Statement

Project title: Hacktopoly

Project Scope/Objective Description

The project is aimed at completing the development of a monopoly-inspired software.

Project Team Members

- Yuanzhe Zhang (40276087)
- Niall O'Neill (40297272)
- Jamaica De Guzman (40137844)
- Roche Francis Palen (40301756)
- Marc Villareal (402976087)
- Matthew Hutchinson (40112152)
- Lorenzo Cueto (40302258)

Project Deliverables

- A **use case diagram** representing the main sets of sequences of user-system interaction and a **corresponding set of written use case descriptions.**
- A **Gantt chart** indicating the main development strands and deliverables over the whole project lifetime.
- An **initial class diagram** representing the most important concepts in the application domain.
- **Use case realisations in the form of sequence diagrams** that show the main sets of sequences of interaction.
- A **Draft Game Layout** & **Final Game Layout.**
- An **Interim Demo** & **Final Demo**.
- **Peer assessment** for the problem and early solution and a **final peer assessment** for the final solution.
- A working system developed in Java.
- Design Documentation.
- A **test plan** based on the original use case requirements specification.
- Final PDF report.

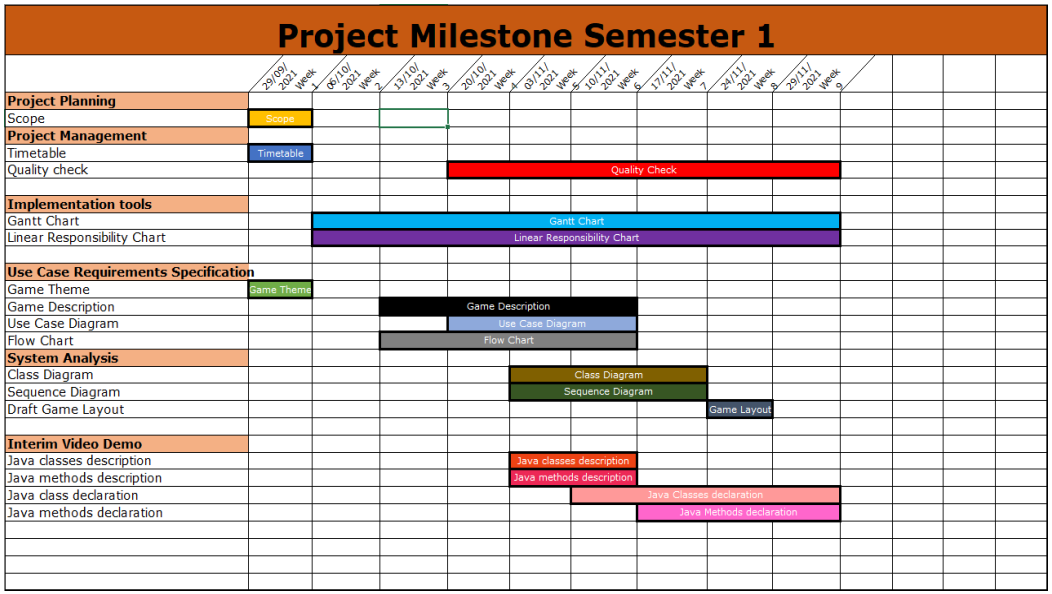
Project Acceptance Criteria

- Semester 1 PDF Report & Semester 2 PDF Report.
- Semester 1 Interim Video Demo & Semester 2 Video Demo and Code.
- Semester 1 Peer Assessment & Semester 2 Peer Assessment.

Project Constraints

- System to be developed exclusively in java.
- Time/resource/personnel limitations.

Project Milestone Semester 1



Linear Responsibility Chart Semester 1

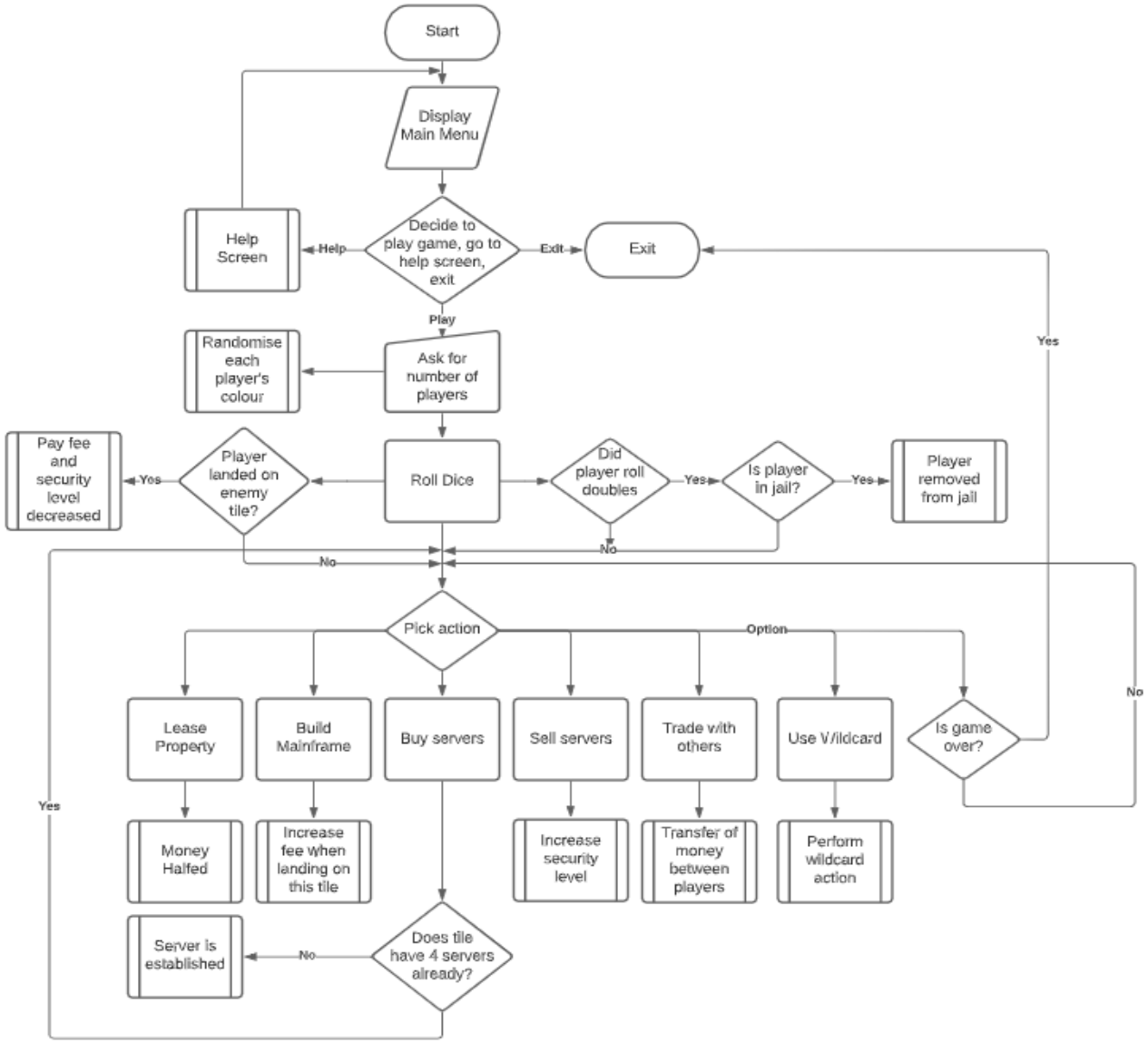
Legend: Responsible = **R** Support = **S** Quality Checker = **QC** Verifier = **V**

	Lorenzo Cueto	*Marc Vincent Villareal*	*Roche Francis Palen*	*Jamaica De Guzman*	*Matthew Hutchinson*	*Yuanzhe Zhang*	*Niall O'Neill*
Tasks							
Write Scope	QC		R	S	V		
Create Timetable		V	S	R	QC		
Create Quality Check Plan	V		S	R	QC		
Write Weekly Minutes			S	R	V	QC	
Update Quality Check Plan	QC		R	V	S		
Update Weekly Minutes	S	QC	V	R			
Create Gantt Chart	V		S	R			
Update Gantt Chart	V		S	R	QC		
Create Game Theme	R	S	R	V	QC	S	S
Write Game Description	QC	S	V			R	S
Create Use Case Diagram	V	S			QC	R	S
Create Flow Chart	QC	S			V	S	R
Create Class Diagram	R		QC	V	S		
Create Sequence Diagram	S		QC	V	R		
Draft Game Layout	S		V	R	QC		
Write Java Classes Description	S		QC	V	R		
Write Java Methods Description	S		QC	V	R		
Declare Java Classes	R		QC	V	S		
Declare Java Methods	R		QC	V	S		
Record&Edit Video Demo	R		S		V		QC
Project PDF Documentation	QC		R	V	S		
Submission	QC		R	V	S		

