Bilkent University

Department of Computer Engineering

CS 319 Course Project

Group: 2C

Analysis Report

- Furkan Kazım AKKURT
- Abdullah Can ALPAY
- Murat ANGIN
- Ümit Yiğit BAŞARAN
- Muhammed Emre YILDIZ

Instructor: Eray TÜZÜN

Teaching Assistant(s): Barış Ardıç, Elgun Jabrayilzade, Emre Sülün

Analysis Report

Project short-name: Monopoly Project

Introduction

A monopoly is a strategy-based board game presented by Hasbro. The purpose of the game is to become the player with the most money by buying properties, collecting rents, and selling the properties while trying other players to declare bankruptcy. The game is played by 2-8 people. Each player starts with $1500, and they roll dice at their turns. Based on the value of the dice, players move on the board and can allocate spaces by buying them. There are also different kinds of squares such as jail, chance, and community chest. When a player visits jail, they should choose one of the two options; either getting out from the jail using a jail card which can be retrieved from the chance cards, or skipping their turns for 2 consecutive turns. When a player passes through the starting point, they receive $200. When a player owns all the properties in the same color, they are allowed to buy houses from the bank, in which the amount of rent other players have to pay increases. Hotels are allowed to be bought when a player has 4 houses on each property in the same color group. The game-winner is determined after all the players declare bankruptcy except one player, in which the winner is counted as the player that did not declare bankruptcy. There are different versions of the Monopoly game, but in this report, the standard version is going to be explained.

Current System

In the current system of the original game which is published by Hasbro, there are a board, 2 dice, tokens, 32 houses, 12 hostels, 16 chance cards, 16 community chest cards, and monopoly money worth 20.580 units.

The Board

The monopoly board is where the gameplay takes place. (Figure 1)

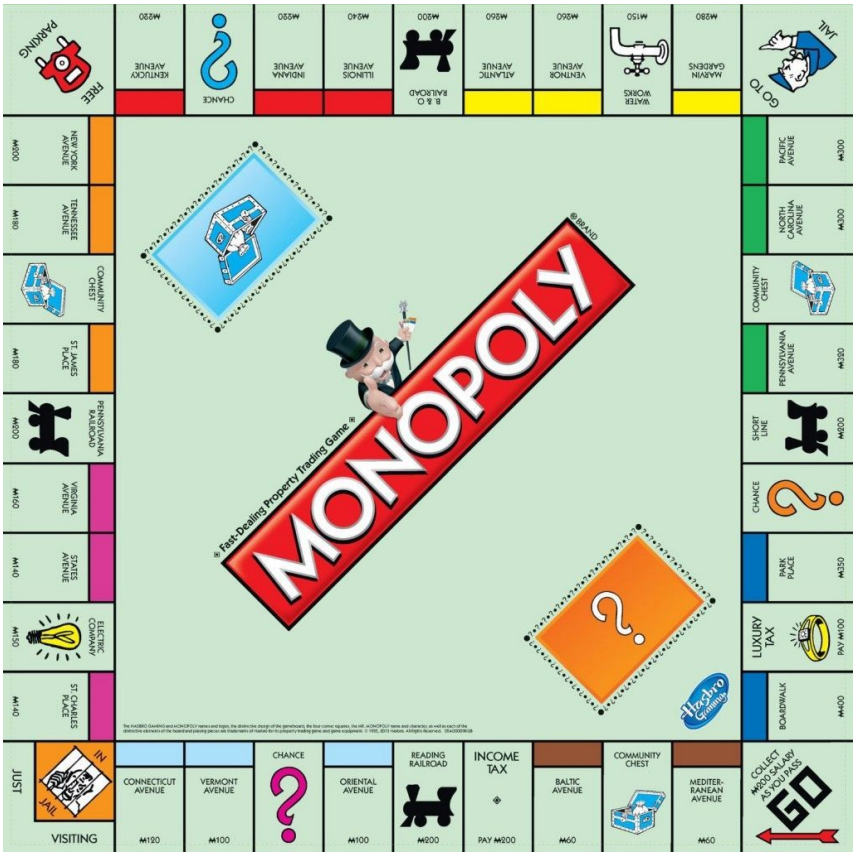

Figure 1: The Official Monopoly Board

Board has some subcomponents such as starting point, properties, chance and chest spaces, free parking space, jail space, go to jail space, and tax spaces.

Starting Point (Go)

Go is the first tile of the board and every player has to start from this tile (Figure 2) at the beginning of the game. After the beginning whenever a player passes to this tile bank should give this player 200 monopoly units.

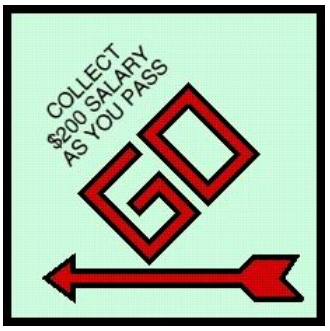

Figure 2: Starting Point (Go) Square

Properties

Because the main objective of the game is to bankrupt all opponents, players should have properties to get other opponent's money. "Properties may be bought in one of 3 ways: landing on the property space and buying it, being the highest bidder in an auction for property, or buying it from an opponent in a trade. Properties may also be received from bankrupted players, provided the Bank didn't bankrupt them".

Whenever a player buys or takes ownership of some property, s/he needs to get title deeds (Figure 3), which has all the information about the bought property.

Figure 3: Title deed example

There are **groups that are labeled with eight different colors, four railroads, and utilities such as Electric Company and Water Works** on the board.

Brown Property Set

These properties are placed at the beginning of the game. They are the cheapest properties of the game. In the standard version of the Monopoly Mediterranean Avenue (60 Unit) and Baltic Avenue (60 Unit) are the members of this set (Figure 4).

Figure 4: Brown Property Set Title Deeds

Light Blue Property Set

Although this property set is more valuable than brown ones, still they are one the cheapest property sets of the game. They are placed right after the brown property set. In the standard version of the Monopoly Oriental Avenue (100 Unit), Vermont Avenue (100 Unit) and Connecticut Avenue (120 Unit) are the members of this set (Figure 5).

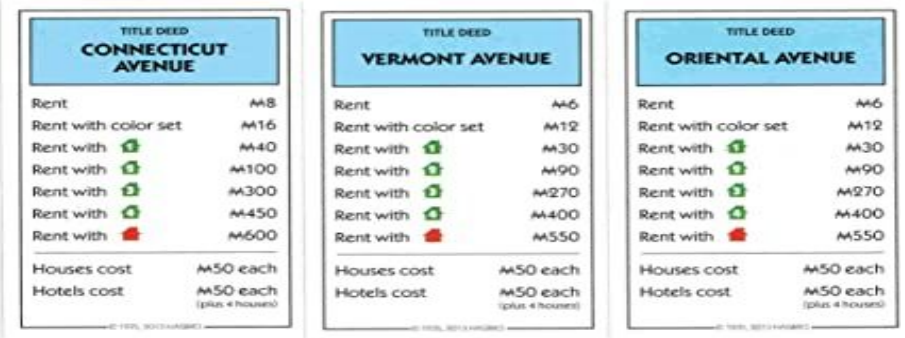

Figure 5: Light Blue Property Set Title Deeds

Pink Property Set

They are more valuable than light blue ones but less valuable than orange ones. They are placed at the beginning of the second side of the board. In the standard version of Monopoly St. Charles Place (140 Unit), States Avenue (140 Unit) and Virginia Avenue (160 Unit) are the members of this set (Figure 6).

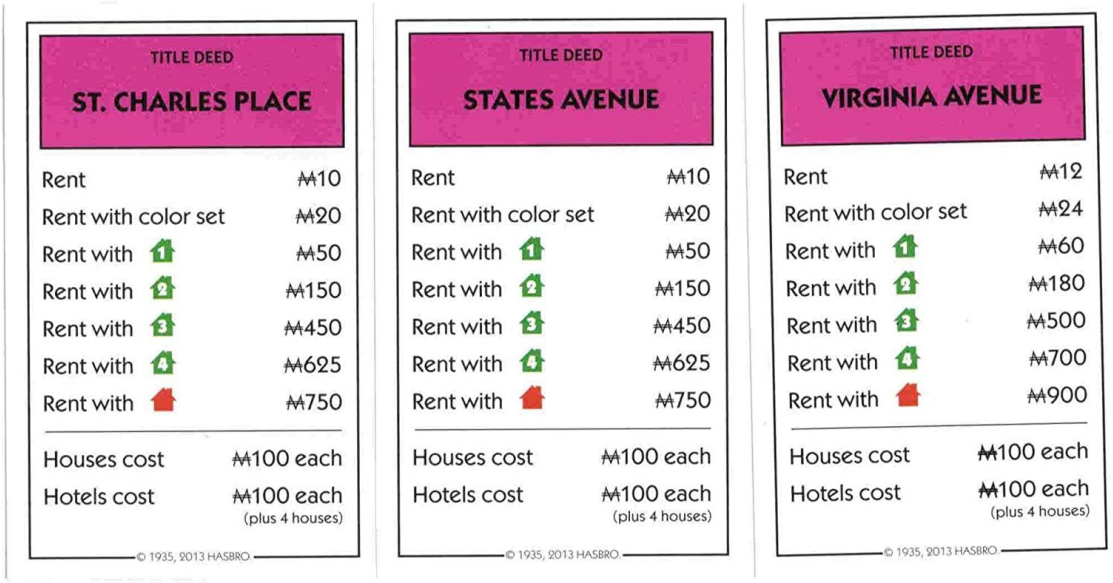

Figure 6: Pink Property Set Title Deeds

Orange Property Set

They are the most valuable properties of the second side of the board. They are placed after the pink property set. In the standard version of Monopoly St. James Place (180 Unit), Tennessee Avenue (180 Unit) and New York Avenue (200 Unit) are the members of this set (Figure 7).

Figure 7: Orange Property Set Title Deeds

Red Property Set

Red property set is more valuable than the orange ones but properties inside the red group are the least valuable ones of the third side of the board. They are placed at the beginning of the third side. In the standard version of Monopoly Kentucky Avenue (220 Unit), Indiana Avenue (220 Unit) and Illinois Avenue (240 Unit) are the members of this set (Figure 8).

Figure 8: Red Property Set Title Deeds

Yellow Property Set

They are the most valuable properties of the third side of the board. They are placed at the end of the third place. In the standard version of Monopoly Atlantic Avenue (260 Unit), Ventnor Avenue (260 Unit) and Marvin Gardens (280 Unit) are the members of this set (Figure 9).

