CS-319 TERM PROJECT

Section 2

Group 2A

Monopoly

Analysis Report

Project Group Members:

1- Can Kırımca

2- Burak Yiğit Uslu

3- Cemre Biltekin

4- Can Kırşallıoba

5- Mustafa Yaşar

6- Emre Açıkgöz

Supervisor: Eray Tüzün

Contents

1. Introduction

As a group of 6 people from CS319 - Object-Oriented Software Engineering course, we have decided to implement a digital version of the Monopoly game that was published by *Hasbro, Inc* in 1935 for the term project using Java. Monopoly is a board game that can be played with 2 to 6 players who aim to make their opponents declare bankruptcy and win the game.

One of the reasons that we have decided to create a digital version of Monopoly is that, considering the publication year, it is impressive that Monopoly is still one of the leading board games that is played in so many countries, thus, it would be entertaining to play this game on a computer with properties that we define. Another reason is that, since there are many instances in the game, we could implement them as objects in Java, thus, it is convenient to design it as a part of Object-Oriented Software Engineering course.

In addition to the classic Monopoly game with all of the base game functionalities, we will add various features in order to make a brand new and more entertaining digital version of the game. These features are discussed in detail in the *Overview* section under their own headlines, and include Artificial Intelligence (AI) players with custom characteristics, teleportation tiles such as airports and train stations, user-created custom boards, save & load game features and more...

2. Overview

2.1. Game Play

The main objective of the game is obtaining wealth by selling and buying properties, and being the only player standing by forcing other players to declare bankruptcy.

The game is played with 2 to 6 players. For multiplayer, every player will be playing the game on the same computer in turns. Players will be able to choose the board they will be playing the game on, and this includes user created custom boards as well.

Every turn players will roll the dice, and move their pieces according to the sum of the dice. After rolling dice and moving accordingly, players will perform the task related to the tile they landed on, such as drawing a chance card or a community chest card, or buying the property they landed on (If the player lands on an unowned property and is not willing to buy that property, auction takes place in that turn.) Players may propose trade offers to other players during their turn. After that, by selecting "End Turn", the turn for that player will be over, and the game will continue with the next player. If a player rolls a double dice in their turn, after playing that round they will roll again. However, throwing double dice three turns in a row results in jail.

2.2. Board

The board in the game will have the same number of tiles as the classical Monopoly board. Thus, it will be a square and every edge will be composed of 11 tiles. With the exception of teleportation tiles (which replace the train stations in the classic Monopoly), and the removal of commodity tiles (such as Water Works) to add more chance and community chest card tiles, the board will be the same as the classic Monopoly. Teleportation tiles are 2 pairs of 2 tiles, which teleport the landing player to its counterpart when landed on. There are a pair of train station tiles as well as a pair of airport tiles.

2.3. Board Customization

By going into the board customization screen using the BoardBuilder, players will be able to customize the property tiles on the board, and save their customized boards. They will later be able to choose the boards they created before beginning a game, and play on their custom boards. While customizing, players will be able to change the names, rents, mortgage values, upgrading with house and hotel costs and the prices of the properties, and the money they get after passing the Go tile, however, they will not be able to change the number of tiles.

2.4. Beginning the Game

Every player will be able to choose their unique token, that they will be represented with on the board. After choosing the token, every player will throw dice and the player with the highest total will play the first turn, and the player who rolled the dice for the second highest sum will play second and so on. All players start with the predefined amount of money, on the "GO" tile. The starting money is determined by the pace of the game and the wage players receive once they pass from the GO tile, which is customizable by the user. Once the first player clicks the "Roll Dice" button, the game starts.

2.5. Bank

Financial actions such as selling, and mortgaging properties, distribution of title deed cards, and building houses and hotels and such will be handled by the bank automatically. Bank also handles the auctions and sells properties. Paying of the wages to the players that pass through the "GO" tile, and players receiving or paying money due to chance or community chest cards are also handled by the bank.

There is not a Bank class in the design, however, the concept of a Bank represents the entirety of financial actions, which are handled by TurnManager, Action, Game Session and the respective tile classes.

2.6. Chance Cards and Community Chests

In the board there are several tiles that require the player to draw a chance or a community chest card. When a player lands on these tiles, the player must perform the task given in the card. The tasks the player may draw varies, and includes instructions like "Go Directly to the Go Tile", "Go Directly to the Jail", "Get out of Jail Free". Additionally, the tasks may require the player to pay or receive a sum of money.

There are also special cards, such as the "Get out of Jail For Free" card. Once drawn, these cards may be kept in the players inventory, and used when necessary. For example, "Get out of Jail For Free" card is a special card that enables the player to get out of jail if they ever go to jail for no cost. The player who owns these cards can also trade them with other players.

2.7. Buying and Selling Property

When a player lands on a property that is not owned, he/she can buy that property from the Bank at its predefined price. If the player buys that property, he/she gets the title deed card containing the information about that property such as the price for upgrading with a house and a hotel. However, if the player landing on an unowned property does not wish to buy that property, or does not want to buy the property at the predefined price, the bank starts an auction where the highest bidder gets that property. The bidding will start at a set price that is half the property's original value with 25 seconds on an auction timer, and every player including the one who denied to buy it can bid to raise the price. Every player bids a set amount that is set based on the value of the property written on the title deed card, and every bid adds 10 seconds to the timer.

Players who own the title deed card of a property can sell that property to another player in their turn by making a trade offer. Other players may accept or refuse the offer. Selling a property to the AI players is also possible, and the AI players decide to accept or decline the trade offer based on their custom characteristics.

If a player collects all properties of a color group, players who land on a property of this group have to pay double the normal rent. This holds even if one or

more property in the color group is mortgaged, as long as the property the other player landed on is not mortgaged. Furthermore, the owner of a set of properties may further upgrade that property by building houses or a hotel on the said property.

The properties that have houses or a hotel built on them cannot be sold. The houses and hotels must be sold to the bank first, and only after that, the property can be sold again. Houses and hotels can be sold any time for half of the price paid for them.

Even if a player is in jail, they are able to sell and buy property, and make or receive trade offers with other players.

2.8. Trade System

Should a player want to trade something to any of the players, he/she sends a request to the player that he/she wants to trade with. If a player initiates a trade offer (which he/she can only do in his/her turn), the trade screen is opened and the player can make offers involving anything the other player has, money, get out of jail free card, title deed cards. Once the first player makes an offer, the player receiving the offer may choose to accept or reject the offer on their turn. For instance, imagine a scenario where a player has 2 of the yellow properties, they need the last yellow property but it's already owned. The player who needs the last yellow property sends a request to the other player and makes a desirable offer, the other player could approve and give the title deed card or reject the offer and the game continues.

2.9. Paying Rent

If a player lands on an owned property, the owner collects money from that player in accordance with the prices in title deed cards, according to the "upgrade level" (i.e. how many houses the owner has) of the property.

If the landed property is mortgaged, no money is collected.