Quality Check Plan Semester 1

Actions	Plan	Check	Finished
Scope	Write scope describing the mechanics of the project	Scope written	✓
Timetable	Create timetable to show the milestone of project tasks	Timetable created	✓
Quality Check Plan	Create quality check plan that contains the description of the actions	Quality Check Plan created	✓
Weekly Minutes	Write weekly minutes after each team meeting with a brief layout of tasks and action plan	Weekly Minutes written	✓
Quality Check Plan	Update quality check plan to make sure actions are up to date	Quality Check updated	✓
Update Weekly Minutes	Update weekly minutes from the previous week's action plan and create new action plan for the following week	Weekly Minutes updated	✓
Gantt Chart	Create gantt chart to show the weekly progress of each tasks of the whole project	Gantt Chart created	✓
Update Gantt Chart	Update gantt chart weekly monitoring the progress of each tasks	Gantt Chart updated	✓
Game Theme	Create game theme describing the idea and framework of the project game	Game Theme created	✓
Game Description	Write game description to show the mechanics of the game	Game Description written	✓
Use Case Diagram	Create Use Case Diagram describing the functions between the system and actor.	Use Case Diagram created	✓
Flow Chart	Create Flow Chart to illustrate the flow of events of the game	Flow Chart created	✓
Class Diagram	Create Class Diagram to define the classes and methods	Class Diagram created	✓
Sequence Diagram	Create Sequence Diagram showing the set of action sequences of the game	Sequence Diagram created	✓
Draft Game Layout	Create a draft Game Layout to show the visual image of the game	Draft Game Layout created	✓
Java Classes Description	Write Java Classes Description determining the classes of the game	Java Classes description written	✓
Java Methods Description	Write Java Methods Description determining the methods of the classes	Java Methods descriptions written	✓
Declare Java Classes	Declare Java Classes defining the classes of the game	Java Classes declared	✓
Declare Java Methods	Declare Java Methods defining the functions of each methods in the class	Java Methods declared	✓
Video Demo	Record&Edit Video Demo to showcase the prototype code fragment of the game	Video recored & edited	✓
Project PDF	Project PDF documenting the whole body of the project in PDF format	Project documented	✓
Submission	Submission requirements needs to be met checking the deliverables	Submission done	✓

Flow Chart


Start
Display Main Menu
Decide to play game, go to help screen, exit
Help
Help Screen
Exit
Exit
Play
Ask for number of players
Randomise each player's colour
Roll Dice
Did player roll doubles
Yes
Is player in jail?
Yes
Player removed from jail
No
Player landed on enemy tile?
Yes
Pay fee and security level decreased
No
Pick action
Option
Lease Property
Build Mainframe
Buy servers
Sell servers
Trade with others
Use Wildcard
Is game over?
Yes
No
Money Halfed
Increase fee when landing on this tile
Increase security level
Transfer of money between players
Perform wildcard action
Does tile have 4 servers already?
No
Server is established
Yes

Hacktopoly

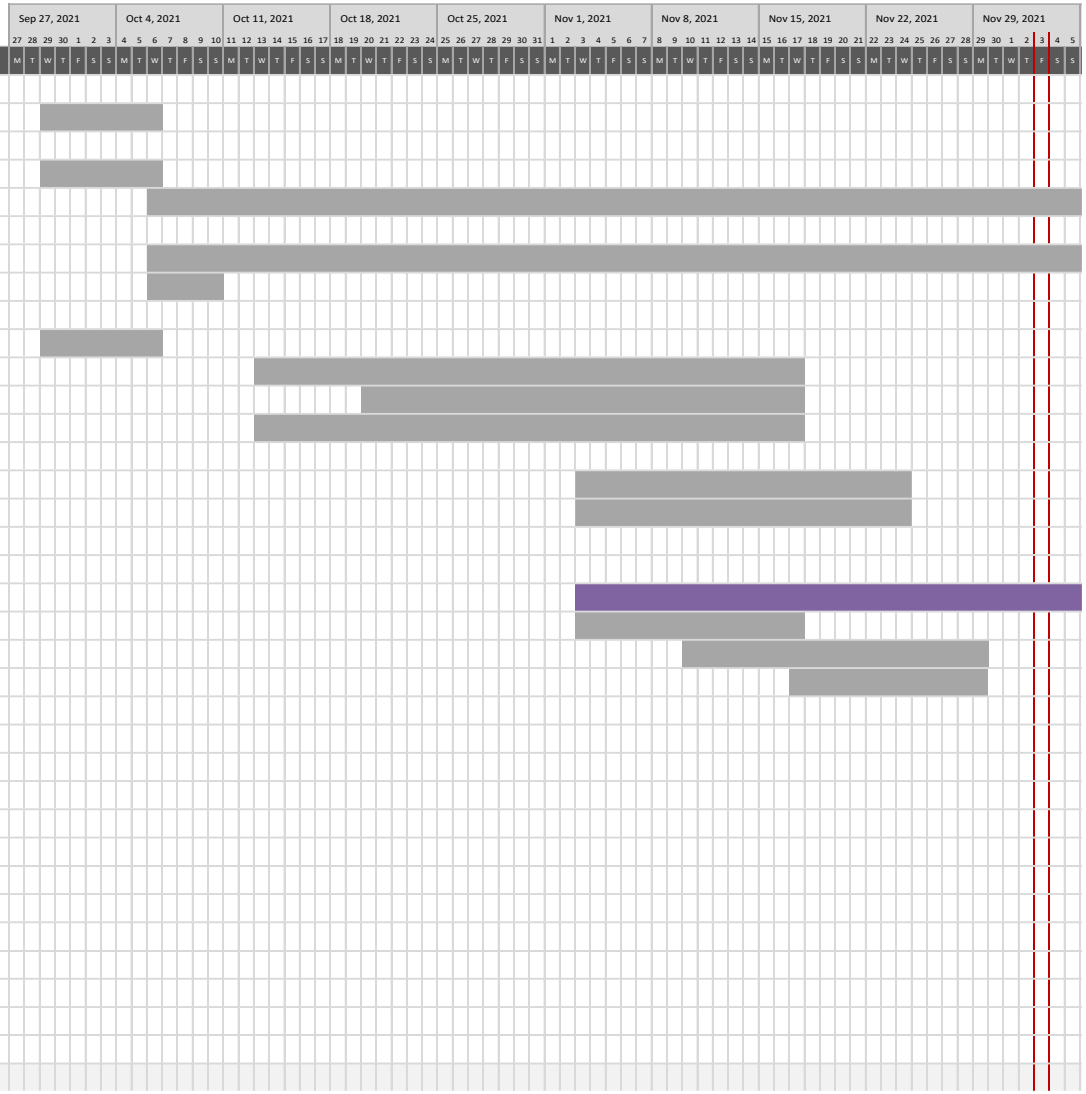
Legend **R.F.P** = Roche Francis Palen, **J.D** = Jamaica De Guzman, **M.V** = Marc Villareal, **L.C** = Lorenzo Cueto, **M.H** = Matthew Hutchinson, **Y.Z** = Yuanzhe Zhang, **N.O** = Niall O'Neill