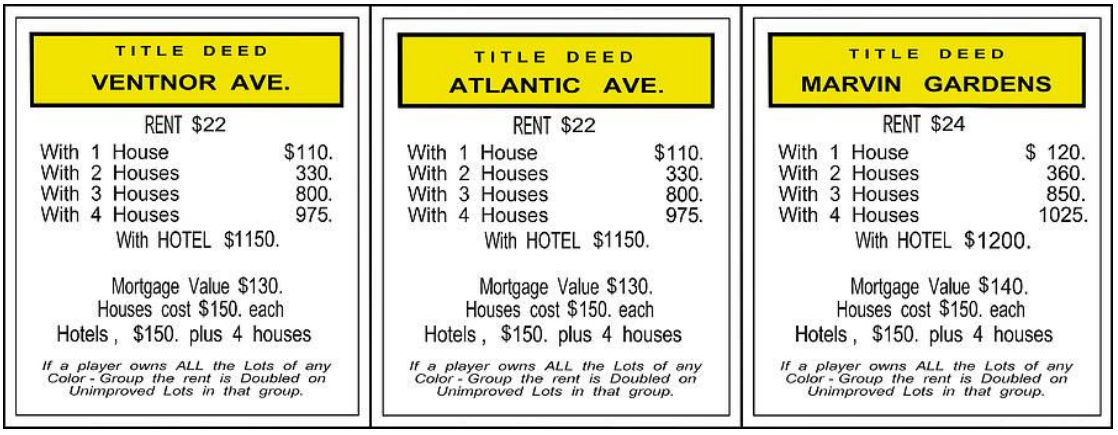

Figure 9: Yellow Property Set Title Deeds

Green Property Set

These properties are in the top two of the game. They are placed at the beginning of the fourth and the last side of the board. In the standard version of Monopoly Pacific Avenue (300 Unit), North Carolina Avenue (300 Unit) and Pennsylvania Avenue (320 Unit) are the members of this set (Figure 10).

Figure 10: Green Property Set Title Deeds

Blue Property Set

This set is the most valuable set of the game. They are placed in the last piece of the last side. In the standard version of Monopoly Park Place (350 Unit) and Boardwalk (400 Unit) are the members of this set (Figure 11).

Figure 11: Blue Property Set Title Deeds

The Four Railroads

The railroads are one of the properties. They are more profitable than other properties. Reading Railroad, Pennsylvania Railroad, B. & O. Railroad, and Short Line are the railroads of the standard Monopoly game (Figure 12).

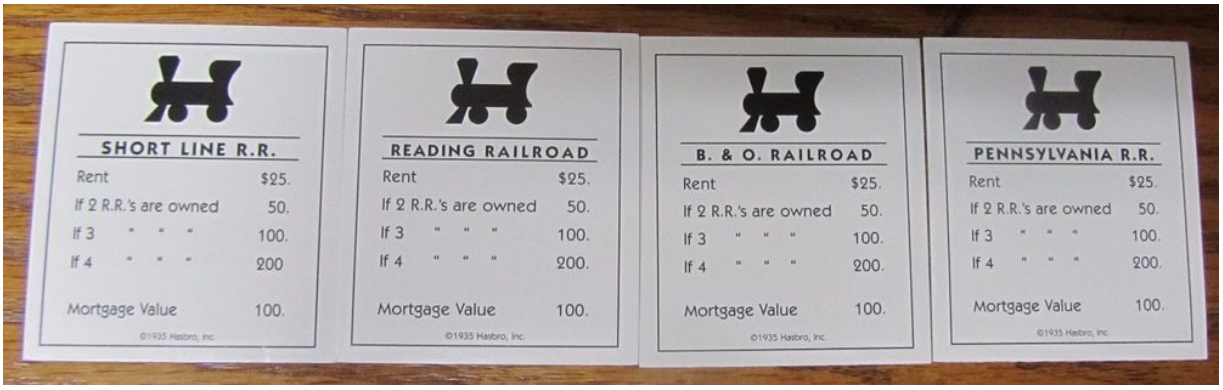

Figure 12: Railroads in the Standard Version of Monopoly

Electric Company

It is one of the cheapest properties. Also, buildings can't be built in this place. It is placed between St. Charles Place and State Avenue properties in the standard version of Monopoly.

Figure 13: Electric Company

WaterWorks

It has the same value as the Electric Company property. It is located between Ventnor Avenue and Marvin Gardens properties in the standard version of Monopoly.

Figure 14: Water Works

Free Parking

Figure 15: Free Parking

When a player comes to Free Parking, nothing changes. He/She only parks on the ground and waits for the next round.

Jail

Figure 16: Jail

One of the most important parts of the game is going to jail. In some game genres, this part may be called prison. Jail is usually a bad thing. Because when you go to jail, you cannot play 3 rounds. In some cases, waiting in jail may give you an advantage. Example: If your opponent has many properties in front of you and you are likely to fall into one of them, you can expect the opponent to pay you 3 rounds of rent instead of leaving the jail.

To go to jail as follows:

1. If the number of dice you roll coincides with the jail tile, you will remain in jail.
2. If you withdraw the go to jail card from the chance cards, you will go to jail.
3. If you roll double dice 3 times in a row in the same round, you will go to jail.

Possibilities to exit jail:

1. When you wait for three rounds, you have the right to play the next round and you exit the jail.
2. If you have a "Get Out of Code" card that you won before, you can use it to exit.
3. You can get out of jail by paying a certain amount of money. You can pay to get out of jail, especially if you are early in the game. Because it is very important to buy property at the beginning of the game. Waiting for you while other players buy 3 rounds of properties may cause you to lag behind the game.

Visitor

If you have not been sent to the Jail and come to this section while the game is in progress, you will be counted as a visitor and will not be penalized. When it is your turn, you continue playing.

One of the most important parts of the game is going to jail. In some game genres, this part may be called prison. Jail is usually a bad thing. Because when you go to jail, you cannot play 3 rounds. In some cases, waiting in jail may give you an advantage. Example: If your opponent has many properties in front of you and you are likely to fall into one of them, you can expect the opponent to pay you 3 rounds of rent instead of leaving the jail.

To go to jail as follows:

4. If the number of dice you roll coincides with the jail tile, you will remain in jail.

5. If you withdraw the go to jail card from the chance cards, you will go to jail.
6. If you roll double dice 3 times in a row in the same round, you will go to jail.

Possibilities to exit jail:

4. When you wait for three rounds, you have the right to play the next round and you exit the jail.
5. If you have a "Get Out of Code" card that you won before, you can use it to exit.
6. You can get out of jail by paying a certain amount of money. You can pay to get out of jail, especially if you are early in the game. Because it is very important to buy property at the beginning of the game. Waiting for you while other players buy 3 rounds of properties may cause you to lag behind the game.

Visitor

If you have not been sent to the Jail and come to this section while the game is in progress, you will be counted as a visitor and will not be penalized. When it is your turn, you continue playing.

Go to Jail

Figure 17: Go to Jail

Whenever a player is placed in this space then s/he needs to go to the Jail space of the game. This space is placed in the third corner of the board in the standard version of the Monopoly.

Tax

This is the space that if you are placed in this space then you have to pay a predefined tax. Income Tax and Luxury Tax are the tax spaces that are in the standard version of the Monopoly.

Figure 18: Income and Luxury Taxes

Chance and Chest Spaces

Whenever a player comes over these spaces, then, s/he needs to take a card from either the community chest card set or chance card set.

Figure 19: Chance and Chest Spaces

How to play monopoly

In turn, all players roll two dice together. The player with the highest roll starts the game and the game continues clockwise.

When it's your turn, roll both dice and move your pawn clockwise as much as the dice you rolled. Two or more pawns can stand on the same digit. Depending on the house where your pawn is, you must do one of the following:

1. Buying plots and other property (if not owned by another player).

2. Asking the banker to auction a property (if you do not want to buy it).
3. Paying rent (if another player has bought that property).
4. Paying taxes.
5. Taking a Chance or Public Fund card.
6. Entering the Jail.

If you own the title deeds of all plots in a color group, you can build houses and hotels on these plots.

If you run out of money and have to pay a loan, you can mortgage or sell your properties to the bank. If you owe more debt to the bank or another player than you can pay by selling your goods, you will go bankrupt and withdraw from the game.

Players cannot borrow money from each other or lend money to each other. However, if a player wishes, he can agree to take a property from another player instead of money for his loan.

If you roll double dice (that is, the same number comes on both dice you roll), move your pawn forward as usual and move according to the point on which you stand. After doing what you have to do, roll the dice again and continue the game. If you roll double dice three times in a row, on the third time you will enter the Jail and cannot advance your pawn.

The last player remaining in the game without going bankrupt wins the game.

Buying Property

There are three types of property: Properties, Railroads, Utilities.

If you stand on unclaimed property, you can buy it. If you decide to buy the property, you must pay the price written on this household to the bank. The bank will give you the Title Deed card as proof of your ownership.

Owning property allows you to receive rent from all players who have paused on your property. It is an advantage to have all the plots in a color group, i.e. a "Monopoly". In this case, you can get more rental income by establishing a house or hotel on these plots.

Pay Rent

If you stand on an owned property, you have to pay the property owner. If the property is mortgaged, the rent is not paid.

The owner must claim the rent before the next player rolls the dice. The rental fee is written on the Title Deed Card of that property and changes according to the number of buildings on the property.

If you own the title deeds of all plots in a color group, the rental prices of all vacant plots in that color group (i.e. plots without houses or hotels) doubles. Even if your properties in a color group are mortgaged, you can get twice the rent for unencumbered ones.

Utilities

Utilities can be bought and auctioned like any other property. If you stop on an owned Utility, you will pay rent based on the damage you've inflicted to get to that house.

If the utility owner has only one, the rent is 400 times the dice you rolled.

If the utility owner has both, the rent will be 1,000 times the dice you rolled.

Railroads

Railroads are bought and auctioned like any other property. If the railroads you are stopping are owned, you must pay the amount indicated on the Deed Card.

The amount to be paid is specified on the Deed Card and depends on how many Railroads cards that player has in total.

Building a House

When you have all the plots in a color group, you can buy a house (or houses) and install them on any of these plots. The price of a house is stated on the Title Deed card. While it's your turn, you can buy houses (or hotels) between rows of other players; however, you need to build a house in a balanced way.

You cannot build your second house on any plot without installing a house on each plot of the same color group. As long as you can afford to pay, you can buy as many buildings as you want from the bank!

If any plot of a color group is mortgaged, you cannot build a house in that group. You need to place your houses and hotels on the property you want to build.

Setting Up a Hotel

In order to buy a hotel, you must have four houses on each plot of a color group. In order to buy a hotel, you have to give the four houses on your property to the bank

and also pay the amount specified in the Deed Certificate. You can only set up one hotel on a Property.

Buying a Building

If the bank has no houses to sell, you must wait for other players to return or sell houses to be able to buy a house.

If there are a limited number of houses or hotels left and two or more players want to buy more buildings than the number of buildings held by the bank, the banker will auction the houses or hotels, starting from the lowest amount stated on the Deed Cards in question. The auction is held individually for each building, and the highest bidder buys the building.

Money Trouble

If you don't have enough money left, you can do the following to raise funds:

1. Sell the building
2. Mortgage a property
3. Selling properties, railroads or utilities (even if it is mortgaged) to another player for an agreed price.