Having all of the properties of a group (in other words, properties with the same color) doubles the rent, therefore, it's a good strategy to buy properties that share the same color. This rule applies to unmortgaged properties even if the other properties that are in the color-group are mortgaged.

Every house and hotel constructed to the property substantially increases the rent.

2.10. Upgrading Properties with Houses and Hotels

Players are able to construct houses and hotels to their properties, given that they have all of the properties in that color group. Building houses or hotels on properties increases the rent income of those properties significantly.

Every property has a set price for upgrading the property. Players pay those prices to upgrade the property. By upgrading, players may have 1 house, 2 houses, 3 houses, 4 houses or a hotel on a property. The upgrading process is a gradual process and players may not skip any of the upgrade levels to upgrade to a higher level.

As said above, to construct a house to a property, the player must have every property in that color-group. If a player buys a house from the bank, they can construct that house to any of the properties in that color-group. The next house to be erected must be put to any of the unimproved properties in that color-group. The players can buy houses in their turns and they can put them to any of their properties, however, they must do that *evenly, (*i.e. if you have 1 house in a property and there are 2 unimproved properties, players cannot put that house to the one that has 1 house. Before further upgrading that particular property, they must first fill the unimproved properties. Similarly, if a player has 2 houses on a property and 1 house on the others, he cannot put the new house to the property with 2 houses, the player must fill the others first). There can be 4 houses maximum on a property. Similar to the building process, while selling, players must sell their houses and hotels to the bank evenly as well.

When a player has 4 houses on each property in the same color-group, they can buy a hotel and construct that hotel to any of the property. The houses in that property are returned to the bank. Only one hotel can be erected to a property.

2.11. Mortgages

Players can mortgage their unimproved properties through the bank at any time. In order to mortgage a property, all of the houses and hotels on the same color-group must be sold at a half-price to the bank. Other players do not pay rent if they land on a mortgaged property, but they do pay rent if they land on other unmortgaged properties. In order to lift the mortgage, the owner must pay the amount of the mortgage plus 10% interest to the bank. If the player can agree with another player on the price, he/she can still sell the mortgaged property to them. The sold property will still be mortgaged, thus, the new owner must lift the mortgage and must pay the amount of the mortgage plus 10% interest to the bank.

2.12. Jail

A player may be thrown to jail due to a number of reasons. These reasons are:

1- Player throws dice and lands on the "Go to Jail" tile.

2- Player draws the "Go Directly to Jail" chance or community chest card.

3- Player throws double dice three times in a row.

The turn of the player who is thrown to the jail due to the above situation immediately ends.

In order to get out of jail, the players can do the followings:

1- They can wait in jail for 3 rounds.

2- They can pay a predefined fine to the bank and continue playing by throwing dice.

3- They can turn in a "Get out of Jail" card if they possess one.

4- They can purchase a "Get out of Jail" card from another player.

5- They throw the dice, if the dice rolled is double, they move their pieces forward as much as the sum of the rolled dice. If they throw the dice for 3 rounds when they are in jail and none of them are double, at the end of the third round, they pay $50 to the bank and continue playing by moving their piece forward as much as the total of the last throw.

2.13. Teleportation Tiles

As an extra feature, the transportation tiles were replaced with 2 pairs of teleportation tiles. Teleportation tiles are two pairs, one a train station pair and the other pair an airport tile pair. Should a player land on any of the teleportation tiles, they will directly be transported to the counterpart teleportation tile, i. e. a train tile transports to the other train tile. If the player passes from the GO tile during this teleportation, they will not be paid their wage by the bank.

2.14. Artificial Intelligence Players

It has been already mentioned above that players will be able to play against real players on local co-op or they may choose to play against AI players. However, there is more to playing with the AI. In addition to being able to analyze the board and their situation, the host will be able to customize the AI's in the game by selecting characteristics for them, which will affect their decision making processes. The sets of characteristics will come as predefined, with unique names such as "Adventurous Capitalist" or "Ebenezer Scrooge" (possibly the most famous stingy persona in popular culture). Setting the AI characteristics to Ebenezer Scrooge, for example will lead the AI to behave very stingy, and try to keep the most of its money in his account, and not take any risks. Adventurous capitalist will be the opposite, and will try to buy more property, and will take risks. There will be more AI characteristics in the finished game.

2.15. Bankruptcy

When players feel that they cannot win the game anymore, they may resign from the game by declaring bankruptcy. A player who lands on a tile that obliges him to pay an amount greater than their balance will be forced to declare bankruptcy and resign from the game. Players may be forced to declare bankruptcy due to tasks including paying tax, paying rent to another player, drawing cards that they require them to pay, and such. The last person standing when everyone else has declared bankruptcy is the winner of the game.

2.16. Save & Load Game

Typically, monopoly games last very long, usually more than a couple hours. As a result of this, it becomes difficult to finish the game in one setting. As an extra feature of the digital version of Monopoly, players can save their games, load back their saved game to continue back anytime. The save game files hold all of the important data required to restore the game back to the exact state of the game before saving, and this includes custom datasets as well. The saved games are stored locally on the host computer.

2.17. Game Pace

As an extra feature in this version of Monopoly, players will be able to manually select the pace of the game. The pace of the game fundamentally alters how features of the game that change the length of the game behave. For example, if the game is set to a faster pace, the purchasing and upgrading prices for properties will be lower (leading to players buying the properties faster) and rent players have to pay the owner of the properties will be higher (leading to faster bankruptcies). The users will also be able to set the pace with their custom-built maps, because the pace will affect all the tiles on the board by the same constant.

In a "High" paced game, all rents will be 20% higher and all property purchases will cost 20% less and "Low" paced game will be the opposite. "Medium" pace will not modify any of these values players will play with the values originally set by the creator of the board. There may be more paces in the release of the game.

3. Functional Requirements

3.1. Starting the Game

- The game can be started in two ways; by creating a new game or loading an existing game.

3.1.1. Creating a New Game

- The players can access this screen from the Main Menu by clicking New Game button. When players enter this screen, they can choose the game pace, they can choose the game board, they can choose how many players will be playing the game and how many of the players will be AI Characters. Additionally, characteristics of these bots can be chosen.

3.1.2. Loading Game

- The players can access this screen from the Main Menu by clicking the Load Game button. When players enter this screen, they can choose the saved game from the database. No other adjustments required.

3.2. How to Play

- In the main menu screen, players can see the How to Play button, by clicking on it, they can access the How to Play screen where the necessary information about the game is present.
- The players are informed about the buttons and their operations in the game.

- The players are informed about the operations such as trading, auction, chance and community card operations, etc.
- The players are informed about the characteristics of the AI players.

3.3. Settings

- Players can access the settings screen from the main menu screen or they can access this screen from the Menu button while playing the game. They can adjust the Music and Sound level on this screen.

3.4. Credits

- Players can access the Credits screen from the main menu screen by clicking the Credits button. The players are informed about who the creators of this game are, what this game is implemented for, etc.