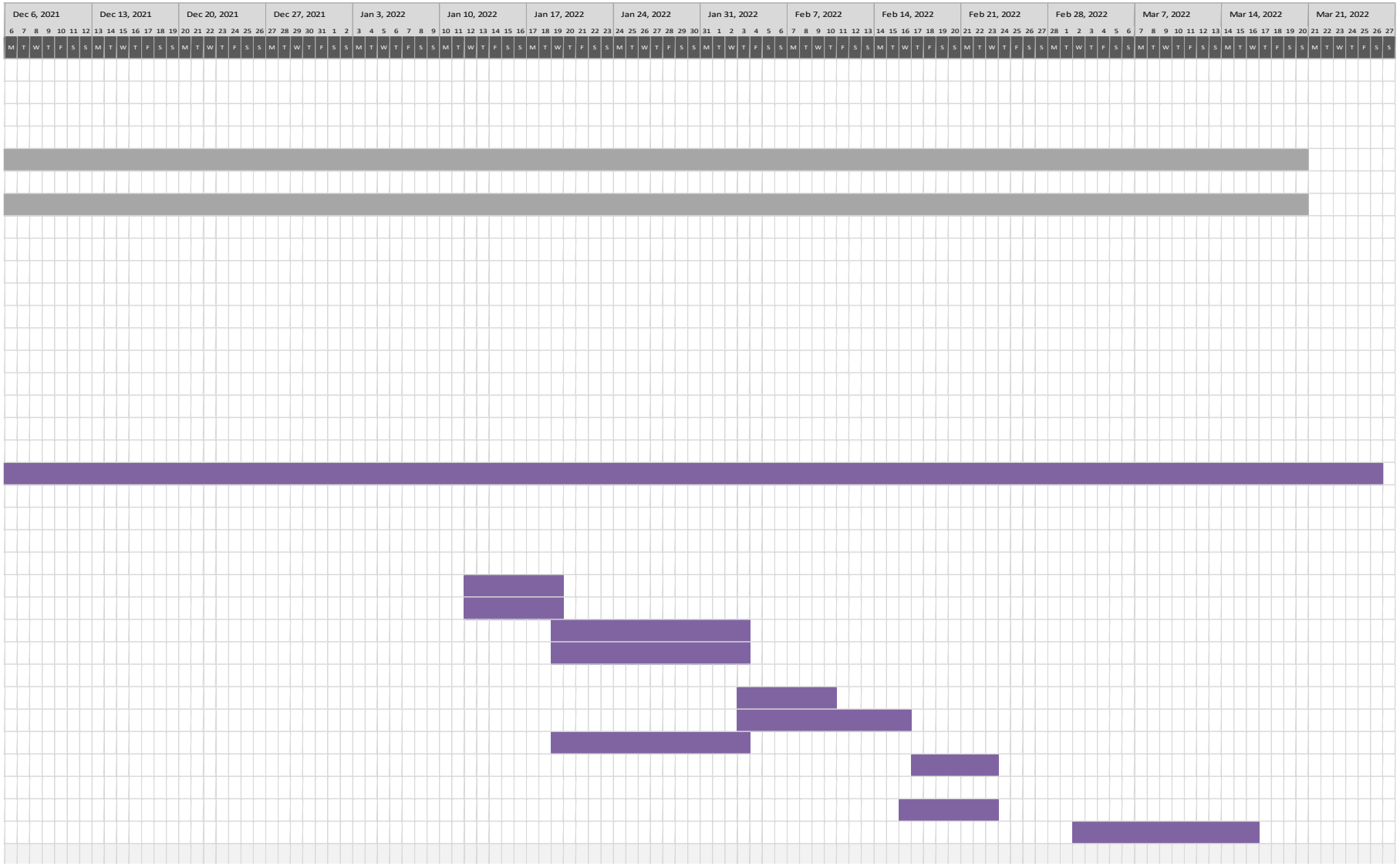
Jamaica De Guzman & Roche Francis Palen

Project Start: Tue, 9/28/2021

Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
1. Project Planning				
Scope	R.F.P	100%	9/29/21	10/6/21
2. Project Management				
Timetable	J.D	100%	9/29/21	10/6/21
Quality Check	J.D	100%	10/6/21	3/20/22
3. Implementation Tools				
Gantt Chart	R.F.P / J.D	100%	10/6/21	3/20/22
Linear Reponsibility Chart	J.D	100%	10/6/21	10/10/21
4. Use Case Requirements Specification				
Game Theme	R.F.P / L.C	100%	9/29/21	10/6/21
Game Description	Y.Z	100%	10/13/21	11/17/21
Use Case Diagram	Y.Z / M.V / N.O	100%	10/20/21	11/17/21
Flow Chart	Y.Z / M.V / N.O	100%	10/13/21	11/17/21
5. System Analysis				
Class Diagram	L.C	100%	11/3/21	11/24/21
Sequence Diagram	M.H	100%	11/3/21	11/24/21
Draft Game Layout	J.D	100%	11/24/21	11/29/2021
6. Interim Video Demo				
Java Classes Description	M.H	100%	11/3/21	11/17/2021
Java methods description	M.H	100%	11/3/21	11/17/21
Java classes declaration	L.C	100%	11/10/21	11/29/21
Java methods declaration	L.C	100%	11/17/21	11/29/21
7. The Working System				
Java classes description revised	M.H / L.C / R.F.P	0%	1/12/22	1/19/22
Java methods description revised	M.H / L.C / R.F.P	0%	1/12/22	1/19/22
Java classes declaration revised	M.H / L.C / R.F.P	0%	1/19/22	2/3/22
Java methods declaration revised	M.H / L.C / R.F.P	0%	1/19/22	2/3/22
8. Design Documentation				
Text User Interface Samples Prompts	J.D / M.V	0%	2/3/22	2/10/22
Class Relationship Model	N.O / Y.Z	0%	2/3/22	2/16/22
Sequence diagram revised	M.H	0%	1/19/22	2/3/22
Final Game Layout	J.D	0%	2/17/22	2/23/22
9. Implementation-Related Documentation				
Test Plan	R.F.P / J.D	0%	2/16/22	2/23/22
JUnit Testing	M.H / L.C / M.V	0%	3/2/22	3/16/22
Insert new rows ABOVE this one				




Dec 6, 2021
Dec 13, 2021
Dec 20, 2021
Dec 27, 2021
Jan 3, 2022
Jan 10, 2022
Jan 17, 2022
Jan 24, 2022
Jan 31, 2022
Feb 7, 2022
Feb 14, 2022
Feb 21, 2022
Feb 28, 2022
Mar 7, 2022
Mar 14, 2022
Mar 21, 2022

Game Theme

Theme name	Hacktopoly
Theme summary	Players have two variables money and security value. Money is used to upgrade buy servers to take over them, but security value is more important, affecting winning or losing the game. Players will move around the board and land on different servers which represents how people normally browse the internet. The goal is for players to lower the security level and steal the money of the other players to hack them. Players will take control of servers and use wildcards to hack the other players.
Properties/ Resources	A group of Property tiles on the board represent companies you can take over. Individual tiles in a group represent countries which a company operates in e.g., America, EU, Asia. Tech companies have separate colours all around the board. You take over a company tile (property) by spending money. You can only take over the servers on an individual property when you have taken over all other related tiles of a single company.
Building houses/ Resources	A player can build and own up to 4 server per company tile. server(house). After building 4 servers, the player has the option to replace all 4 servers on the tile with a mainframe which incurs a higher fee from opposing players.
Rent	Instead of paying rent, the player that lands on an owned server tile loses some of their security rating, which represents the said player losing data/passwords. Players will also lose some of their money. The data/money given will scale with the number of servers on the tile or if the tile has a mainframe.
Selling houses/Resources	When in need of increasing security level due to being hacked by other players, players can remove their control over some servers they have on a property to increase their own security. They will have to spend money again to upgrade it
Mortgages	A player can lease their property to receive a half of their security rating value and monetary price. Should another player land on the tile then no data fees will be incurred upon the player.
Trading	Trading will be set out as usual. There will be a give and take of property and/or money. Trading will remain unrestricted meaning a player can absolutely request and not give anything back at all. Players make a request to trade with one player and the player receiving the request can either accept or decline.
Wildcards	Like monopoly there are two wildcard types these are **Attack vectors**, more focussed on targeting other players and **Security cards** which are focused on protecting yourself from other players or providing advantageous bonuses. Wildcard targets will either be the current player or another player/players that the current player chooses. A Wildcard's effect which targets the current player will just occur, but a player can choose to use a card that targets another player/players. **Examples – Attack Vectors**

	• **Security Breach –** The current player selects another player from the game and hacks them directly, lowering the security level of the other player and adding it to the current player's security level. • **Scam –** Players that are targeted by this card lose some of their money to the player that is using it on them • **Malware leak** – The hacked company retains control over one of the servers a player has on a tile, reducing the tile's rent value. **Examples – Security Cards** • **Encryption** – players implement more security for themselves increasing their security level a certain value • **VPN** – When a player lands on another player's tile where they have taken over a server, they will not lose any money or security level
Jail	**Ransomware attack** When players land on the ransomware attack tile, they go to the vulnerable tile and remain there for 3 turns. If they roll 2 6s they can get out early but while they are in there, they lose 100 money and 100 security level each turn
Parking (if applicable)	Security search tiles will have players either pay a fee or none.
Winning the game	Player wins when the security of all other players drops to 0 and those players don't own any other servers/malware assets. When the security level drops to 0, the player automatically loses all the money they have to the player tile that they landed on, and they are eliminated from the game.