Property Sale

You can sell vacant lots, railroads, and utilities to another player at a mutually agreed price. However, if any of the plots belonging to a particular color group have a building, you cannot sell any plot of that color group to another player. First of all, you have to sell the buildings on these plots to the bank.

Houses and hotels are sold for half of the amount purchased to the bank (shown on the Title Deed Card). You can sell your buildings while it's your turn or between the other players' turn.

Selling house

You must sell the houses the same way they purchase them. So if you are going to sell more than one house to raise funds, all of these houses should not be on the same plot.

Selling hotel

While the hotel is being sold; the bank pays the player half of the hotel price and half of the four house prices the player gave to the bank to buy a hotel. If necessary, funds can be created by turning hotels into homes. In this case, you sell a hotel to the bank and buy four houses, along with half the hotel price.

Sale of a mortgaged property

A mortgaged property can be sold to another player at a mutually agreed price. The new owner can immediately remove the mortgage by paying the mortgage amount and 10% interest. If he decides not to remove the mortgage immediately, he must pay the 10% interest immediately. In this case, when he decides to cancel the mortgage, he pays 10% interest once again in addition to the mortgage amount.

Bankruptcy

If you have more debt than you can pay by selling your goods, you will go bankrupt and withdraw from the game.

If you owe a bank:

Give all your Deed Cards to the bank. The bank auctions them one by one and sells them to the highest bidder (s). If you have it, you must also put your "Get out of Jail for free" card at the bottom of the relevant deck.

If you owe another player

If any, sell all your houses and hotels for half of the amount you bought to the bank (shown on the Title Deed Card). Transfer all your money, Title Deed cards, and, if any, your "exit Jail for free" card to this player.

Chance and Community Chest

The player standing in one of these squares draws the card from the top of the respective deck and applies what is indicated on the card. After following the instructions written on the card, you should place the card at the bottom of the deck, turning the printed side down.

You can just keep the "Quit Jail for free" card until you need it or sell it to another player.

If the card you draw tells you to move to another digit, move in the direction of the arrow. If you pass the start, get the start fee. If the card you withdraw sends you to

jail or tells you to go back to a house without going through the Starting point, you will not be able to get the start money.

Proposed System

Overview

In the current version of the Monopoly game, there are two types of cards, community chest, and chance cards, that can add fun to the flow of the game. Also, the flow of the game starts to repeat itself after some point, players roll dice, buy properties, build houses and hotels, and etc. Therefore, we plan to not bore the players by exposing them to the boring flow of the game. In order to do that, we added 7 special characters to the game, in which each of them has their own strongness and weaknesses. In addition to the special characters, there is also a quest feature in the game, in which the players are to complete the quests and earn rewards depending on the difficulty of the game. The proposed system is not just limited to these features only. There are also special characters and buildings added to the game so that the flow of the game is going to be changed. Within the system that is proposed, we plan to make the game more enjoyable, fun, and not boring.

Functional Requirements

Special Characters

Fugitive

The player can escape from jail with a fugitive character at most two times. When a player escapes from jail the player can go anywhere.

Traveler

The player with a traveler character can go any location if the player arrives at one of the travel agencies.

Driver

The player with a driver character can go forward or backward.

Additional Special Cards

Natural Disaster Cards

Natural Disaster Cards can be used for reducing the profit of any land.

Profit Cards

Profit Cards can be used for increasing the profit of any land.

Special Quests

Seven Wonders

One other way to win the game is, if the player buys all the Seven Wonders' land the player will win the game.

Downtown Roll

If the player arrives at selected locations with specific dice combinations, they earn extra money.

Non-functional Requirements

Usability

The game doesn’t require an installation process, or an installer. There is only one executable and users can enter the game directly after running this executable.

The input devices for browsers are mouse and keyboard, and because the selected technology for implementations is a Chromium based technology, it will use the same input devices. There will be no differences on the usual input styles used in browsers such as swiping, dragging and right clicking.

Performance

Our aim in terms of Performance is to reach a stable 60 frame per second gameplay in the whole game. In order to reach it we investigate some times to research different technologies that can be used with Electron. We found 2 technology WebGL

and HTML5 Canvas. In a benchmark which draws random bird textures on screen and moves them randomly, Canvas system gets 30-44 FPS at 512 bird and WebGL gets 60 FPS if you open vsync. When you try to increase the number of birds in benchmark, for example 1024, Canvas reaches limits of itself and gets 5-10 FPS while WebGL is still on 60FPS. Therefore we concluded on a rendering framework built on WebGL framework. It will boost our speed according to benchmarks.

Also we will use websockets for real time communication and it will increase our performance according to benchmarks conducted between HTML requests and websockets. On average, an HTML request took nearly 107ms while a websocket request took 83ms. If we try to increase request count, at 1 seconds HTML can only handle 10 requests while websocket handle nearly 4000 requests. The main reason for this large difference is that the browser limits the number of concurrent HTTP connections (6 by default in Chrome), while there is no limitation how many messages a websocket connection can send or receive.

Platforms

Windows	95.65%	+0.04%
Windows 10 64 bit	89.28%	+0.20%
Windows 7 64 bit	4.39%	-0.09%
Windows 8.1 64 bit	1.46%	-0.06%
Windows 7	0.22%	-0.01%
Windows 10	0.13%	0.00%
Windows 8 64 bit	0.11%	0.00%
OSX	3.45%	-0.01%
MacOS 10.15.6 64 bit	0.86%	-0.56%
MacOS 10.15.7 64 bit	0.85%	+0.85%
MacOS 10.14.6 64 bit	0.51%	-0.07%
MacOS 10.13.6 64 bit	0.40%	-0.01%
MacOS 10.15.5 64 bit	0.18%	-0.14%
MacOS 10.12.6 64 bit	0.14%	0.00%
MacOS 10.15.4 64 bit	0.10%	-0.03%
MacOS 10.11.6 64 bit	0.10%	-0.01%
MacOS 10.15.3 64 bit	0.05%	-0.02%
MacOS 10.16.0 64 bit	0.05%	+0.05%
Linux	0.90%	-0.04%
Ubuntu 20.04.1 LTS 64 bit	0.21%	+0.02%
"Manjaro Linux" 64 bit	0.10%	-0.01%
"Arch Linux" 64 bit	0.10%	-0.01%
Ubuntu 18.04.5 LTS 64 bit	0.07%	0.00%
Linux Mint 20 64 bit	0.05%	0.00%

Figure 20: Platform Figure

Since the game will be implemented using the Electron framework of JavaScript, it will actually run embedded into a web browser, by combining the Chromium rendering engine and Node.js runtime. According to Electron's official website[4], the supported platforms section contains all the major OSes, MacOS, Windows and Linux. The version requirements for these OSes in order, 10.10 Yosemite, Windows 7 and Ubuntu 14.04, or Fedora 24. We made some research in order to find usage statistics for OSes below these requirements. We looked at the charts published by Steam, the massieve game seller, producer, publisher company. You can see their statistics below.

As you can see in here, there is no data for OSes not supported in this chart. The scale of these numbers in millions(Steam's total user count is 21 million in September 2020[5]). Therefore we can clearly say that no major problem will happen because of incompatibility on computer platforms in general. Of Course we didn't offer a mobile version therefore we are excluding mobile OSes such as IOS and Android from this conversation.

System Models

Use-Case Models

Player Turn in Monopoly Game Use-Case Model

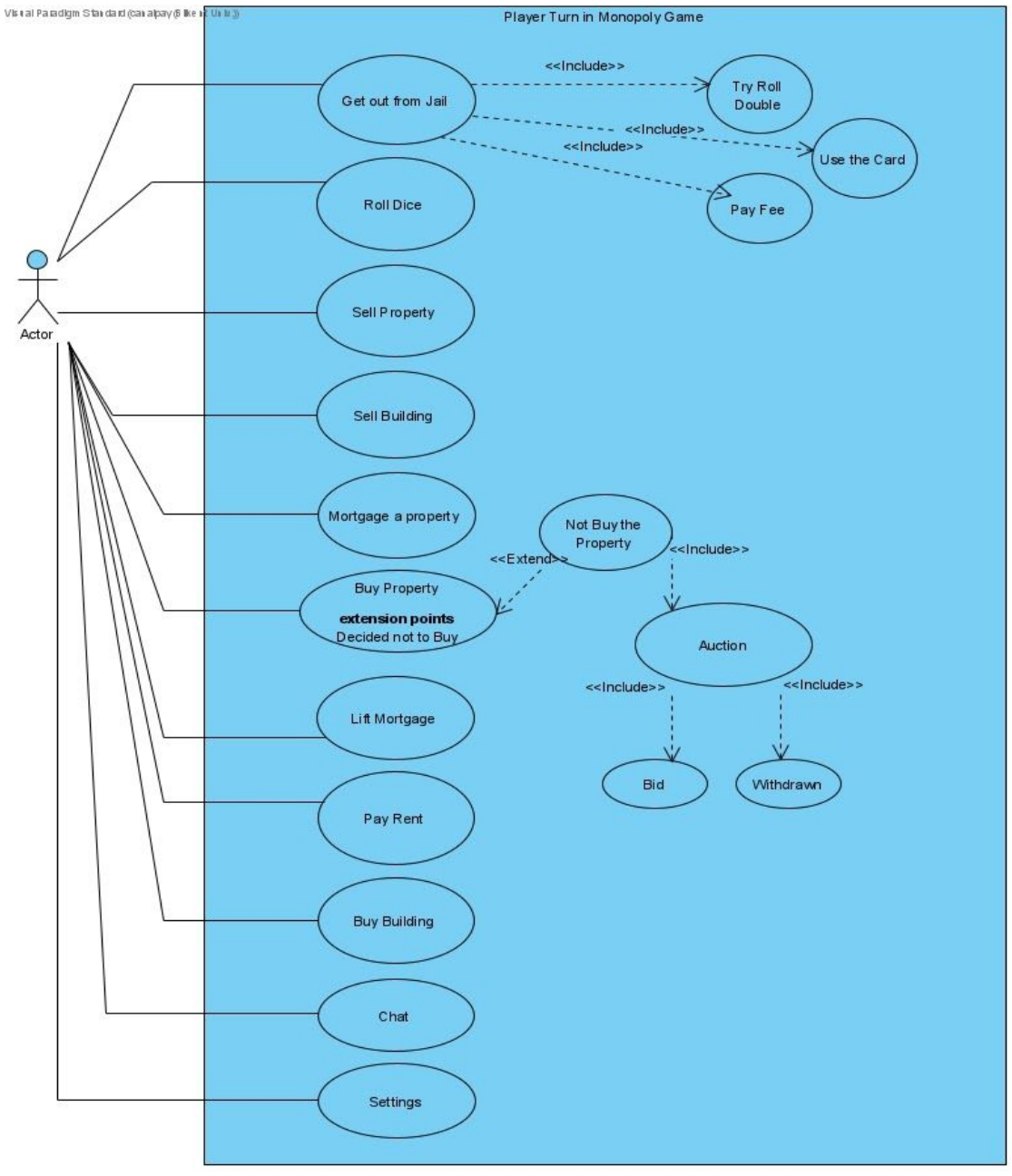

Figure 21: Player Turn in Monopoly Game Use-Case Model

Use Case	Play Turn in Monopoly Game
Actor	Player
Possibilities	1. Roll dice 2. Sell Property 3. Sell Building 4. Buy Building 5. Buy Property 6. Mortgage a Property 7. Lift Mortgage 8. Pay Rent 9. Get out from Jail 10. Chat 11. Settings
Flow of events	In the player turn, if the player is inside jail, the player starts with getting out of jail by doing one of the three options. After that the player can buy, sell properties and buildings and change settings or send messages. The player can need to pay rent. Additionally, the player can mortgage or lift mortgage of properties. If the player does not choose the buy property option, the not buy the property case is occured in which the

	property is sold by auction.
Enter Condition	Player’s Turn
Exit Condition	When Player finishes the turn Player cannot get out of the Jail

Use Case	Get out From Jail
Actor	Player
Possibilities	Player can choose to use a special card. Player can choose to pay the fee. Player can choose the chance of rolling double.
Flow of events	Player previously locked into the jail. Player tries one of the three possibilities to escape from jail. If the player succeeds, the player plays according to the dice. If not, the player waits for the next turn and in the third turn the game forces the player to pay the fee or use the special card.