3.5. In Game Functionalities

- Players can buy properties.
- Players can upgrade their properties.
- Players can downgrade their properties.
- Players can throw dice.
- Players can teleport between tiles by landing at one of the Teleportation tiles, whether that be the airport or the trains station.
- Players can go jail by landing on a jail tile with their *token*, 3 consecutive double dice rolling, or drawing a "*Go to Jail*" card.
- Players can bail out of jail by trying to roll double dice, paying money, using "*Get out of Jail card*", purchasing "Get out of Jail card" from another player, and waiting for 3 rounds.
- Players can mortgage their properties.
- Players can remove mortgages from their properties.
- Players can draw *Community Chest Cards* or *Chance Cards.*

- Players can offer money to other players to buy their properties.
- Players can declare bankruptcy.
- Players can save their game session.

4. Nonfunctional Requirements

4.1. Usability

- A user whose English proficiency level is A1 or A2 (beginner-level) should understand button labels, menu headings and basic action words (verbs) in game (ex. Draw a card, Trade, Buy, Sell etc.). They might need a dictionary for complex texts like the game manual and the instructions on the chance and community chest cards.

- A user whose English proficiency level is B1 or higher should read and understand button labels, menu headings, action words, game manual, instructions on the chance and community chest cards (all text in the game).

- A user without prior knowledge of the board game should learn the rules of the game and play the game by reading and referring to the How To Play screen.

- A typical user does not have to refer to the user manual to understand all buttons' functions since all buttons are labelled with at most 4 words to become self-explanatory.

- All screens should be reached, and functions can be performed with at most 13 button clicks.

- A user with prior knowledge of the board game does not have to refer to the rules of the game since there are visual parallels between the physical and the digital version to create a sense of familiarity in the user interface. The board colour (RGB (191, 219, 174)), content of title deed cards, colour groups, and dice should be visually identical to the classic board game. Hotel and house tokens and some player tokens (car, hat, motorcycle) should keep their visual characteristics from the board game like hotels are red buildings, houses are smaller than hotels in size and are green, and player tokens represent tokens from the classic board game as their 2D version.

- The orientation for the text and image inside all tiles on the board must be upright for the user to read them and acknowledge the tile's purpose in less than a minute.

- The user does not have to know keyboard functions since all actions in the game are executed by mouse control except the trade panel for ease of use.

In the trade panel, the user should only use numpad to enter an amount of money in an offer.

4.2. Reliability

● Should the game crash or the user quit inappropriately, the data of the user is saved and stored in a text file. The user must only restart the game, choose his/her desired save file, and continue from where it is left off. Saving and loading data is especially crucial in Monopoly since it is a long game.

● In terms of the most data loss that can occur, the last turn data might not be stored which is the player's data who did not end his/her turn before the game crashed.

● The game must be reliable 99% in a single game period.

4.3. Performance

● The game is a turn-based game and supports up to six players on the same computer.

● Only one player at a time must execute actions in game.

● AI players' turn must not be longer than four seconds, so that the human player's average turn time should determine the game time.

● The maximum response time between click and reaction must be two seconds.

● The average response time between click and reaction must be between 0.1 seconds to 1 second.

● The loading time at the start for the game must be between 0.1 seconds to 2 seconds.

4.4. Supportability

● GitHub must notify about release of a new version of the game which might include updates that are related to new modules, new play styles, and new

board themes if the user is following the game repository. If the user wants to play the new version, he/she must download it from GitHub.

● The bugs must be maintained via Github Issues by the developers. Developers must respond to an issue within 48 hours.

● The game must be built to support Mac OS and Linux after its release on Windows within a month.

4.5. Implementation

● The game must run on Windows.
● All related software associated with the game must be written in Java.
● The user must install Java (JRE) before downloading the game.
● The computer must support 1920x1080 resolution to run the game.

4.6. Interface

The save file and the customized boards must be stored in a text file which includes the last status of the game including the possessions of the players, and whose turn it is.

4.7. Packaging

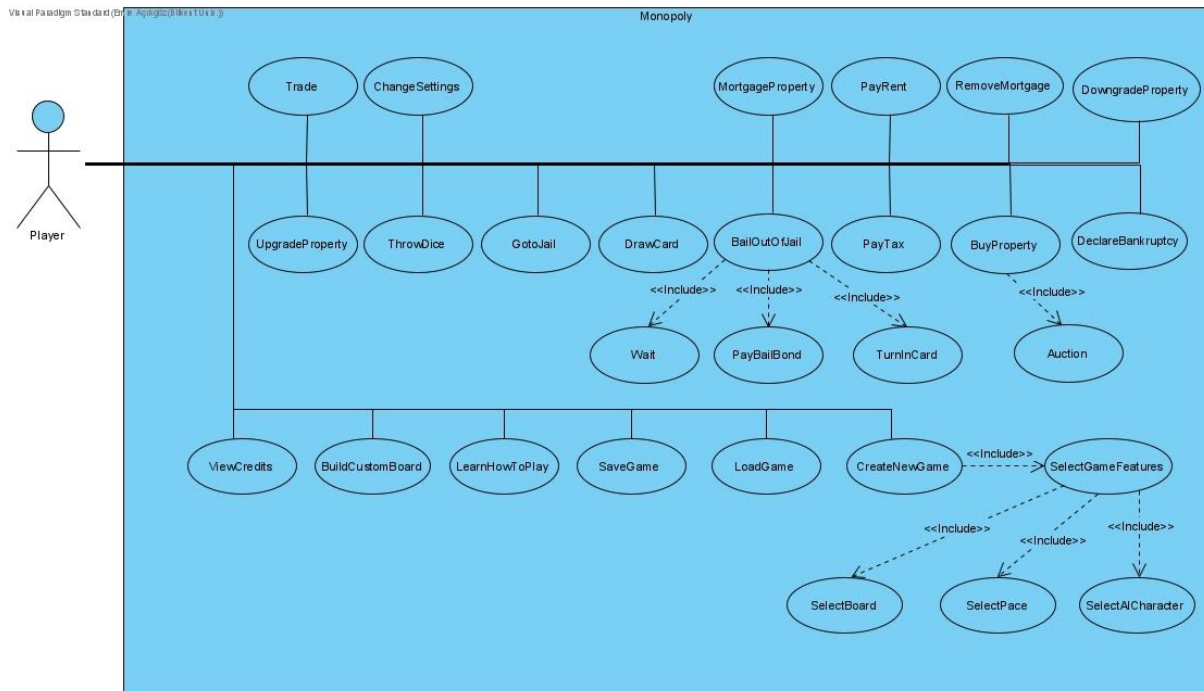The user must download the JAR file and run it to play the game.

4.8. Legal

The game is hosted on GitHub as open source.

The system is licenced with Apache License 2.0.

5. System Models

5.1. Use Case Model

Use case name: Trade

Participating actors: Initiated by Player

Flow of Events:

1- Player initiates a trade or receives a trade from another Player.