Game Descriptions

Flow of Events for the *Pre-game Setup* use-case	
Objective	To register each player's name and initialize their status.
Pre-condition	The game is successfully loaded.
Main Flow (for each player)	1. The system display current player's number.
	2. The player type in their name.
	3. The player is given a designated amount of money and security value.
	4. Next player repeat the process until every player is registered.
Alternative Flows	At 2, if the name typed in is illegal (contain illegal character or too long), the system displays a warning, player is asked to type in again.
Post-condition	Player is ready for gameplay. Game-play menu will display.

Flow of Events for the *Game Play* use-case	
Objective	To allow player select one of the game-play actions.
Pre-condition	The player has done pre-game registration.
Main Flow	1. The system display all available game-play options, plus "next player" option.
Alternative Flows	At 1, if the player selects "next player" option, the Game-play Menu will display for the next player.
Post-condition	Player will start the selected game-play action.

Flow of Events for the *Roll Dice* use-case	
Objective	To determine the next tile the player will land on. The player could go to jail if unlucky.
Pre-condition	The player selected "Roll Dice" in the Game-play Menu.
Main Flow	1. The player choose to start rolling dice.
	2. The system returns the result (1 to 6).
	3. The player lands on a new tile according to the result.
	4. The player pays rent and lose security value if the new tile is owned by another player. The more servers there is on the new tile, the more rent the player should pay, the more security value the player will lose.
	5. The system display the game-play menu again based on the updated location.
Alternative Flows	At 3, if the player lands on a jail tile, Go Jail use case will be utilized.
	At 4, if the player loses all their security value, they will be out of the game.
Post-condition	Player's location is updated.

Flow of Events for the *Go jail* use-case	
Objective	To stop the player from doing anything or moving for three rounds.
Pre-condition	The player is sent to jail by landing on a jail tile.
Main Flow	1. The system shows a message saying the player is now in jail and explaining the rule of jail.
	2. The player will be asked if they want to pay bribe to get out of jail
	3. If bribe is not paid, the player will remain in jail for three rounds.
Alternative Flows	At 2, if bribe is paid, the player will be allowed to roll dice to go out of jail.
Post-condition	Player will be assigned “in jail” status.

Flow of Events for the *Buy Company* use-case	
Objective	To buy a branch of the company (represented by a tile)
Pre-condition	The player has rolled a dice and landed on the tile.
Main Flow	1. The system check if the player has already brought the company branch.
	2. The price for the company branch is displayed.
	3. The system ask the player if they want to buy the company branch or not.
	4. The company branch is marked as owned by the player. The player’s money is deduced.
Alternative Flows	At 1, If the player already owns the company branch, the system displays a message and return to the game-play menu.
	At 3, If the player does not have enough money, the system displays a warning and return to the game-play menu.
Post-condition	Player will be able to build servers on the tile if they own every branch of the company.

Flow of Events for the *Auction Company* use-case	
Objective	To put a branch of the company (represented by a tile) on auction.
Pre-condition	The player has rolled a dice and landed on the tile.
Main Flow	1. The system check if the player has already brought the company branch.
	2. The player is asked to put a starting price
	3. Any player may put a higher bid within a certain amount of time.
	4. The last bidder when the time is up is the winner
	4. The company branch is marked as owned by the winner. The winner’s money is deduced.
Alternative Flows	At 1, If the player already owns the company branch, the system displays a message and return to the game-play menu.
	At 3, If the bidder enters an amount that is not greater than the current bid, or it is exceeding their total money, the bid will not be allowed. The system displays a warning, and players are asked to bid again.

Post-condition	The winner will be able to build servers on the tile if they own every branch of the company.

Flow of Events for the *Build Servers* use-case	
Objective	To build a server on the selected tile.
Pre-condition	The player has rolled a dice and landed on the property.
Main Flow	1. The system lists every tile the player owns.
	2. The player selects one of the tiles.
	3. The system prints out how much money and security level it will cost to build a server and ask for confirmation.
	4. The player confirms the decision.
	5. The number of servers on the selected tile increases by one.
Alternative Flows	At 1, If the player does not own any tile. The system displays a warning. Player will return to the game-play menu.
	At 2, If the selected tile already has 4 servers or a mainframe built onto it. The system displays a warning. Player will return to the game-play menu.
	At 4, if the player regrets the decision, Player will return to the game-play menu.
Post-condition	Player will return to the game-play menu.

Flow of Events for the *Build Mainframe* use-case	
Objective	To build a mainframe on the selected tile. (Note: a mainframe can only be built if there are already 4 servers, and the player cannot build more than one mainframe)
Pre-condition	The player has rolled a dice and landed on the property.
Main Flow	1. The system lists every tile the player owns.
	2. The player select one of the tiles.
	3. The system prints out how much money and security level it will cost to build a mainframe and ask for confirmation.
	4. The player confirms the decision.
	5. The number tile now has a mainframe.
Alternative Flows	At 1, If the player does not own any tile. The system displays a warning. Player will return to the game-play menu.
	At 2, If player has less than 4 servers already built on the tile, or there is already a mainframe, Player will return to the game-play menu.
	At 4, if the player regrets the decision, Player will return to the game-play menu.
Post-condition	Player will return to the game-play menu.

Flow of Events for the *Lease Property* use-case	
Objective	To regain half of the money and security level that has spent of the property, with the cost of no longer able to hack other players when they land on it.
Pre-condition	The player has rolled a dice and landed on the property.

Main Flow A	1. The system list every property (servers or mainframe) the player owns.
	2. The player select one of the properties.
	3. The system display the money and security level the player will regain and ask for player's confirmation.
	4. The player confirms the decision.
	5. The tile is marked as "leased".
Alternative Flows	At 1, if the player does not own any property. The system will display a warning. The player cannot do anything until next round.
	At 4, if the player regrets the decision, nothing will change. The player can do nothing until next round.
Post-condition	The player gained money and security level. Player will return to the game-play menu.

Flow of Events for the *Sell Properties* use-case	
Objective	To sell properties (server and mainframe) the player owns.
Pre-condition	The player selected "Sell Properties" in the Game-play Menu.
Main Flow	1. The system lists every server the player owns.
	2. The player select one of the servers.
	3. The system prints out how much money and security level the player will gain by selling the server and ask for confirmation.
	4. The player confirms the decision.
	5. The number of servers on the selected tile decreases.
Alternative Flows	At 1, If the player does not own any servers. The system displays a warning. Player will return to the game-play menu.
	At 4, if the player regrets the decision, Player will return to the game-play menu.
Post-condition	The player gained money and security level but lost the property being sold. Player will return to the game-play menu.

Flow of Events for the *Trade with Others* use-case	
Objective	To allow the player sell any of property they own by making an offer to other players
Pre-condition	The player selected "Trade with Others" in the Game-play Menu.
Main Flow	1. The system list every property (servers or mainframe) the player owns.
	2. The player select one of the properties and enter a price to make an offer.
	3. The system list every other player
	4. The player select a target player to send an offer.
	5. The target player either accept or decline the offer. If the offer is accepted, money is transfer to the target player, while the property is transferred to the current player.
Alternative Flows	At 1, If the player doesn't own any property. The system displays a warning. Players return to the game-play menu.
	At 2, If the player enters an illegal amount of money (e.g., £-1000). The system displays a warning. Player will return to the game-play menu.

	At 4, If the target player declines the offer. Player will return to the game-play menu.
Post-condition	Player will return to the game-play menu.

Flow of Events for the *Use Wildcards* use-case	
Objective	To allow the player select one type of wildcards to use.
Pre-condition	The player selected “Use Wildcards” in the Game-play Menu.
Main Flow	1. The system display five types of wildcards, plus “return to game-play menu” option.
Alternative Flows	At 1, if the player selects “return to game-play menu” option, the Game-play Menu will display again.

Flow of Events for the *Use Bonus Card* use-case	
Objective	To award the player a random amount of money and security value.
Pre-condition	The player selected “Use Bonus Card” in the wildcard’s selection menu. The player has a Bonus Card.
Main Flow	1. The system generate two random numbers according to pre-set rules, then display the numbers.
	2. The player gained the generated amount of money and security value.
Post-condition	The player gained a random amount of money and security value. Player will return to the game-play menu.