Enter Condition	The player gets a penalty of going to jail.
Exit Condition	Player accomplishes one of the possibilities.

Use Case	Roll Dice
Actor	Player
Possibilities	Player rolls the dice
Flow of events	Player rolls the dice and moves on the board accordingly. If the player rolls double, the event is repeated again. If a player rolls double three times in a row, the player is sent to jail.
Enter Condition	The player decides to roll the dice
Exit Condition	Player arrives at a location according to the dice.

Use Case	Sell Building
Actor	Player
Possibilities	Player can sell the owned buildings.

Flow of events	Player chooses which building to sell and the Game Bank buys the building.
Enter Condition	Player chooses a building and clicks sell
Exit Condition	Bank buys the building.

Use Case	Sell Property
Actor	Player
Possibilities	Player sells property. Player chooses how to sell the property. After the requirements checks, property can be sold either by selling to another player or selling via auction.
Flow of events	Player chooses the property to sell. If the building exists in the CityGroup, the player first sells all the buildings in the CityGroup. After that, the player can sell the property to one specific player or can sell via auction.
Enter Condition	The player chooses a property to sell.
Exit Condition	Player sells the land.

Use Case	Pay Rent
Actor	Player
Possibilities	Player pays rent. If the owner collects all the CityGroup, the player pays the double rent. If the owner sets buildings in the City, the player pays extra rent.
Flow of events	Player stops at the property. If the property is owned by someone else, the player pays rent money to the owner.
Enter Condition	The player arrives at the owned location.
Exit Condition	Player pays rent to the owner.

Use Case	Buy Property
Actor	Player
Possibilities	Player decides to buy the property. Player decides not to buy the property.
Flow of events	If the player decides to buy the property, the budget is to check whether the

	player has enough money or not and does the required update and the player owns the property. Next time someone else arrives at that location, the player can take rent from the arriving player. If the player decides not to buy the property, the auction starts where every player can bid (including the player that decided not to buy). Highest bidder owns the property.
Enter Condition	Player arrives at the unowned property.
Exit Condition	The unowned property is sold to the player or someone else.

Use Case	Mortgage a Property
Actor	Player
Possibilities	The Player can mortgage a City Property with no buildings made on the CityGroup.
Flow of events	The player decides to mortgage a property. If the mortgage property's CityGroup has a building, the player first

	sells all of the buildings in the CityGroup.
Enter Condition	Player decides to mortgage a property.
Exit Condition	Player mortgages a property.

Use Case	Lift Mortgage
Actor	Player
Possibilities	Player can lift mortgages if the player has enough money.
Flow of events	Player decides to lift the mortgage. If the player has enough money, the player lifts the mortgage.
Enter Condition	Player decides to lift the mortgage.
Exit Condition	Player lifts the mortgage. Player does not have enough money (unsuccessful).

Use Case	Buy Building
Actor	Player

Possibilities	The player does have enough money and all the properties of the CityGroup. The player does not meet the requirements.
Flow of events	Player decides to buy a building for a City property. If the City property's all of the CityGroup properties is owned by the player and the player has enough money, the player can buy buildings. The building can set to a City property where each CityGroup's City properties have at least the same number of the current buildings in the City property.
Enter Condition	Player decides to buy a building.
Exit Condition	Player buys a building. Player does not meet the requirements.

Use Case Name	Chat
Actor	Player
Possibilities	Player chats with everyone Players chat privately.

	Player can see the messages
Flow of events	1-Player clicks the chat button. Chooses to talk publicly. Write the message to the message blog. Clicks send. Every player sees the message. 2-Player clicks the chat button. Chooses a player to chat with. Writes the message to the message blog. Clicks send. Only the selected player sees the message. 3-Player clicks the chat button. Sees the unread messages. Selects the desired one. Enters the conversation with the owner of the message.
Enter Condition	Player clicks the chat button.
Exit Condition	Player clicks the exit button or outside the chatbox. Player returns the board.

Use Case Name	Settings
Actor	Player
Possibilities	Player can change game music. Player can change screen resolution.

Flow of events	Player selects settings and sees two options. The player can exit the settings or the player can change one or the two options.
Enter Condition	Player selects settings.
Exit Condition	Player clicks exit.

Main Menu Use Case Model

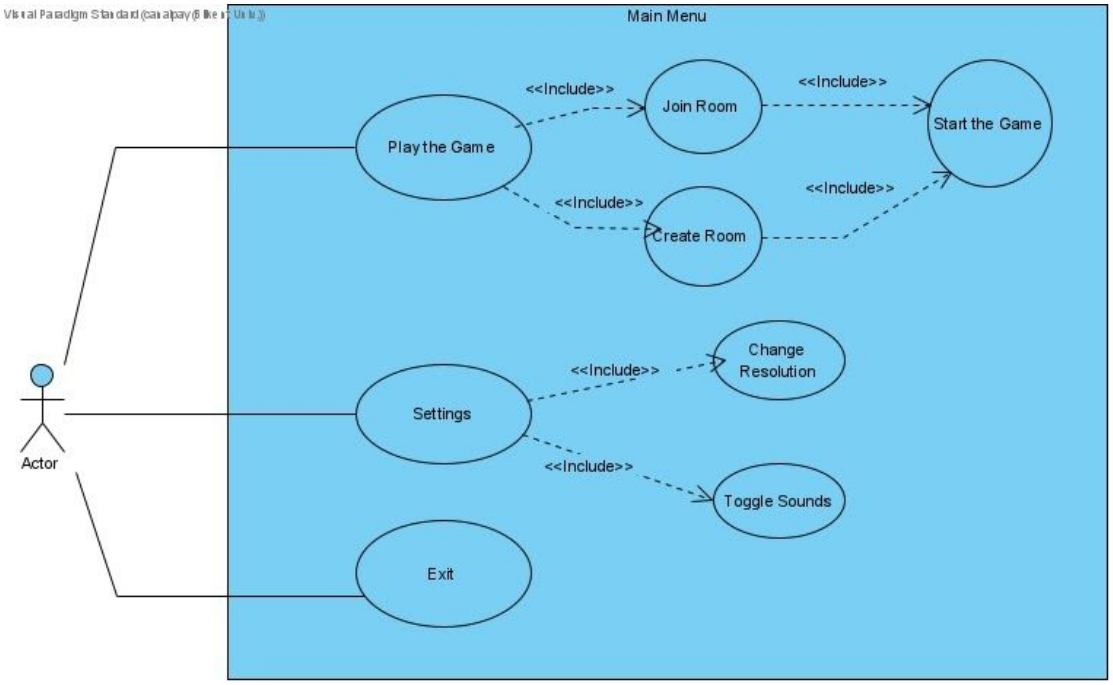

Figure 22: Main Menu in Monopoly Game Use-Case Model

Use Case	Main Menu Use Case
Actor	Player

Possibilities	Player can select: Play the Game Settings or Exit
Flow of events	Player starts the program and this menu will be displayed.
Enter Condition	Start the program.
Exit Condition	Player exits the program or starts the game.

Use Case	Play the Game
Actor	Player
Possibilities	Join a Room Create a Room
Flow of events	1- If the player chooses to join a room section, the player selects a room from a list of available rooms and joins one of them. 2- If the player chooses to create a room, the player sets the necessary

	requirements and waits for other users. After the player number is enough, the admin starts the game.
Enter Condition	Player clicks Play the Game button
Exit Condition	The game starts.

Use Case Name	Settings
Actor	Player
Possibilities	Player can toggle the sound. Player can change screen resolution.
Flow of events	Player selects settings and sees two options. The player can exit the settings or the player can change one or the two options.
Enter Condition	Player selects settings.
Exit Condition	Player clicks exit.