2- The system shows the TradePanel where players can see each other's properties and money and the offer being made.

3- Player makes an offer to the other Player by choosing properties and tradable cards (property card, bail out of jail card) from his/her owned properties list and adding money to the offer with AddButton by his/her preference. When the offer is complete, Player clicks on ProposeButton.

4-Otherwise, if Player who wants to make an offer changes his/her mind about doing a trade, he/she clicks on BackButton to exit the trade.

5- Player either rejects the offer by clicking the RejectButton or accepts it by clicking the AcceptButton.

6- The system does not execute possession exchange between the players in the trade if the offer is rejected, and the system closes the TradePanel.

7- Otherwise, if the other Player accepts the trade, their selected possessions are exchanged between them by the system. If there is any improved property that is going to be traded, the system sells the buildings of that property's colour group with DowngradeProperty use case and gives the money to its owner before exchange. The system completes the trade by updating the property list and balance of Player.

Entry Conditions:

Player is in the game and not bankrupt, AND

Either the seller or the buyer Player must have at least one property, AND

It must be Player's turn who initiates the trade.

Exit Conditions:

Player accepts the trade and all his/her possessions are updated by the system, OR

Player rejects the offer and the system closes the trade, OR

Player changes his/her mind about trade and clicks on BackButton and the system closes the trade.

Quality Requirements:

AI players (bots) cannot initiate a trade. The bots can either accept or reject the proposal of a human player.

Use case name: ThrowDice

Participating actors: Initiated by Player

Flow of events:

1- Player clicks on ThrowDiceButton to roll dice at the start of his/her turn.

2- The system displays the result of the dice throw on the game screen.

3- The system moves Player's token according to this result on the board clockwise. If the token passes over or lands on the GO tile, the system adds the board salary to Player's balance.

4- The system invokes a use case or executes an operation according to the tile which the token lands on. If the tile is an unowned property,

BuyProperty use case is invoked. If the tile is an owned property by another player, PayRent use case is invoked. If it is the Income Tax tile, PayTax use case is invoked. If it is a Chance Card or Community Chest Card tile, DrawCard use case is invoked. If it is a Teleportation tile, the system moves the token of Player to the next matching Teleportation tile (airport to other airport and train station to other train station). If it is the Go to Jail tile, the system invokes the GoToJail use case and does not let Player throw dice again even if the dice result is a double.

5- If the dice throw result is a double, the system lets Player throw the dice again with ThrowDice use case. If it is the third time within a single turn of the player the dice throw result is a double, the system sends the token of Player to the Jail tile with GoToJail use case and does not let Player to throw dice again.

Entry condition:

The turn is the Player's turn.

Exit condition:

The system moves the Player's token to a tile on the board.

Use case name: BuyProperty

Participating actor: Initiated by Player

Flow of events:

1- If Player chooses to buy the property at the given price specified on the tile, he/she clicks on BuyPropertyButton.

2- Otherwise, if Player does not want to buy it, then he/she clicks on AuctionButton to start an auction for the property with Auction use case.

3- The system subtracts the amount from Player's balance and adds the property to Player's owned properties list.

Entry conditions:

Player lands on an unowned property tile in his/her turn, AND

Player has enough money to cover the price of the tile he/she lands on.

Exit conditions:

The system subtracts the price of property from Player's balance and adds the property to the Player's owned properties list.

Quality requirements:

This use case **includes** Auction use case since buying a property might be done by starting an auction.

Use case name: Pay Tax

Participating Actor: Initiated by Player

Flow of Events:

1- The system hides Player's owned properties list, balance, and buildings information to allow Player to make an estimation.

2- The system prompts two options to pay the fee: 1) Pay $200, or 2) Pay %10 of your total worth.

3- Player chooses option 1 OR option 2.

4- The system calculates the tax that Player must pay.

5- If Player is not able to pay the money, Player chooses to sell his/her properties with Trade use case, mortgage his/her properties with MortgageProperty use case or sell his/her buildings with DowngradeProperty use case. If Player is still in debt after giving all of his/her properties, Player declares bankruptcy by invoking the DeclareBankruptcy use case.

6- Otherwise, if Player has enough money, he/she pays the income tax.

7- The system subtracts the fee from Player's balance.

Entry Conditions:

Player lands on the "Income Tax Tile".

Exit Condition:

System subtracts the tax money from Player's balance.

Use case name: PayRent

Participating Actor: Initiated by Player

Flow of Events:

1- System calculates the rent to be paid by checking the owner Player's property information which includes information regarding the buildings on the property and the colour-group completion.

2- If Player is not able to pay the rent, Player chooses to sell his/her properties with Trade use case, mortgage his/her properties with MortgageProperty use case or sell his/her buildings with DowngradeProperty

use case. If Player is still in debt after giving all of his/her properties, Player declares bankruptcy by invoking the DeclareBankruptcy use case.

3- Otherwise, if Player has enough money, he/she pays the rent.

4- The system subtracts the rent fee from Player's balance and adds that amount to owner Player's balance.

Entry Conditions:

Player lands on a PropertyTile that is owned by another Player and is not mortgaged.

Exit Condition:

System subtracts the rent money from Player's balance and adds that amount to the other Player's balance who is the owner of the property.

Use case name: MortgageProperty

Participating Actor: Initiated by Player

Flow of events:

1-The system displays Player's owned properties list.

2- Player selects a Property from his/her owned properties list that is not yet mortgaged and clicks on MortgageButton.

3- If Player has any buildings on the property to be mortgaged or on the color-group of it, Player must sell all of the buildings on that color-group with DowngradeProperty use case before mortgaging.

4- The system marks the property as mortgaged and adds the mortgage value to the balance of Player.

Entry conditions:

Player owns the property that he/she wishes to mortgage and the property is not already mortgaged.

Property to be mortgaged and every Property on that color-group must not have any buildings on them.

It must be Player's turn who wishes to mortgage.

Exit condition:

The system adds the mortgage value of property to Player's balance to complete mortgaging.

Use case name: DrawCard

Participating actor: Initiated by Player

Flow of events:

> 1- The system shows the first card from the deck whose type is defined by the player's landing tile. If Player lands on the Chance Card Tile, the system displays a chance card to Player, similarly if the player lands on the Community Chest Card Tile, the system displays a community chest card to Player.

2- If the Card that the system displays to Player is Bail out of Jail Card, the player saves it for later use to get out of jail with BailOutOfJail use case. Otherwise, the player reads the card instruction then performs the instruction written on the card.

> 3- If the card demands Player's token to move to another tile, system moves Player's Token to that tile. If the card is a bonus payment card, the system adds the amount written on the card to Player's balance. If the card asks for Player's Token to be moved to Jail Tile, GoToJail use case is invoked. If the card is a debt card, the system subtracts the asked amount from Player's balance.