Flow of Events for the *Use Damage Card* use-case	
Objective	To allow the player to attack another player by reducing their money and security value by a random amount.
Pre-condition	The player selected “Use Damage Card” in the wildcard’s selection menu. The player has a Damage Card.
Main Flow	1. The system list every other player, asking the player to select a target.
	2. The system generate two random numbers according to pre-set rules, then display the numbers.
	3. The targeted player lose the generated amount of money and security value.
Alternative Flow	At 3, the targeted player could lose the game if the randomly generated security value is more than the value they already have.
Post-condition	The target player loses money and security value. Player will return to the game-play menu.

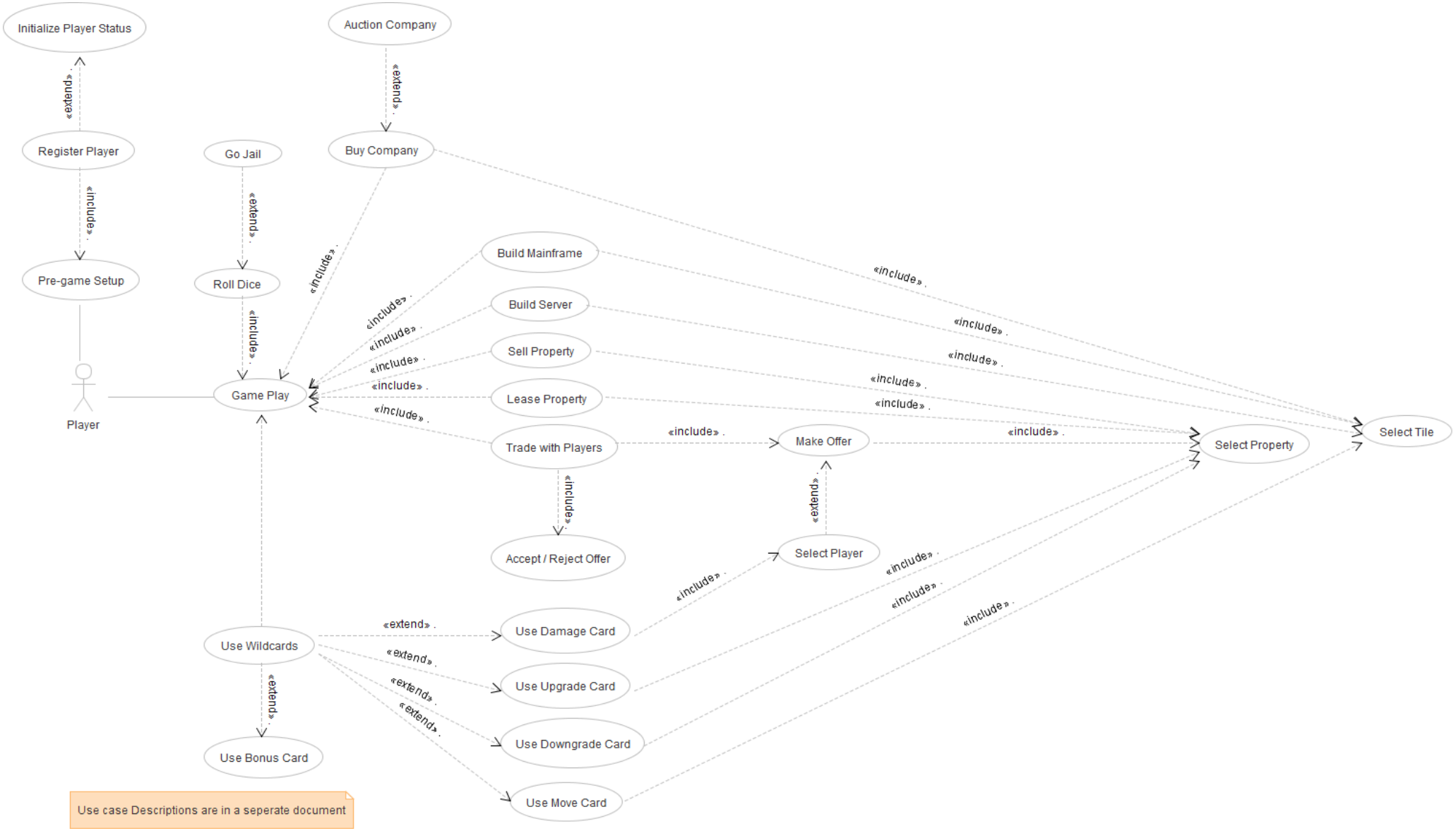
Flow of Events for the *Use Upgrade Card* use-case	
Objective	To allow the player to build a server or mainframe for free.
Pre-condition	The player selected “Use Upgrade Card” in the wildcard’s selection menu. The player has an Upgrade Card.
Main Flow	1. The system list every tile the player owns, asking the player to select.
	2. The server numbers on the tile is added by one. If there are already 4 servers, then

	it is replaced by a mainframe.
Alternative Flow	At 1. If the player owns no tile, the system displays a warning with nothing changed. The card is wasted
	At 2. If there is already a mainframe on the tile, the system displays a warning with nothing changed. The card is wasted.
Post-condition	The player gained a server or mainframe for free. Player will return to the game-play menu.

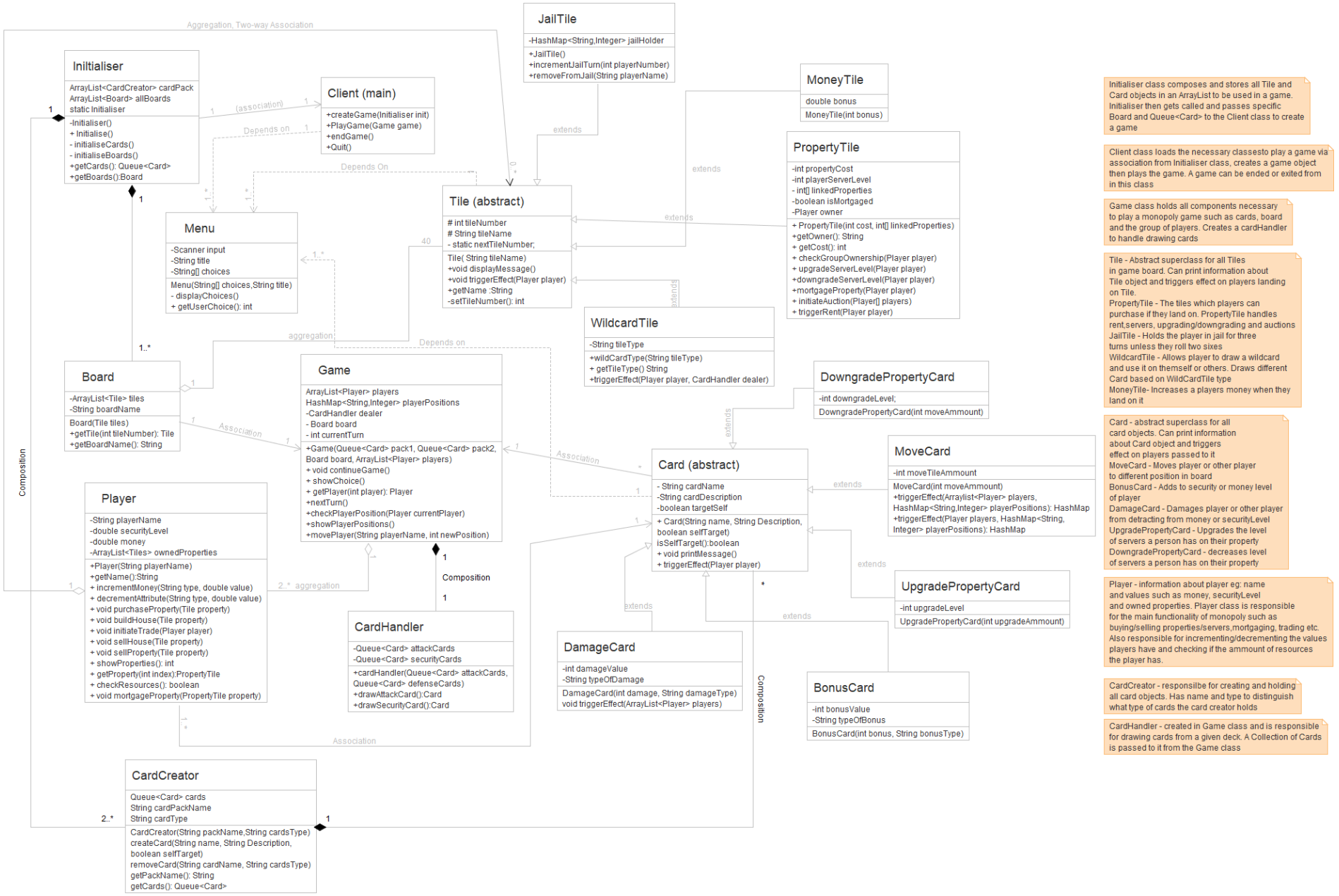
Flow of Events for the *Use Downgrade Card* use-case	
Objective	To allow the player to attack another player by downgrading their server or mainframe.
Pre-condition	The player selected “Use Downgrade Card” in the wildcards selection menu The player has a Downgrade Card.
Main Flow	1. The system list every other player, asking the player to select a target.
	2. If the target player has a mainframe, it is downgraded to four servers. Otherwise, the target player will lose one server.
Post-condition	The target player loses money and security value. Player will return to the game-play menu.