Object Class Diagram

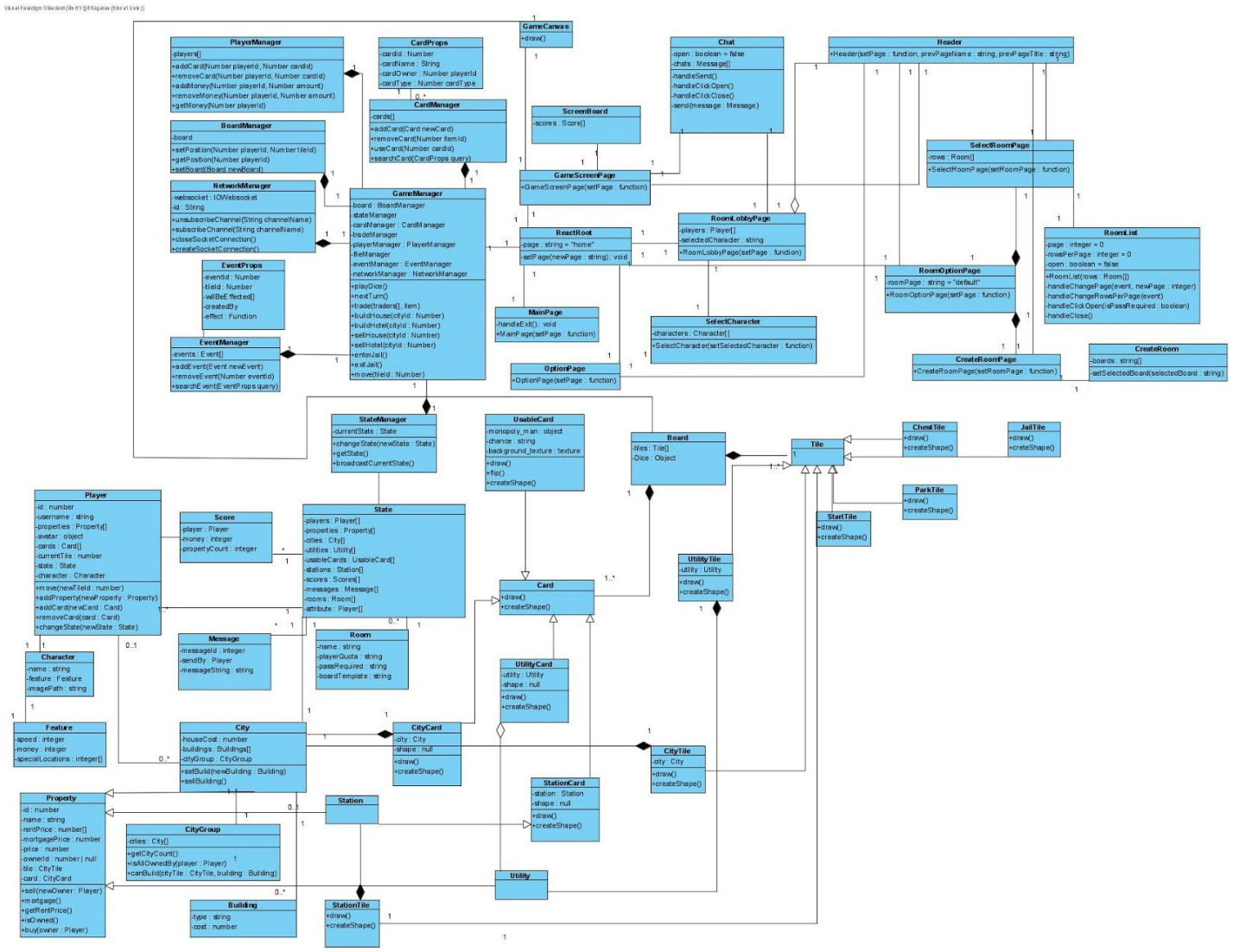

Figure 23: Object Class diagram

GameManager (Class):

GameManager class will handle all the managers and establish the connection between all of them. The data flow between different managers will be handled in this class. We can consider GameManager as the “Main controller” class. When the game starts, we will construct a GameManager object and it will initialize other controllers such as StateManager, FileManager, CardManager, EventManager, NetworkManager, TradeManager, BoardManager and PlayerManager. We can consider all of these managers different services, works and manage a subsystem of Monopoly games.

PlayerManager (Class):

PlayerManager class will handle the dataflow about a player. In Monopoly, a user can obtain cards, use cards, gain or lose money. It will hold all the player data in a game.

BoardManager (Class):

BoardManager class handles movements in the game board. Also according to the selected board in the room creation, it will create a special board.

TradeManager (Class):

TradeManager class handles the trade system which starts when a user doesn't want to buy a property. It handles the loop and logic about trading and auction systems.

StateManager (Class):

This object contains the current state of the game. GameManager will use StateManager to change and get the current state. State objects will be used in order to synchronize the current game between different users via localhost or a server. It is the main component of syncing an online game.

FileManager (Class):

The FileManager class will handle the IO stuff needed for Monopoly games. We will use this class in the setting system in order to store user preferences between sessions.

NetworkManager (Class):

The NetworkManager class will handle the online connection via websocket connection. We will use socket.io for implementation. Besides implementation, NetworkManager gives GameManager an interface for listening to different channels on socket connection.

CardManager (Class):

CardManager class will load a card set for a game and will be used for searching a card in the card set.

CardProps (Class):

This class is a wrapper for a query object which will be used for searching a card from cards array. It contains different parts of a card for searching.

EventManager (Class):

EventManager contains different events and controls the flow of events among the whole game. Actually we can assign a lot of different events therefore EventManager will be used a lot. We can give “pay to a player for passing over a city” or “passing over the starting point” as different examples for events.

EventProps (Class):

This class is a wrapper for a query object which will be used for searching an event from an events array. It contains different parts of an event for searching.

ReactRoot (Class):

ReactRoot class is the root class of the whole UI which is implemented with React.js. Every time there is a change in UI, at the end ReactRoot class renders all changes.

MainPage (Class):

This class provides users the ability to create a new game, navigate to the options menu, or exit game.

OptionPage (Class):

OptionPage class consists of a header and options such as adjusting fullscreen, adjusting volume and changing music.

RoomLobbyPage (Class):

This class consists of the Header class, the Chat class, the SelectCharacter class. Also, this class lists all the players for a room

SelectCharacter (Class):

This class lists all Character options.

RoomOptionPage (Class):

This class provides users the ability to go create a room page or select a room page.

CreateRoomPage (Class):

This class consists of the Header class which is described in the following sections of this report and the CreateRoom class.

CreateRoom (Class):

This class provides users the ability to create a room with room name and room password (this option is not required).

RoomList (Class):

This class lists all available rooms and their information. Also for each room there is a join option to this room

SelectRoomPage (Class):

This class consists of the Header class, and the RoomList class. In this class user can select rooms and chat with other players from this room.

GameScreenPage (Class):

This class is the user interface of the gameplay. The game will be played on a canvas which will be implemented by using Pixi.js. Therefore, this class contains the ScreenBoard class, the Chat class and the GameCanvas class.

ScreenBoard (Class):

This class is the user interface that displays a table that contains the Score object information

Chat (Class):

This class is a user interface for chatting. In this window, users can chat with each other while either in the room lobby or in the game.

Header (Class):

This class is used in every page except the MainPage class. It is used for navigation inside the application. It provides users to navigate the current page to the previous page.

GameCanvas (Class):

This class is used for connecting the user interface that will be implemented with React and the user interface that will be implemented with Pixi.js.

Card (class):

The card is a class which has got only a render method in order to draw cards that users have.

UsableCard(Class):

UsableCard class will be responsible for cards that can be consumed such as chest cards, chance cards, secret cards and quest cards. It will be used by tile classes and card classes.

UtilityCard (Class):

UtilityCard is a class that was inherited from Card interface. It has a render method and a property which contains the utility reference. You can get this reference by calling getUtility method.

StationCard (Class):

StationCard is a class that was inherited from Card interface. It has a render method and a property which contains the utility reference. You can get this reference by calling getStation method.

CityCard (Class):

CityCard is a class that was inherited from Card interface. It has a render method and a property which contains the city reference. You can get this reference by calling getCity method.

Tile(Class):

Tile is a class that extended from every tile. It has no attribute defaultly.

CityTile (Class):

CityTile class will be used in all cities that exist in the game. It will handle the drawing procedures of city tiles onto the game board. The image property will be used in order to separate this tile from tiles. This class has got a reference to the city which it shows via a property called the city.

UtilityTile (Class):

UtilityTile class will be used in all utilities that exist in the game such as “Water Works” and “Electric Company”. It will handle the drawing procedures of city tiles onto the game board. The image property will be used in order to separate this tile from tiles. This class has got a reference to the utility which it shows via a property called utility.

StartTile (Class):

StartTile class will be used in order to show the start tile. It will handle the drawing procedures of this tile on to the game board. The image property will be used in order to separate this tile from tiles. It has a connection to StartManager class in order to process players coming to this tile.

ParkTile (Class):

ParkTile class will be used in order to show the park tile in the original monopoly game. It will handle the drawing procedures of this tile on to the game board. The image property will be used in order to separate this tile from tiles.

JailTile (Class):

JailTile class will be used in order to show the jail tile. It will handle the drawing procedures of this tile on to the game board. The image property will be used in order to separate this tile from tiles. The logic of the jail is controlled by EventManager.

Board(Class):

Board class contains tiles array and dice object. Because every property has a tile on the board and they are drawn. Therefore, the tiles array is sufficient for us in board class.

ChestTile (Class):

ParkTile class will be used in order to show the Chest tile in the original monopoly game. It will handle the drawing procedures of this tile on to the game board. The image property will be used in order to separate this tile from tiles.

Player (Class):

The Player class contains all the information about a player. It has got different properties such as id, username, avatar, properties, cards, currentTile, and state. Properties and cards hold the stuff the player has got. Avatar and username will be used to differentiate the player from other ones. It has got different methods that change these properties such as addCard, removeProperty, and move.

Property (Class):

Property is a class that must be inherited by all the properties such that Cities and Utilities in-game. It has got different properties such as id, name, rent and mortgage price, ownerId, tile, and card which is a reference to the card object of this property. It has got methods that change these properties such as sell and buy.

City (Class):

City class inherited from the Property interface. It has got all the methods inherited from Property, in addition, it has got buildings and cityGroup properties, as a consequence it has got methods like build and sellBuilding. This object will be used in CityTile and CityCard for references.

Utility (Class):

Utility class inherited from the Property interface. It has got all the methods inherited from Property. This object will be used in UtilityTile and UtilityCard for references.

Station (Class):

Station class inherited from the Property interface. It has got all the methods inherited from Property. This object will be used in UtilityTile and UtilityCard for references.

Board (Class):

Board class contains cityGroups in cityGroup array. It has got a background. getCityCount method traverses all cityGroups and returns the total number of cities in the game. In the end, there is a render method that draws a board into the screen.

Message (Class):

Message class is used to hold instances for messages that are sent and received in the chat window. It has properties such as messageId to uniquely identify a message, sendBy to specify the sender of the message and messageString that contains the message context.

Score (Class):

Score class is used to hold the scores, in essence, the current amount of money that the players have and the number of properties that they have.

Room (Class):

Room class is used to hold the information regarding the game lobby. It holds the name of the room, player quota, whether a password is required or not and board template.

Character (Class):

All players in the game choose their character at the start. Different characters may have different features, which are created in Feature class. Each character has a unique name and imagePath that specifies the location of the image that the character is represented.

State (Class):

In order to record the current state of the game, State class holds the current state of all models, such as Player, Property, City, and the other model classes. For instance, an array of players is an attribute of State class, holding the information about all the players.

Feature (Class):

Feature class is used to give characters different features. Depending on a character, different features can be given to that character. Feature objects have speed, money, and specialLocations as their attributes.

Building (Class):

The building is a class that has a string property called type and an integer property called cost. It represents buildings in general.

CityGroup (Class):

CityGroup is a class that contains cities in the same group. It can check whether the group is owned by a player or not and determine if a special building can be built on a cityTile.