4- If Player cannot pay debt on the card, Player chooses to sell his/her properties with Trade use case, mortgage his/her properties with MortgageProperty use case or sell his/her buildings with DowngradeProperty use case. If Player is still in debt after giving all of his/her properties, Player declares bankruptcy by invoking the DeclareBankruptcy use case.

> 5- The system returns the card to the bottom of the deck from which the card is drawn unless it is Bail Out Of Jail Card.

Entry condition:

Player lands on the Chance Card Tile or Community Chest Card Tile.

Exit Condition:

System returns the card to the bottom of the deck to which it belongs, OR Player saves the Bail Out of Jail Card.

Use case name: DeclareBankruptcy

Participating actor: Initiated by Player

Flow of events:

1- Player declares bankruptcy by clicking DeclareBankruptcy Button.

2- The system sells his/her properties from the value calculated from title deed cards. If Player has to pay some money to another player, the system gives to the other player %50 of the money gained from these sold properties.

3- Player resigns from the game and is unable to play his/her turn again on this game session.

4- The system removes Player from the game. If Player is the only human player of the game, the system ends the game.

Entry condition:

Player has debt more than he/she can pay either to another Player or to Bank, OR

Player chooses to declare bankruptcy.

Exit condition:

System removes Player from the game, OR

System ends the game.

Use case name: UpgradeProperty

Participating actors: Initiated by Player

Flow of events:

1-System shows the owned properties list of Player.

2- Player selects the property to be upgraded (improved) and clicks on the UpgradePropertyButton.

3- System checks whether Player who is trying to upgrade their property is trying to upgrade evenly. If not, the system rejects the upgrading operation.

4- If Player has enough money for an upgrade and the property is still open for an upgrade, the system takes the money needed for an upgrade from Player and sets the house or hotel to the property depending on the upgrade level. Otherwise, if Player does not have enough money, the system does not perform the upgrade.

Entry conditions:

Player owns every property in the same color-group of the property to be upgraded.

Property that Player wishes to upgrade or any property in its colour group is not mortgaged.

It is Player's turn who wishes to upgrade property.

Exit Conditions:

System performs the upgrade, OR

System checks if Player is trying to upgrade property evenly or not, if the upgrade is not made evenly, the system does not perform the upgrade.

System does not perform the upgrade due to not enough money of Player.

Use case name: Auction

Participating actors: Every player that is playing the game.

Flow of events:

1- Player who has the turn does not buy the unowned property.

2- The system opens the AuctionPanel for all Players.

3- Every player in the game can raise their offered price (bid). Player cannot bid more than the amount in his/her balance.

4- The system updates the current price according to the prices that players give.

5- The system ends the auction when the remaining time reaches 0.

6- Player who makes the highest bid buys the property.

7-The system subtracts the highest bid amount from the highest bidder Player's balance and adds the property to that Player's owned properties list.

Entry conditions:

Player lands on an unowned property tile and chooses not to buy it at the price that is written on the tile.

Exit conditions:

System subtracts the highest bidding amount from the highest bidder Player's balance and adds the property to that Player's owned properties list.

Quality requirements:

When the auction starts, the first raise must be made in 25 seconds.

When a raise is made, the auction time increases by 10 seconds.

Every raise is the 50% of the minimum sale price automatically.

Minimum sale is the 50% of the sale price of the property.

Use case name: GotoJail

Flow of events:

        1- The system moves the token of the player to the Jail tile. If Player's token passes over the Go Tile on the way, the system does not give money.

Entry Conditions:

Player rolls double dice for 3 rounds in a row, OR

Player lands on the "Go to Jail" tile, OR

Player draws a "Go Directly to Jail" chance card, or "Go Directly to Jail" community chest card.

Exit Conditions:

System moves Player's token to the Jail tile.

Use case name: BailOutOfJail

Participating actor: Player

Flow of events:

        1- The system shows the options of Player that they can choose. Options are as follows: Player can wait in jail by invoking Wait use case. If they have already waited for 3 rounds, they are out of jail immediately. They can pay a $50 fine to the bank by invoking PayBailBond use case. If they have a "Get out of Jail" community chest or chance card, they can turn in these cards with TurnInCard use case. They can throw dice with ThrowDice use case, if the dice rolled are double, they continue playing. If they throw the dice for 3 rounds when they are in jail and none of them are double, at the end of the third round, they pay $50 to the bank and continue playing.

2- The player chooses an option defined above, and plays accordingly.

Entry conditions:

Player's token is on Jail tile.

Exit condition:

The player chooses an option defined and performs the action according to the chosen option.

Quality requirement:

This use case **includes** Wait, PayBailBond and TurnInCard use cases. These use cases are initiated when Player makes a choice about how to bail out of jail as stated in the flow of events.

Player must choose an option in a predefined seconds.

Use case name: ChangeSettings

Participating actor: Initiated by Player

Flow of Events:

1- Player opens the Change Settings menu by clicking on ChangeSettingsButton.

2- The system brings a panel that includes settings about volume of sounds.

3- Player chooses configurations then leaves this screen.

Entry condition:

Player opens the Change Settings Menu.

Exit condition:

Player closes the Change Settings Menu.

Quality Requirement:

Saved configurations must take place after Player exits.

Use case name: LearnHowToPlay

Participating actor: Initiated by Player

Flow of Events:

1- Player opens the How To Play screen by clicking on HowToPlayButton.

2- Player reads the How To Play text.

3- After reading the text, Player exits the menu by clicking on BackButton.

Entry condition:

Player opens the How To Play screen.

Exit condition:

Player closes the How To Play screen.

Use case name: BuildCustomBoard

Participating actor: Initiated by Player

Flow of Events:

1- Player opens the Map Builder screen by clicking on MapBuilderButton.

2- Player changes the names, prices, rents, property values, or mortgage values of the properties and board name and board salary.

3- After making the desired changes, Player exits the menu by clicking BuildButton. If Player does not want to keep changes, Player exits the menu by clicking on BackButton.

4- The system saves the board to the board list.

Entry condition:

Player is on the main menu screen.

Exit condition:

Player closes the Map Builder Menu by clicking on BuildButton or BackButton.

Quality Requirements:

Board should be saved within 5 seconds by the system.

Use case name: CreateNewGame

Participating actor: Initiated by Player

Flow of events:

1- Player selects the "Create New Game" option on the Menu.

2- The system responds by prompting Player to select the features of the new game by invoking SelectGameFeatures use case.

3- Player presses the "Start Game" button to start the new game.

4- The system starts a new game with the features specified by Player.

Entry condition:

Player is on the main menu screen.

Exit condition:

Player finishes creating a new game, OR

Player cancels the creation by selecting the BackButton.

Quality Requirements:

Selected features must be applied to the game.

This use case **includes** SelectGameFeatures use case for specifying game modes.

Use case name: LoadGame

Participating actor: Player

Flow of events:

1- Player selects the "Load Game" option on the menu.

2- The system presents the list of saved games to Player.

3- Player selects the desired game by clicking on it then presses the StartButton.