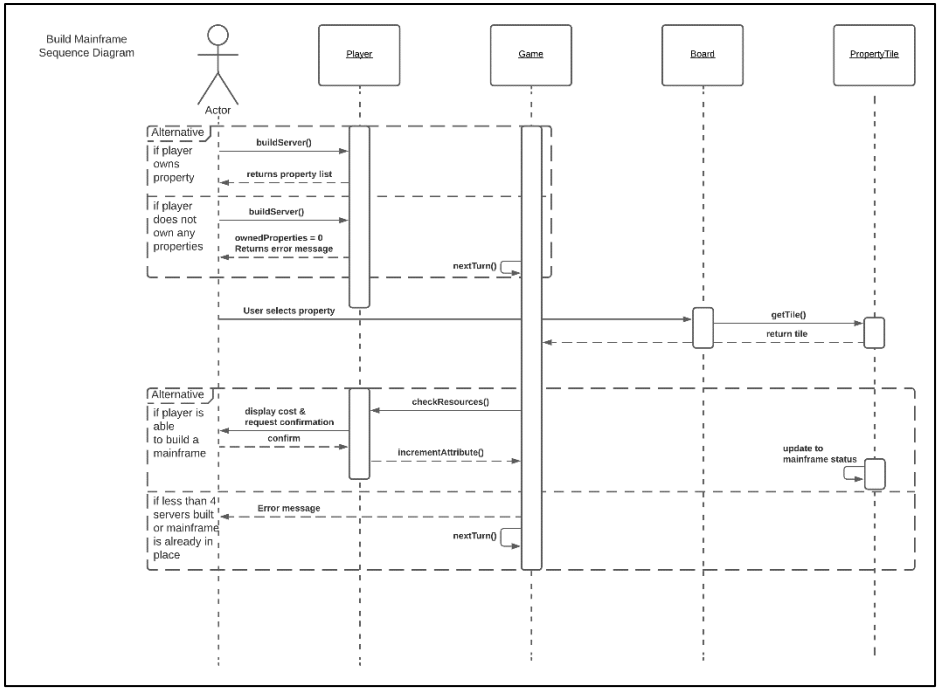
Flow of Events for the *Use Move Card* use-case	
Objective	To allow the player jump to any tiles.
Pre-condition	The player selected “Use Move Card” in the wildcard’s selection menu. The player has a Move Card.
Main Flow	1. The system list every tile on the map, asking the player to select.
	2. The system display a menu of action the player can perform on the tile. The list is the same as if the player as rolled a dice and landed on the tile.
Post-condition	The player is move to the selected tile. The player is ready for actions on the tile.