Dynamic Models

Sequence Diagram

Auction Sequence Diagram

Below is the sequence diagram regarding the auction case. Auction is done between players in order to give city to one of the players via getting offers from each player where the maximum offer gets the property. Auction starts after the UI sends a request to GameManager for starting the auction. GameManager then informs NetworkManager that the auction has started and NetworkManager again makes GameManager to initialize the auction with users participating in the auction, and the given property that the auction is performed for. GameManager controls the auction by communicating with the TradeManager class which is responsible for taking bids, or closing the auction when the conditions are satisfied (that all players except the winner did not bid in their turn). After starting the auction, the State of the game has to be changed, and that is guaranteed by making GameManager call updateState() function of the StateManager class. The loop that is responsible for taking offers from players, updating the game state accordingly by communicating the appropriate classes, and closing the auction, continues to execute until the closing condition is satisfied. If all players except one player skips their turn to offer price, the auction is closed via closeTrade() method and the property is given to the winner. In order to make sure that these are happening, all the methods are called consecutively and accordingly. addCard() method in GameManager calls addCard() method of PlayerManager which then calls addOwner() method of Property to give the property to the winner player and close the auction. In addition to that, removeMoney()

method is making sure that the offered money is withdrawn from the player's current balance. Auction case is completed after GameManager changes the current state of the game and informs the UI that the auction is completed.

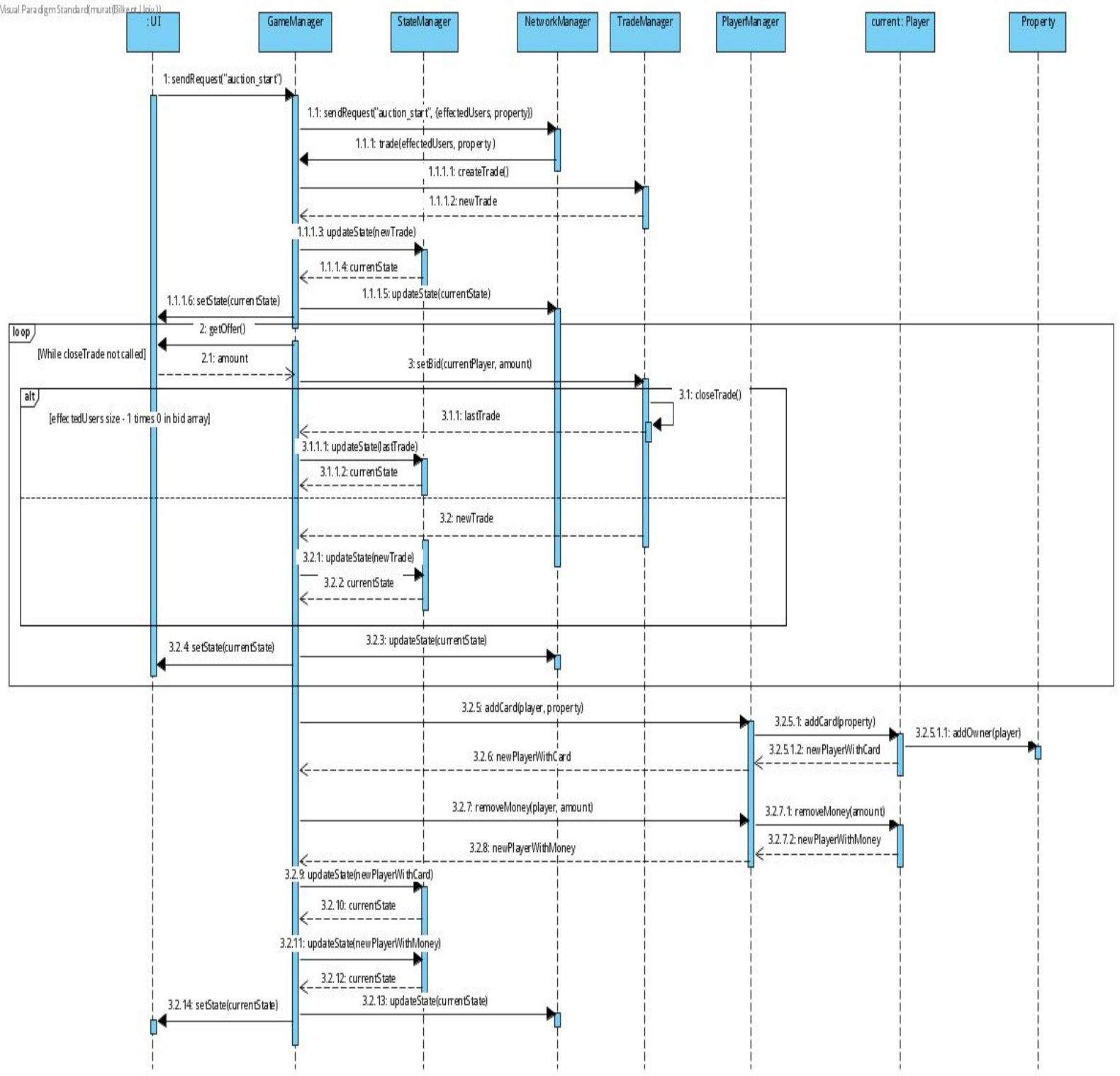

Figure 24: Sequence Diagram of Auction

State Diagram

State Diagram of Setting House

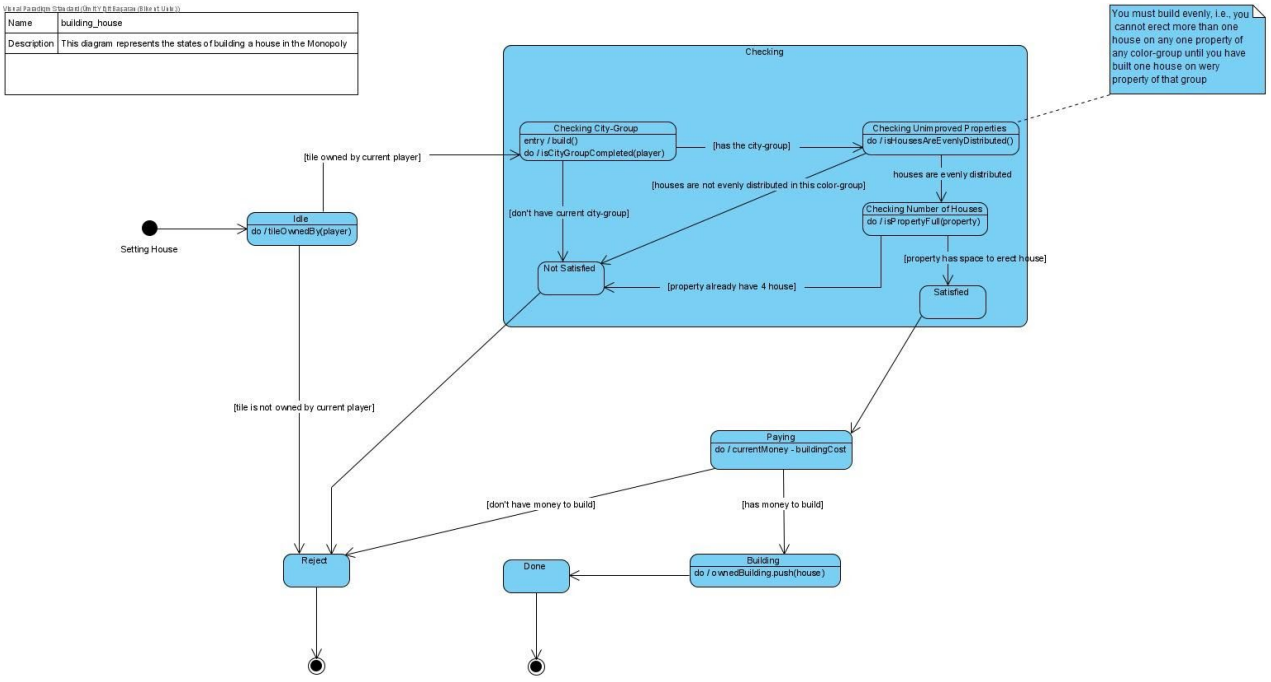

Figure 25: State Diagram of Setting a House

In Figure 30, we draw a state diagram for setting a house in Monopoly. This state diagram starts with “Idle” state where the current tile is checked to see whether this tile is owned by the current player or not. If this property is owned by the current player then the state goes to the “Checking” state otherwise building operation is rejected. In “Checking” state, there are sub-states that check the situation about building a house. In the “Checking Color-Group” state, the program checks whether the user has the color-group to set a house or not, if the player has the color-group

then the state goes to “Checking Unimproved Properties” state and if not state goes to “Not Satisfied” state. In “Checking Unimproved Properties” state, the program checks whether there are unimproved properties in the color-group. That is, it looks at all the tiles and if the houses are not evenly distributed then state goes to a not satisfied state, otherwise state goes to “Checking Number of Houses” state. In the “Checking Number of Houses” state, the program checks whether there is empty space to set a house on property or not and if there is space then the state goes to “Satisfied” state otherwise “Not Satisfied” state. From “Not Satisfied” state, state goes to “Reject” state. From “Satisfied” state, state goes to “Paying” state. In this state, the program takes the money from the user and if the budget doesn't satisfy the house, the state goes to “Reject” state. However, if the money satisfies the cost of the house then the state goes to “Building” state. In this state, the house is added to the list of houses that the current player has. After the house is added to the list, the state goes to “Done” state. After “Done” and “Reject” states, the state diagram is finished.