4- The system loads and starts the selected game.

Entry condition:

Player is on the main menu screen.

Exit condition:

The system loads the saved game, OR

Player cancels the load by selecting the BackButton.

Quality Requirements:

The system must load the game within 3 seconds.

Use case name: ViewCredits

Participating actor: Player

Flow of events:

1. Player selects the "View Credits" option on the main menu.

2. The system displays the credits information text to Player in a panel.

Entry condition:

The Player is on the main menu screen.

Exit condition:

The Player selects the "Back" option on the "View Credits" screen.

Use case name: SelectGameFeatures

Participating actor: Initiated by Player

Flow of events:

1- System opens up a panel that shows the number of players options, AI characteristic option, game pace specification, and a board option to be played on.

2- Player selects or changes these options to his/her preference. Player specifies AI characteristic by invoking SelectAICharacter use case. Player specifies game pace by invoking the GamePace use case, and selects board by invoking SelectBoard use case.

4- Player exits the Select Game Features screen by clicking StartGameButton or BackButton.

Entry condition:

Player has invoked CreateNewGame use case.

Exit condition:

The system complies the preference of Player to game which are AI characteristic, game pace specification, board, and number of players (bots and human).

Quality Requirements:

The selected features must be saved within 3 seconds.

This use case **includes** SelectAICharacter, SelectPace and SelectBoard use cases for specifications.

Use case name: DowngradeProperty

Participating actor: Initiated by Player

Flow of Events:

1- If there is one building on the property to be sold, Player can freely sell this building, or else, if there are houses in the properties of the same color-group, houses must be sold one by one from these properties from the same color-group. Every building can be sold at the half price of the house and hotel price.

2- The system gives to Player the price of the house or hotel that is being sold.

Entry conditions:

Property that Player wants to downgrade has been upgraded at least once and is not mortgaged.

Player wants to sell a house, houses or hotel on a property by downgrading.

Exit conditions:

Player waits for the system to take the house or hotel, and upgrade the current money of the player.

Quality requirements:

The system must get the house or hotel from the player and give the money to the player within a second.

Use case name: RemoveMortgage

Participating actor: Initiated by Player

Flow of Events:

1- Player clicks on RemoveMortgageButton after selecting the property.

2- If Player has enough money and the property was mortgaged by the same Player, the system subtracts the mortgage amount of the property plus the 10% interest rate. If Player has enough money and he/she wants to lift a mortgage off a property he/she has just bought from a trade (new owner), the system subtracts the mortgage value plus 10% interest rate. If Player is the new owner, has enough money and decides not to lift the mortgage immediately after the trade but wants to lift it later in the game, the system subtracts the mortgage value amount plus 10% interest rate and plus an additional 10% interest rate.

3- The system lifts off the mortgage from that property.

4-Otherwise, if Player does not have enough money, the system does not perform lifting off the mortgage.

Entry Conditions:

Player has the turn who wishes to lift the mortgage, OR

Player buys a mortgaged property from another Player in a trade and wishes to lift the mortgage immediately, OR

Player buys a mortgaged property from another Player in a trade and wishes to lift the mortgage later in the game.

Exit Conditions:

System lifts the mortgage off a property of Player at a price, OR

System does nothing due to not enough money of Player.

Use case name: SaveGame

Participating actor: Initiated by Player.

Entry Condition: The game starts.

Player clicks Menu button while playing the game, and clicks Save Game Button.

Flow of Events:

1- After Player clicks Save Game Button, system saves every current

game feature including properties and their prices, owners of the properties, money and title deed cards of every player, locations of the tokens of players and turn information. The saved game will have the name of the current date and hour.

Exit Condition: Player waits for the system to save the game.

Quality Requirements: Saving operation should be done in less than 3 seconds.

Use case name: Wait

Participating actor: Initiated by Player

Flow of events:

　　1. Player chooses to wait in jail.

　　　　2. The system ends the turn of Player, who does nothing in his/her turn, and the next Player continues to play.

Entry condition:

　　Player is in jail and chooses the "Wait" option in jail.

Exit condition:

　　After clicking the wait option, the Wait use case is automatically terminated and the Player's turn passes.

Quality requirements:

　　Player can choose to wait in jail for at most 3 rounds consecutively.

Use case name: PayBailBond

Participating actor: Initiated by Player

Flow of events:

　　1. Player chooses the Pay Bail Bond option when they are in jail.

　　　　2. The system checks if Player has at least $200. If yes, the system approves the action, else the system rejects the action.

　　3. If approved, the player continues to play by rolling dice, otherwise, Player continues to wait in jail.

Entry condition:

　　Player is in jail and chooses to pay $200 to get out of jail.

Exit condition:

　　Player gives $200 OR,

　　Player has no $200 and system does nothing.

Quality requirements:

Player should have $200 at least in his/her balance.

Use case name: TurnInCard

Participating actor: Initiated by Player

Flow of events:

> 1. The system checks if Player has a 'Get out of Jail Free' card. If yes, the system approves the action and takes the card from Player, else
> the system rejects the action.

> 2. If approved, Player gives 'Get out of Jail Free' card and continues to play by rolling dice, otherwise, Player waits in jail.

Entry condition:

> Player is in jail and chooses Turn In Lucky Card option when the system asks.

Exit condition:

> Player gives 'Get out of Jail Free' card OR,
>
> Player has no 'Get out of Jail Free' card.

Use case name: SelectAICharacter

Participating actor: Player

Flow of events:

> 1. Player selects or changes a character for the AI that will behave accordingly in the game.

Entry condition:

> Player is on the Select Game Features screen.

Exit condition:

> An AI character is selected.

Use case name: SelectPace

Participating actor: Player

Flow of events:

1. Player selects or changes the pace of the game.
2. Once in the game, features concerning the length of the game are adjusted according to this selection.

Entry condition:

> Player is on the Select Game Features screen.

Exit condition:

> A game pace is selected.

Use case name: SelectBoard

Participating actor: Player

Flow of events:

1. Player selects or changes the dataset used for initializing the board. Players will be playing on that board when they start the game.

Entry condition:

Player is on the Select Game Features screen.

Exit condition:

A BoardDataset is selected.

5.2. Dynamic Models

5.2.1. Sequence Diagrams

5.2.1.1. General Sequence of A Turn

Scenario: Player plays a turn.