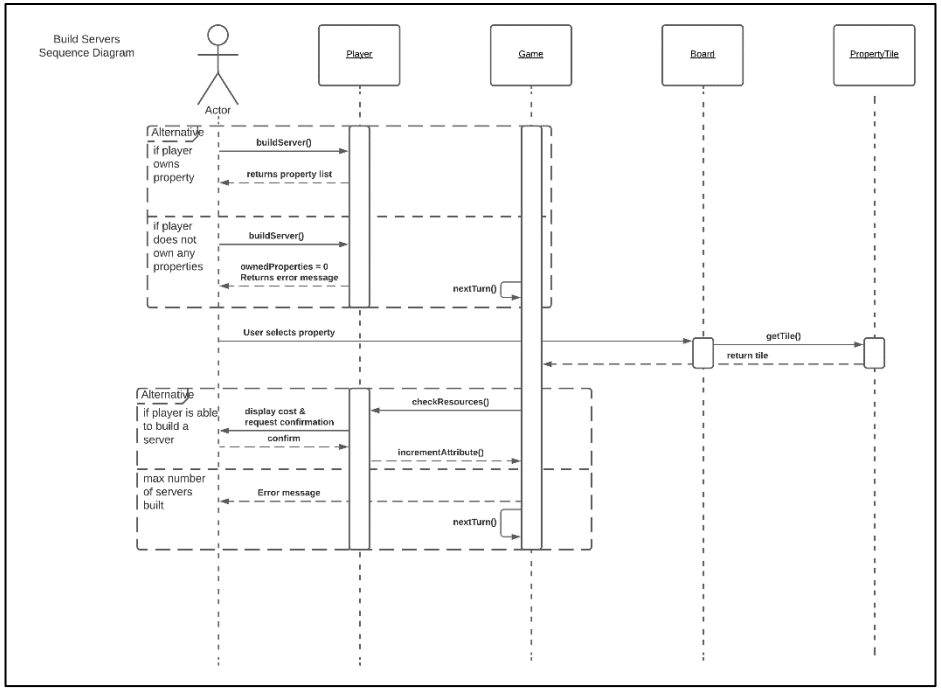
Use Case Diagram

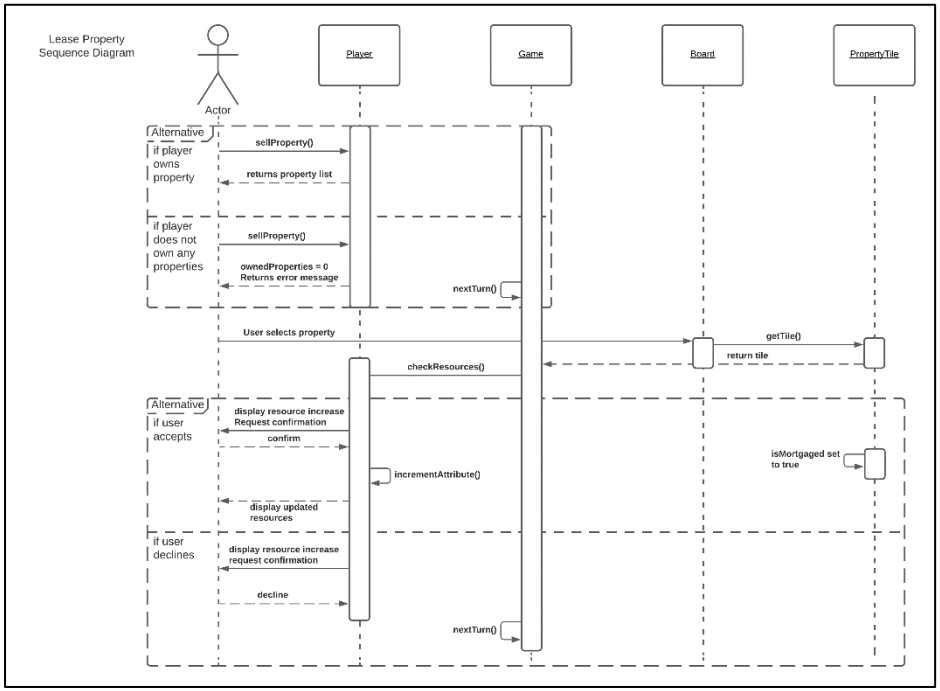


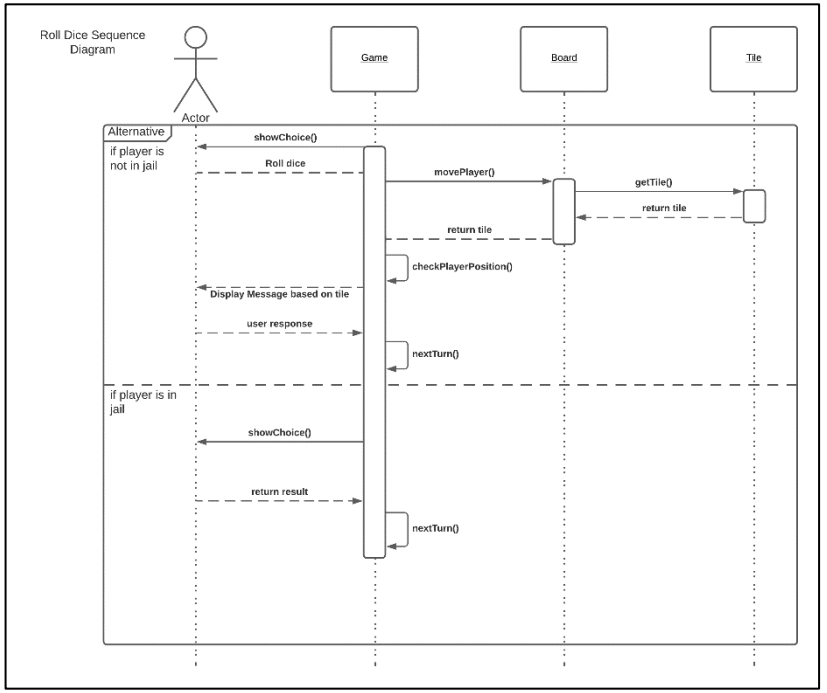
Class Diagram

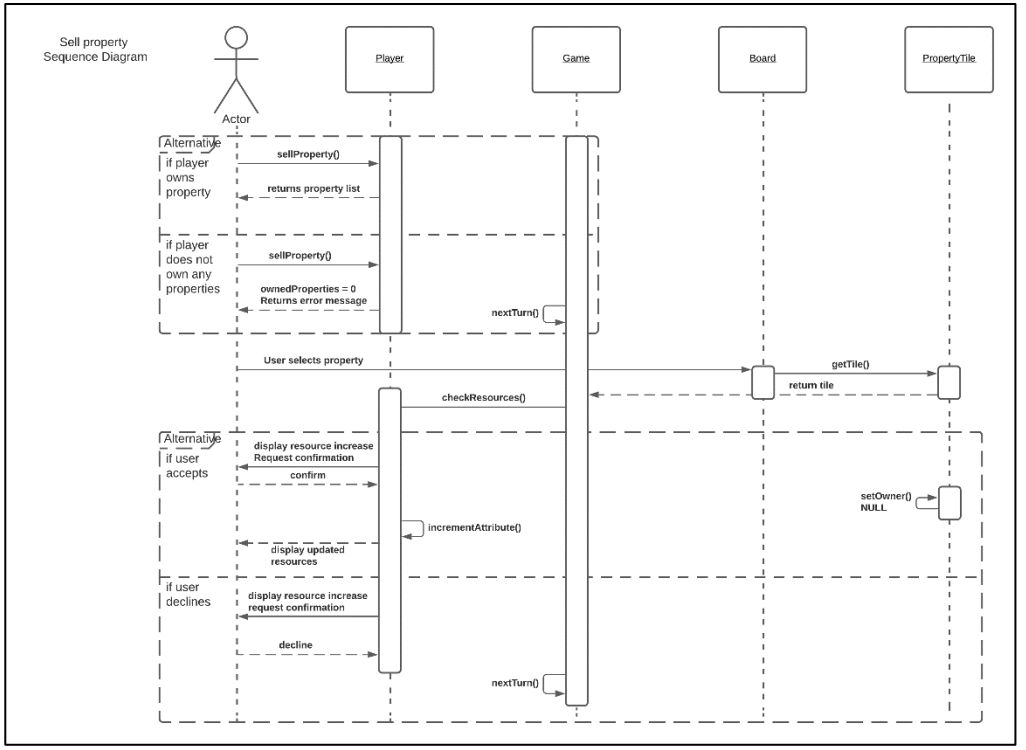


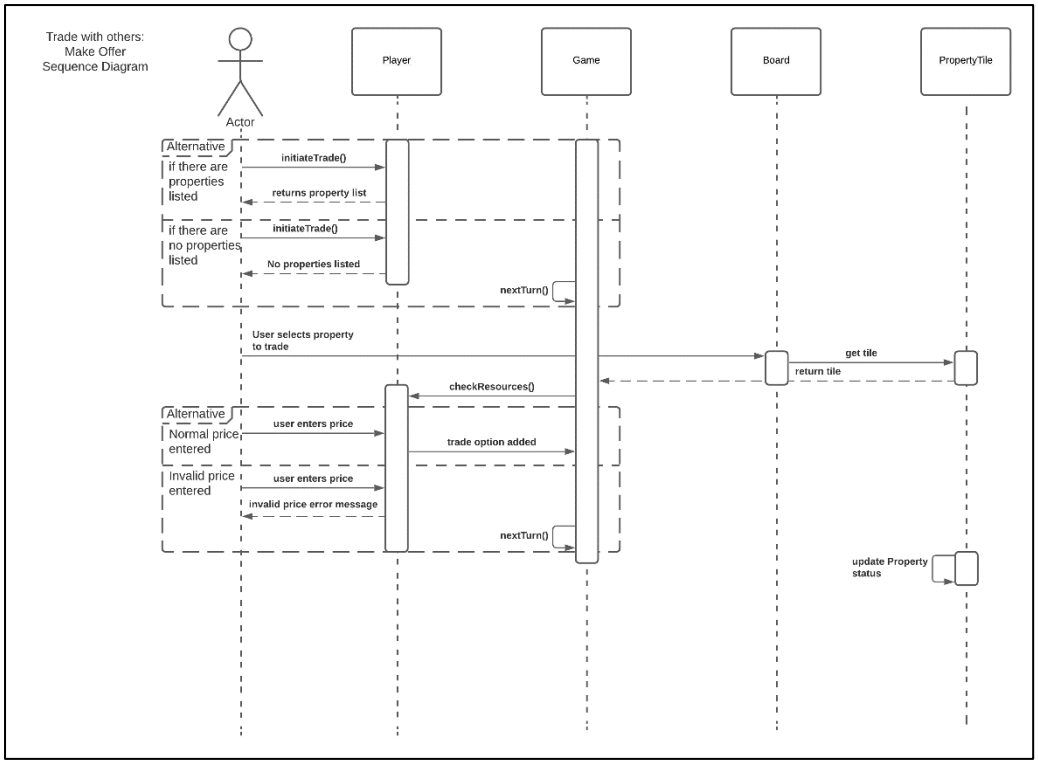

Build Mainframe
Sequence Diagram
Actor
Player
Game
Board
PropertyTile
Alternative
if player
owns
property
buildServer()
returns property list
if player
does not
own any
properties
buildServer()
ownedProperties = 0
Returns error message
nextTurn()
User selects property
getTile()
return tile
Alternative
if player is
able
to build a
mainframe
checkResources()
display cost &
request confirmation
confirm
incrementAttribute()
update to
mainframe status
if less than 4
servers built
or mainframe
is already in
place
Error message
nextTurn()