Activity Diagram of the Gameplay

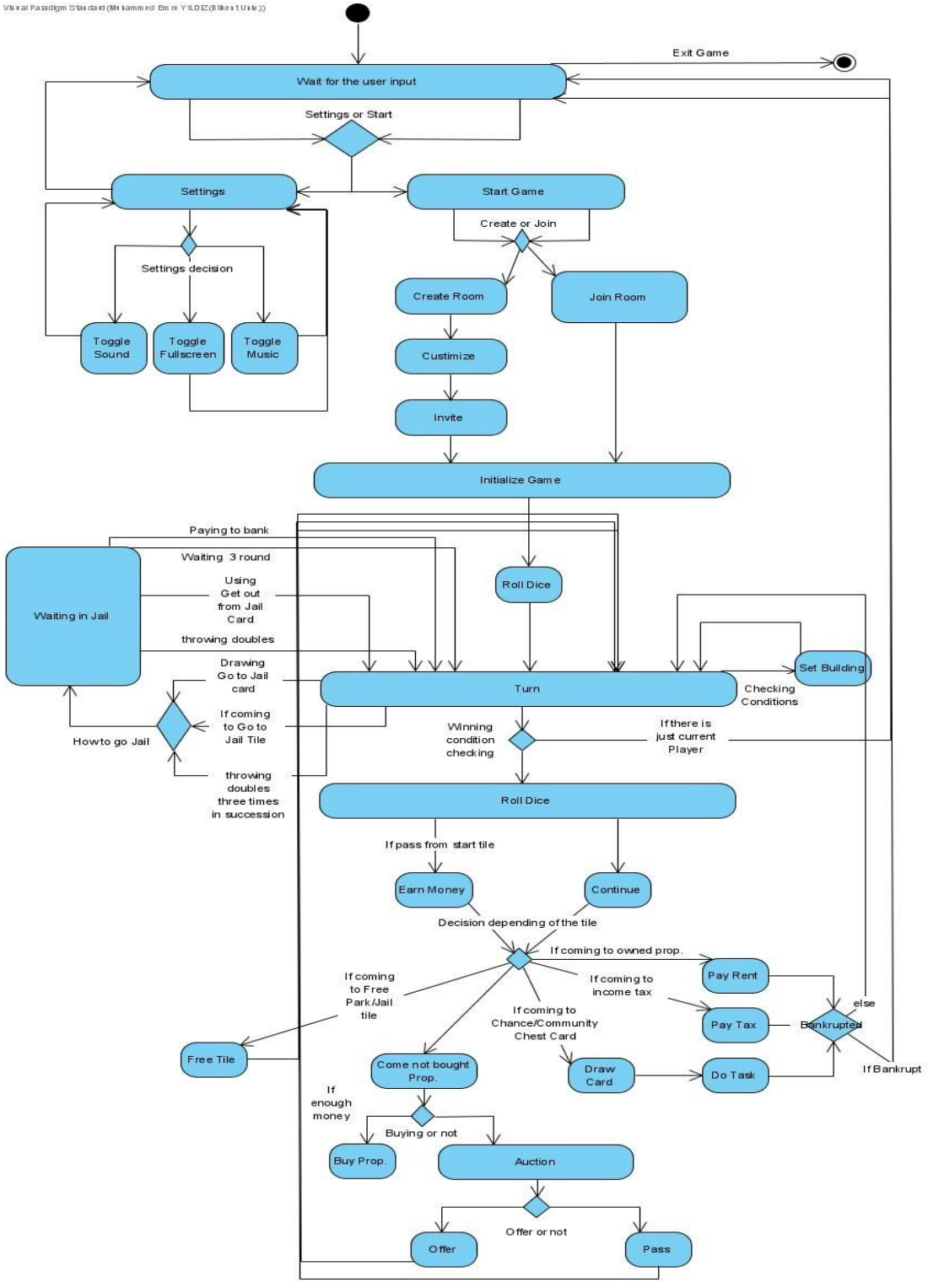

Figure 26: Activity Diagram of the Gameplay

In the Activity Diagram, the flow of the game is shown by representing the actions as rectangles, and the relationship between the actions by arrows. The game is opened by the user as the first action, in which the main menu is shown to the user. At that point, there are two actions that the user can select to perform, starting a game, or going into the settings menu. If a user decides to go with the settings, s/he can adjust the setting according to their preferences and can go back. Else, s/he can start a game by creating a new room or joining an existing room. While creating the game, the user can also invite other players and customize the game. After completing these steps, the Join Room and Create Room actions merge together in the Initialize Game action. After that point, the diagram represents how the in-game actions flow. After rolling dice, the player moves, and the first if condition to be checked is whether the player passes through the starting point or not. If a player passes through the start tile, s/he earns money, else s/he does not. Then there are several if conditions, in which the next action for the current player is determined. For example, if the player ends their move at a tile that is owned by another player, s/he has to pay rent to that user. Or, if the player is at one of the card tiles at the end of the move, the Draw Card and Do Task actions are performed respectively. After a player completes their turn, the actions are merged and returned back to the Turn action, where the steps are repeated until the end of the game. If a player bankrupted, it directly goes to wait for the user input activity so the player can exit or play a new game. Also, if there is just one player, otherones bankrupted, the current player is the winner. Diagram goes to wait for the user input activity again with the same purposes.

UI Mockups

Figure 27: Main Menu

This is the expected template for the main menu of the game. There is the logo of the game at the top right corner, as well as three buttons for starting a game, adjusting the settings, and exit.

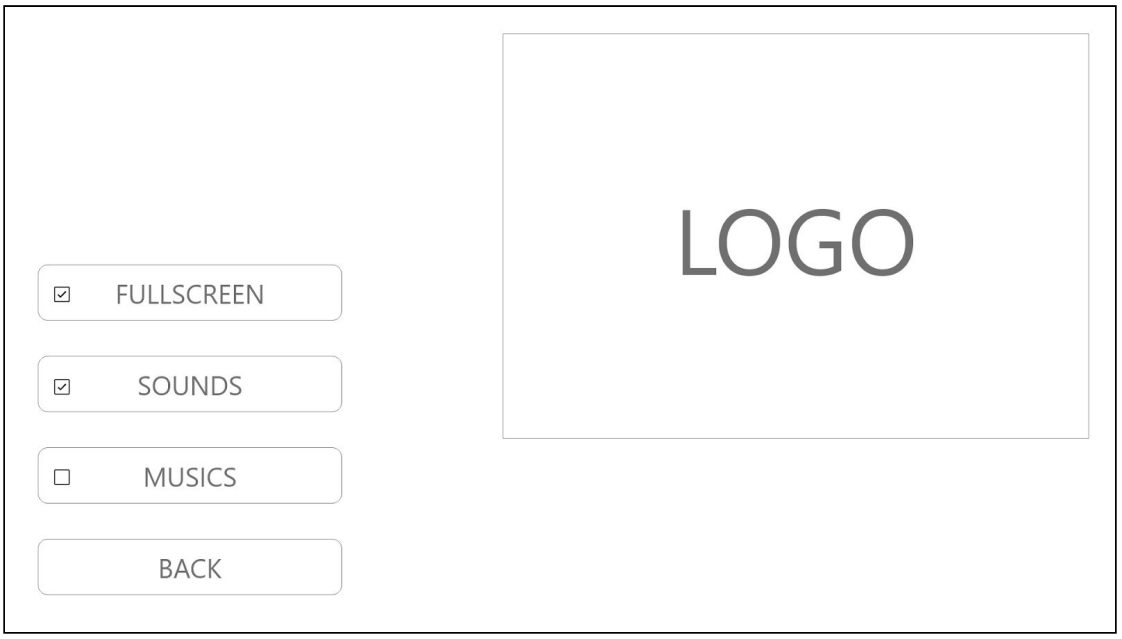

Figure 28: Options

The template above is to be used for the settings menu. Users can adjust sound, music, screen resolution, and can go back to the main menu by using one of the four buttons.

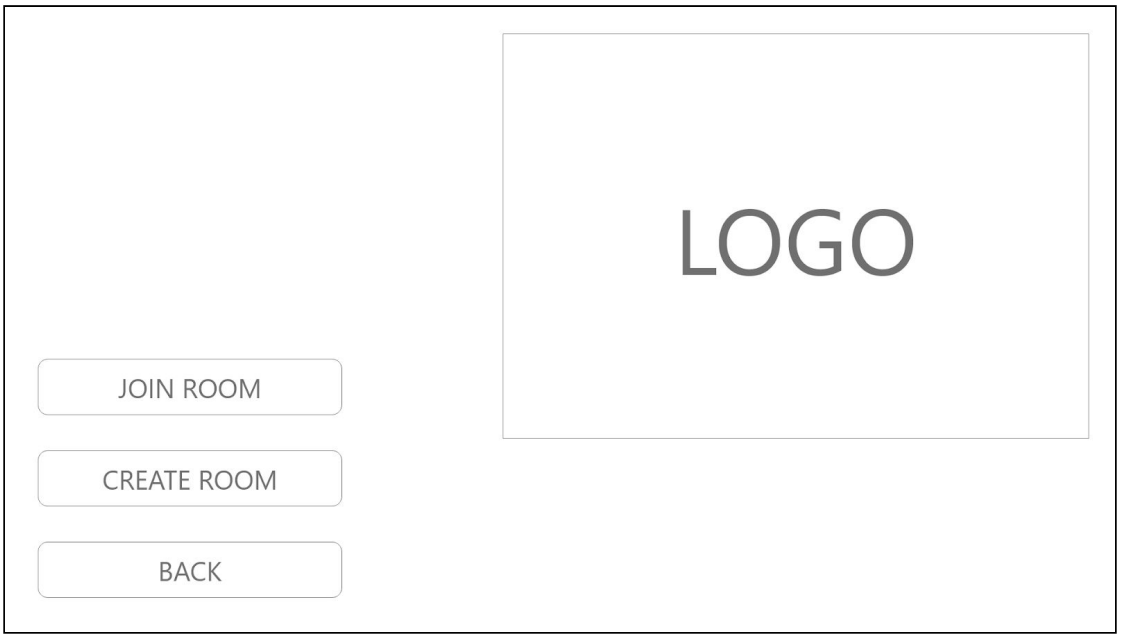

Figure 29: Start the game menu

The screen above is shown to the user when s/he clicks on the start game button on the main menu. Here the player can create a new room or can join an existing room by selecting the corresponding option. Also, the back button can be used to go back to the main menu.

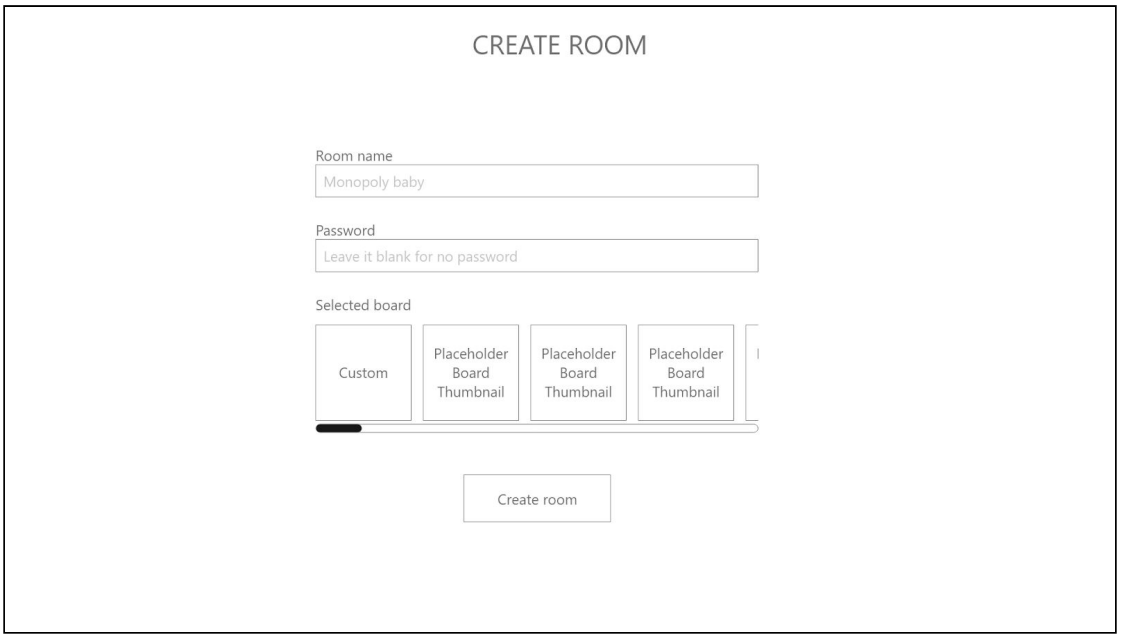

Figure 30: Create Room screen

The template above is used to create a new room. It consists of two text areas, in which the user enters the name of the room and the password for the room. In addition to that, users can select from the predefined templates while creating the room. The Create Room button below completes the process and creates the room according to the prompted settings from the user.