Overall sequence diagram of how a turn is played by a player.
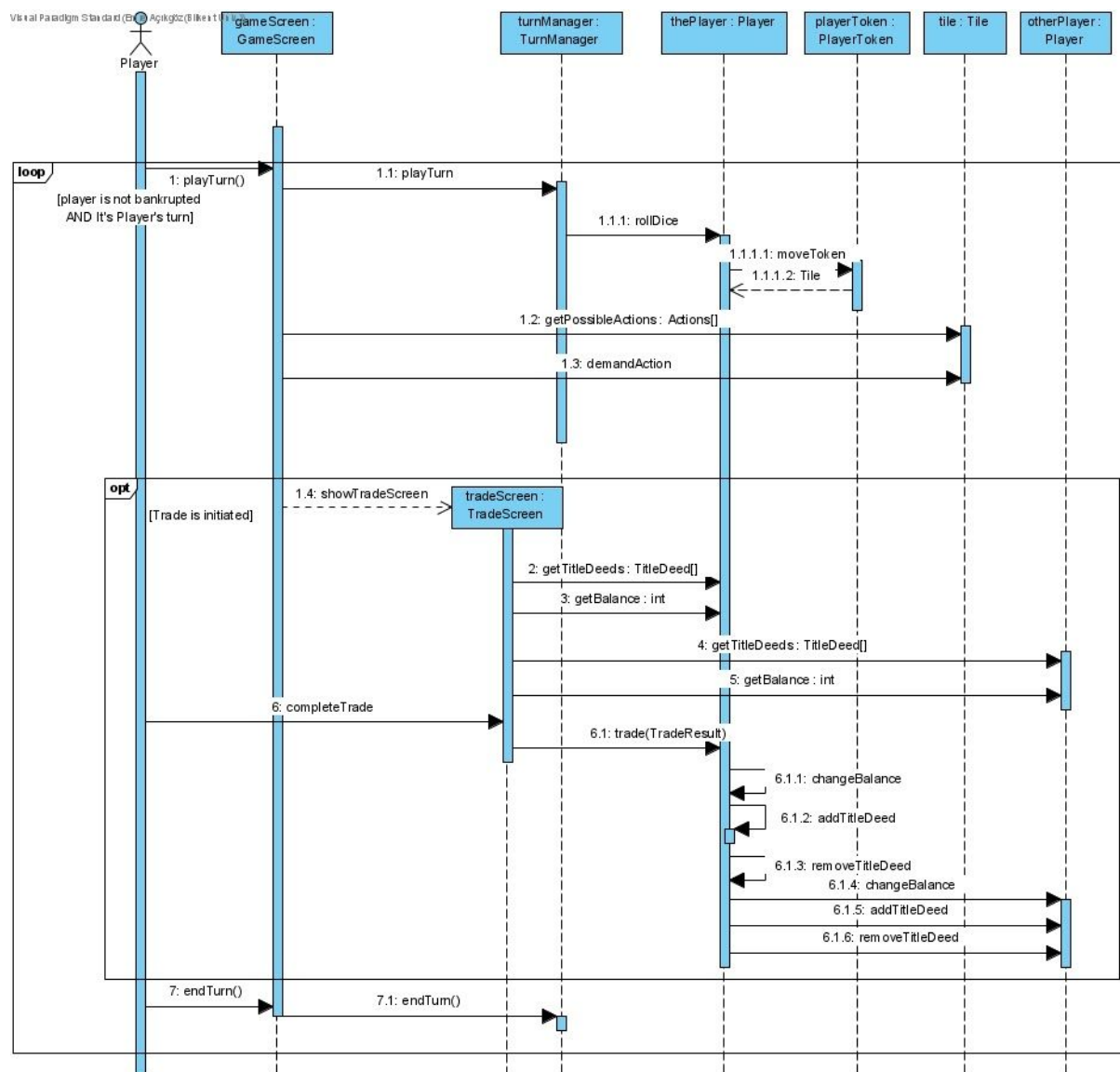
5.2.1.2. Sequence Diagram for Player Landing on Card Tile

Scenario: Player lands on the "Chance Card" tile. Player draws a card from the deck. Then, the player performs the instruction written on the card.
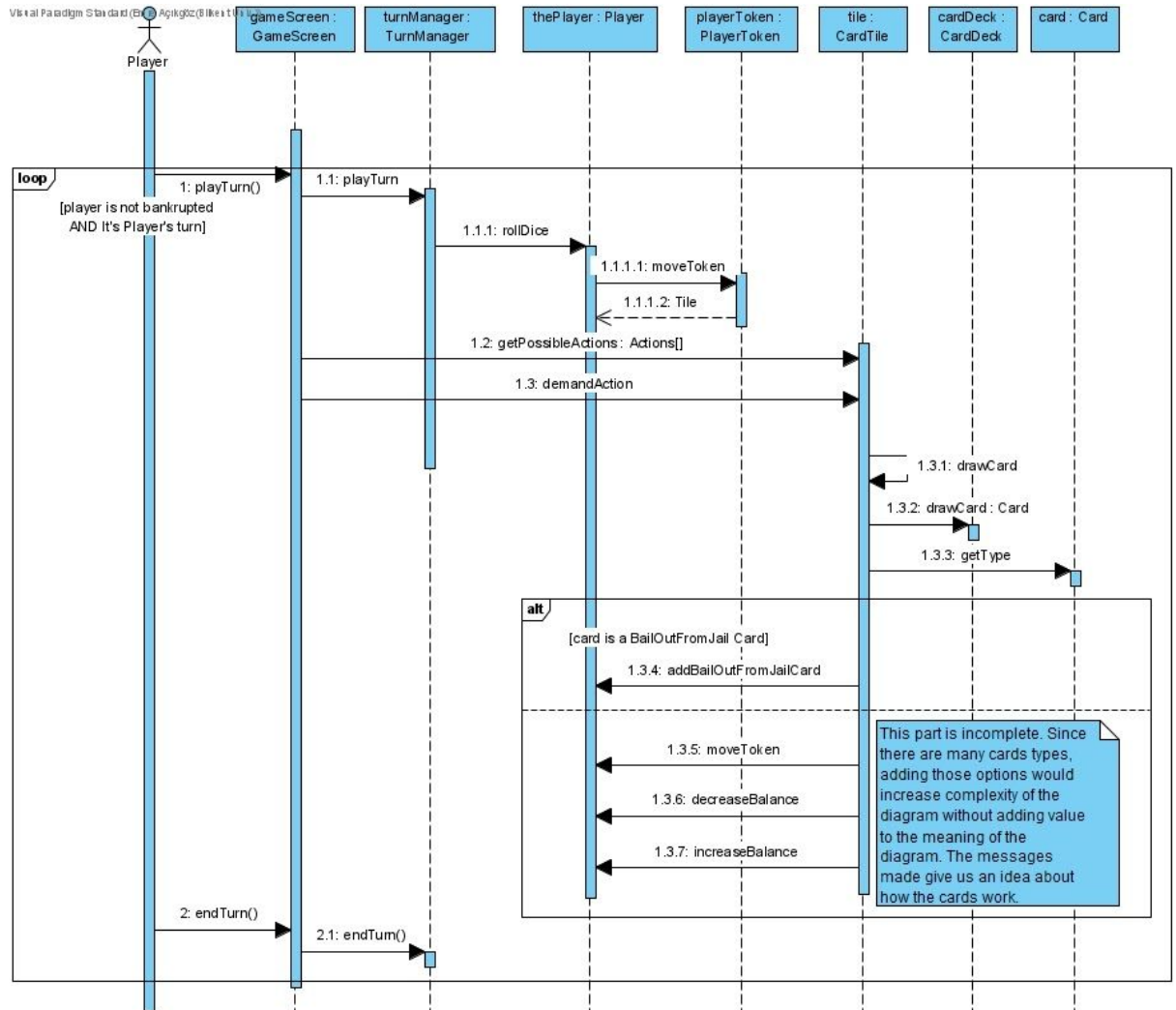
5.2.1.3. Sequence of Events for Landing on an Income Tax Tile