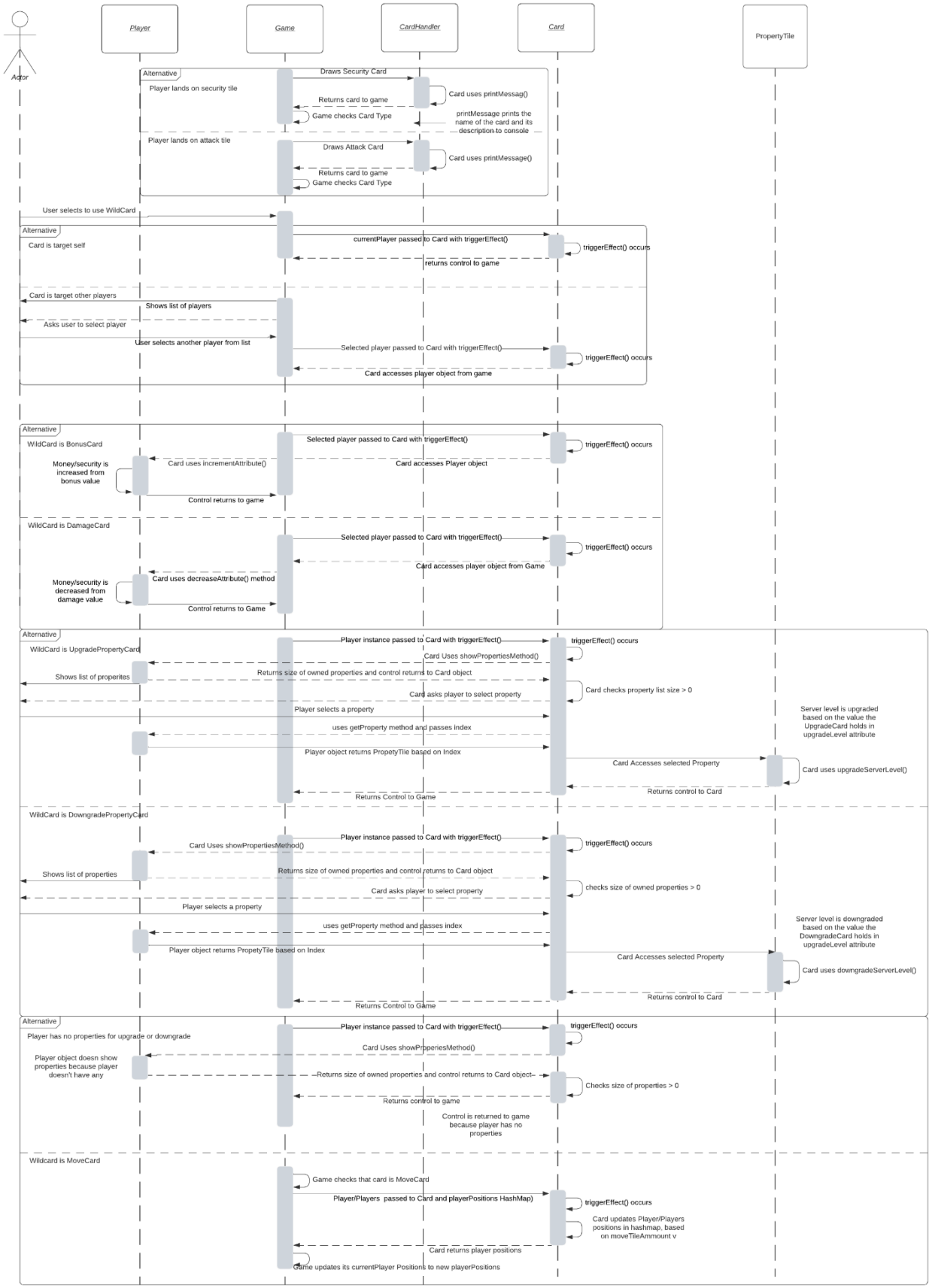

Build Servers
Sequence Diagram
Actor
Player
Game
Board
PropertyTile
Alternative
if player
owns
property
buildServer()
returns property list
if player
does not
own any
properties
buildServer()
ownedProperties = 0
Returns error message
nextTurn()
User selects property
getTile()
return tile
Alternative
if player is able
to build a
server
checkResources()
display cost &
request confirmation
confirm
incrementAttribute()
max number
of servers
built
Error message
nextTurn()


Lease Property
Sequence Diagram
Actor
Player
Game
Board
PropertyTile
Alternative
if player
owns
property
sellProperty()
returns property list
if player
does not
own any
properties
sellProperty()
ownedProperties = 0
Returns error message
nextTurn()
User selects property
getTile()
return tile
checkResources()
Alternative
if user
accepts
display resource increase
Request confirmation
confirm
incrementAttribute()
isMortgaged set
to true
display updated
resources
if user
declines
display resource increase
request confirmation
decline
nextTurn()


Roll Dice Sequence
Diagram
Actor
Game
Board
Tile
Alternative
if player is
not in jail
showChoice()
Roll dice
movePlayer()
getTile()
return tile
return tile
checkPlayerPosition()
Display Message based on tile
user response
nextTurn()
if player is in
jail
showChoice()
return result
nextTurn()


Sell property
Sequence Diagram
Actor
Player
Game
Board
PropertyTile
Alternative
if player owns property
sellProperty()
returns property list
if player does not own any properties
sellProperty()
ownedProperties = 0
Returns error message
nextTurn()
User selects property
getTile()
return tile
checkResources()
Alternative
if user accepts
display resource increase
Request confirmation
confirm
setOwner()
NULL
incrementAttribute()
display updated resources
if user declines
display resource increase
request confirmation
decline
nextTurn()


Trade with others:
Make Offer
Sequence Diagram
Actor
Player
Game
Board
PropertyTile
Alternative
if there are properties listed
initiateTrade()
returns property list
if there are no properties listed
initiateTrade()
No properties listed
nextTurn()
User selects property to trade
get tile
return tile
checkResources()
Alternative
Normal price entered
user enters price
trade option added
Invalid price entered
user enters price
invalid price error message
nextTurn()
update Property status


Actor
Player
Game
CardHandler
Card
PropertyTile
Alternative
Player lands on security tile
Draws Security Card
Card uses printMessag()
Returns card to game
Game checks Card Type
printMessage prints the name of the card and its description to console
Player lands on attack tile
Draws Attack Card
Card uses printMessage()
Returns card to game
Game checks Card Type
User selects to use WildCard
Alternative
Card is target self
currentPlayer passed to Card with triggerEffect()
triggerEffect() occurs
returns control to game
Card is target other players
Shows list of players
Asks user to select player
User selects another player from list
Selected player passed to Card with triggerEffect()
triggerEffect() occurs
Card accesses player object from game
Alternative
WildCard is BonusCard
Selected player passed to Card with triggerEffect()
triggerEffect() occurs
Money/security is increased from bonus value
Card uses incrementAttribute()
Card accesses Player object
Control returns to game
WildCard is DamageCard
Selected player passed to Card with triggerEffect()
triggerEffect() occurs
Card accesses player object from Game
Money/security is decreased from damage value
Card uses decreaseAttribute() method
Control returns to Game
Alternative
WildCard is UpgradePropertyCard
Player instance passed to Card with triggerEffect()
triggerEffect() occurs
Card Uses showPropertiesMethod()
Shows list of properites
Returns size of owned properties and control returns to Card object
Card checks property list size > 0
Card asks player to select property
Player selects a property
Server level is upgraded based on the value the UpgradeCard holds in upgradeLevel attribute
uses getProperty method and passes index
Player object returns PropetyTile based on Index
Card Accesses selected Property
Card uses upgradeServerLevel()
Returns control to Card
Returns Control to Game
WildCard is DowngradePropertyCard
Player instance passed to Card with triggerEffect()
triggerEffect() occurs
Card Uses showPropertiesMethod()
Shows list of properties
Returns size of owned properties and control returns to Card object
checks size of owned properties > 0
Card asks player to select property
Player selects a property
Server level is downgraded based on the value the DowngradeCard holds in upgradeLevel attribute
uses getProperty method and passes index
Player object returns PropetyTile based on Index
Card Accesses selected Property
Card uses downgradeServerLevel()
Returns control to Card
Returns Control to Game
Alternative
Player has no properties for upgrade or downgrade
Player instance passed to Card with triggerEffect()
triggerEffect() occurs
Card Uses showPropertiesMethod()
Player object doesn show properties because player doesn't have any
Returns size of owned properties and control returns to Card object
Checks size of properties > 0
Returns control to game
Control is returned to game because player has no properties
Wildcard is MoveCard
Game checks that card is MoveCard
Player/Players passed to Card and playerPositions HashMap)
triggerEffect() occurs
Card updates Player/Players positions in hashmap, based on moveTileAmmount v
Card returns player positions
Game updates its currentPlayer Positions to new playerPositions

Draft Game Layout

20 PAID PARKING	21 COMPANY TILES	22 COMPANY TILES	23 ROB BANK	24 COMPANY TILES	25 OTHER PROPERTY	26 COMPANY TILES	27 SECURITY	28 COMPANY TILES	29 COMPANY TILES	30 JAIL	31 COMPANY TILES	32 DATA BREACH	33 COMPANY TILES	34 ATTACK!
19 UNIQUE PROPERTY														35 OTHER PROPERTY
18 COMPANY TILES														36 COMPANY TILES
17 ATTACK!														37 UNIQUE PROPERTY
16 COMPANY TILES														38 COMPANY TILES
15 OTHER PROPERTY														39 COMPANY TILES
14 SECURITY	13 COMPANY TILES	12 COMPANY TILES	11 COMPANY TILES	10 GO TO JAIL	9 COMPANY TILES	8 COMPANY TILES	7 SECURITY	6 COMPANY TILES	5 OTHER PROPERTY	4 COMPANY TILES	3 ATTACK!	2 COMPANY TILES	1 ROB BANK	0 START / FREE PARKING

A © K T O P O L Y

WILD  **CARD**