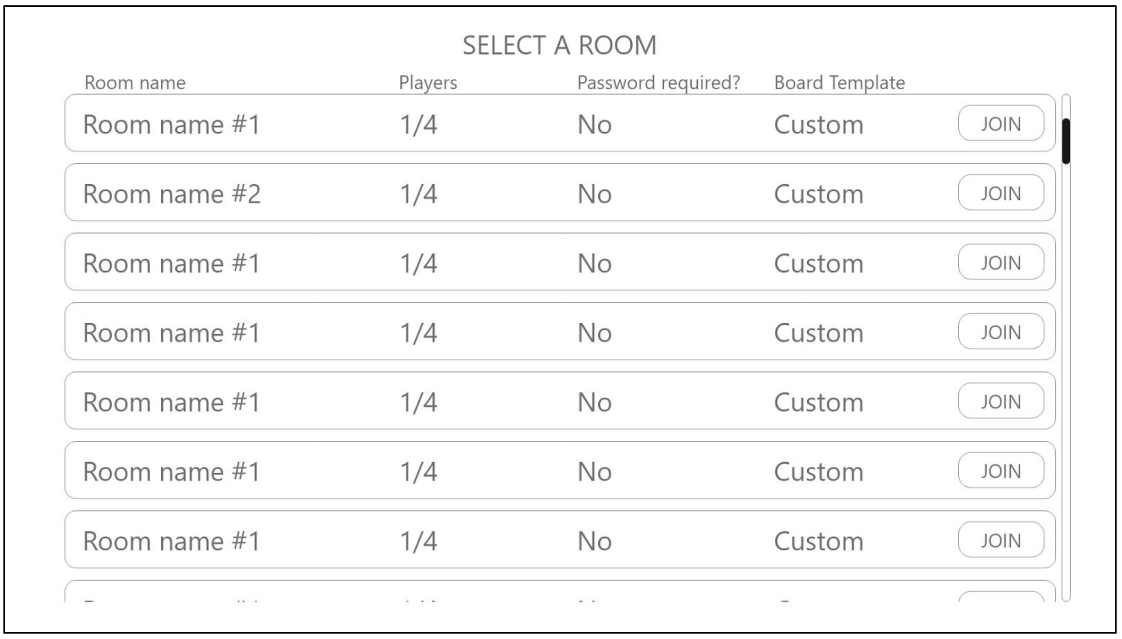

Figure 31: Join Room screen

While joining an existing room, users can select the room they want to join. From the menu, users can see the current situation of the rooms, such as the name of the rooms, how many players are in the room, and etc. After deciding on which room to join, the user can click on the Join button and can join the room.

Figure 32: Ask password when joining a room

After pressing the join button, the player must enter the password of the room that would play.

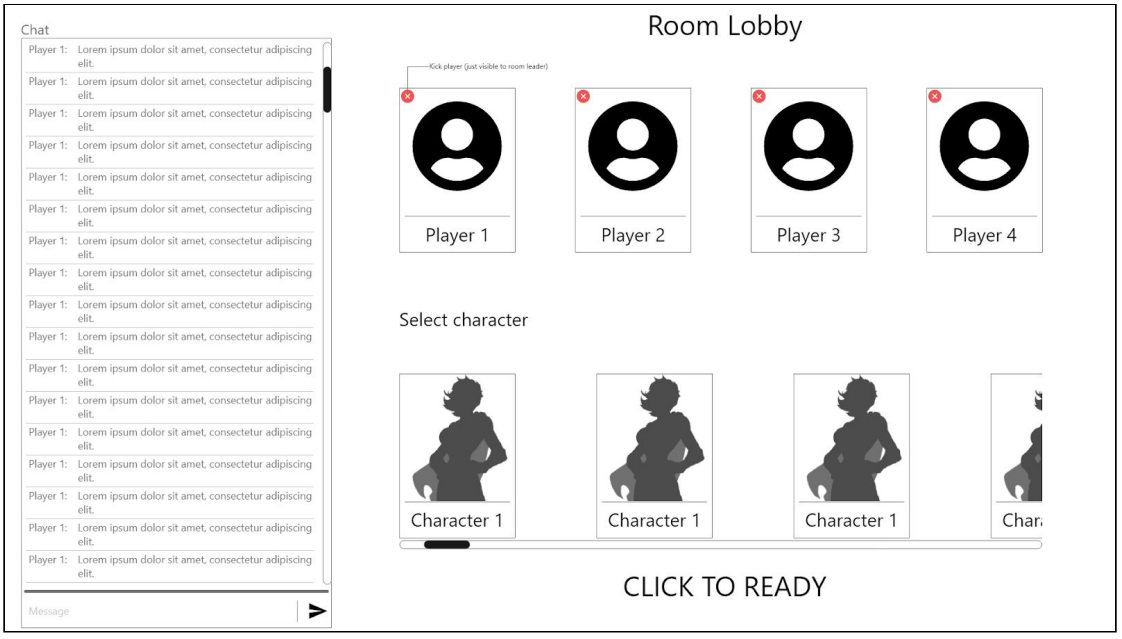

Figure 33: Room lobby screen

In the lobby screen, players can select their characters that have special skills. On the left side of the screen, players can chat with each other. After all, players state that they are ready, the game will initialize.

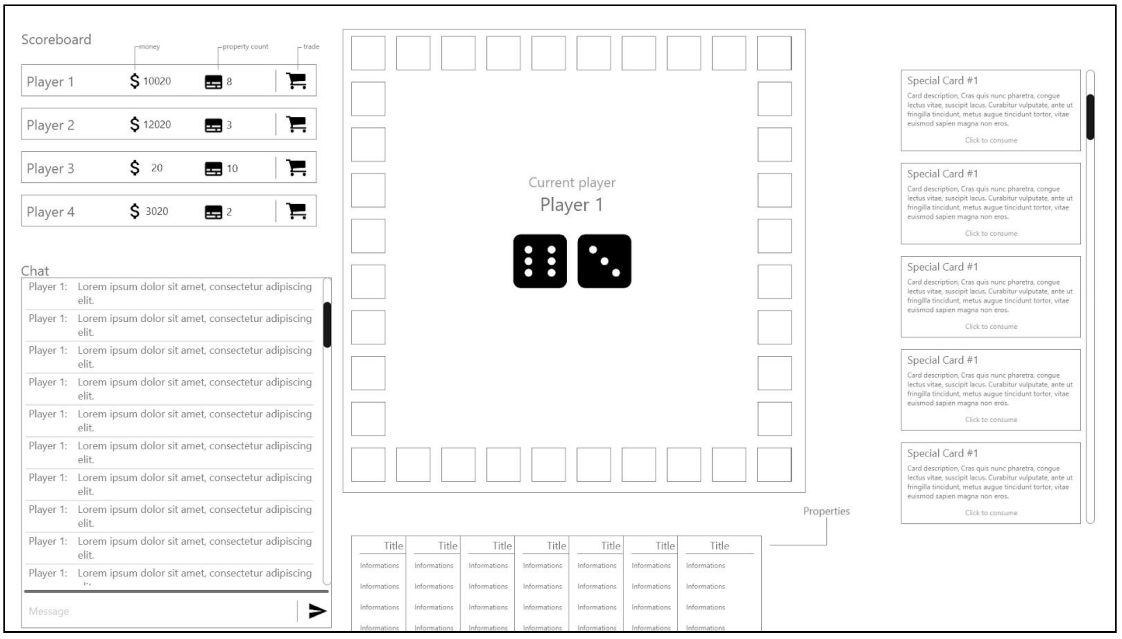

Figure 34: Game screen

This is the Game Screen mockup. The bottom part of the screen represents the cards of the user, the left side is a scoreboard and chat, the right side is the special cards the user earned. Players who are waiting for their turns would see this screen. In the middle of the board, 2 dice and a Current Player are shown.

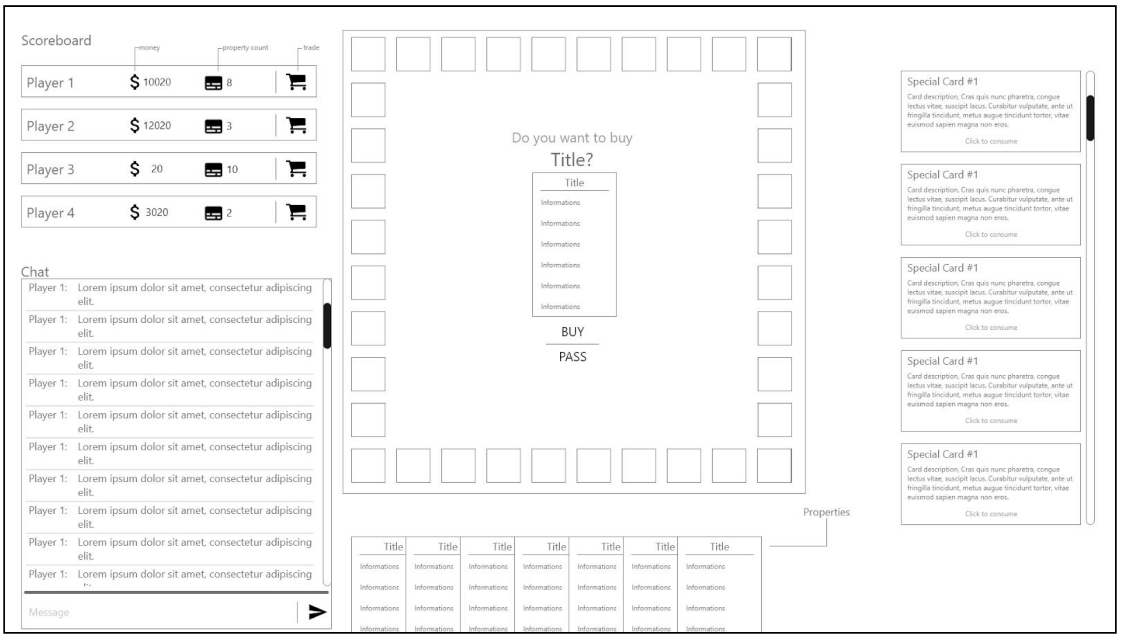

Figure 35: Buying a property

This is buying a property mockup. The bottom part of the screen represents the cards of the user, the left side is a scoreboard and chat, the right side is the special cards the user earned. In the middle of the board, there is a pop-up and asks “Do you want to buy tile?” and the user can select BUY or PASS.

References

[1] (n.d.). Retrieved October 28, 2020, from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

[2] (n.d.). Retrieved October 29, 2020, from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/

[3] Bruegge, B. (2014). *Object-oriented software engineering using UML, Patterns, and Java*. Pearson.

[4] Electron Support. (n.d.). Retrieved December 12, 2020, from https://www.electronjs.org/docs/tutorial/support

[5] Gough, C. (2020, October 16). Number of Steam users 2020. Retrieved December 12, 2020, from https://www.statista.com/statistics/308330/number-stream-users/