Scenario: After rolling dice, Player lands on an Income Tax Tile where they should pay the predefined fine to the bank.
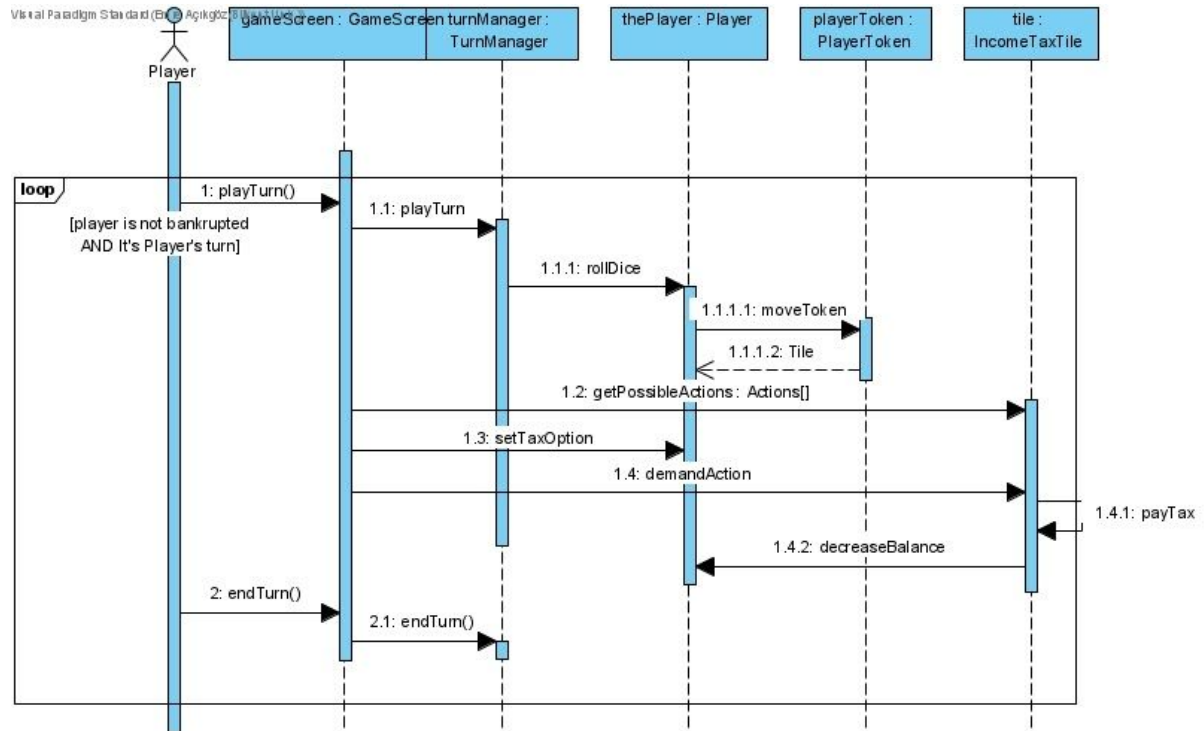
5.2.1.4. Sequence of Events for Player Landing on a GoToJailTile

Scenario: Player lands on the GoToJailTile.
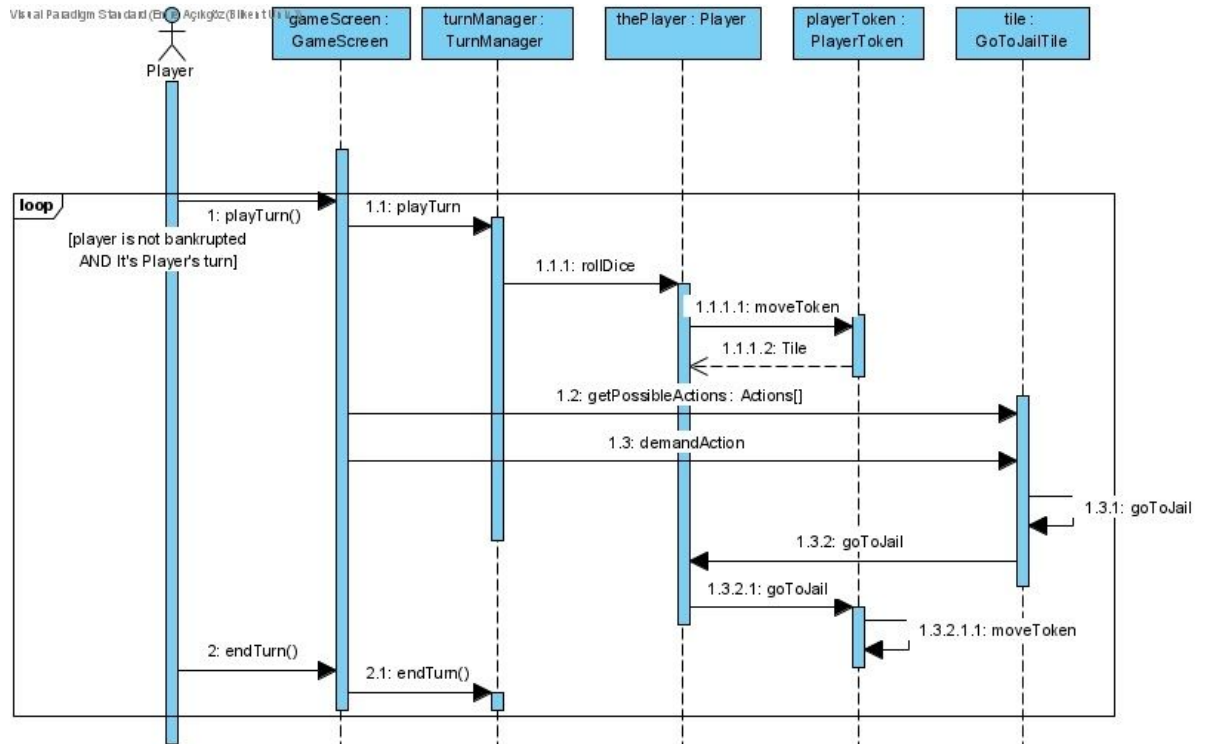
5.2.1.5. Sequence of Events for a Jailed Player Bailing Out of Jail

Scenario: A player (due to any reason stated in the section 2.12) is thrown to jail.

After a player enters jail they can wait to get out of the jail for free, they can roll double dice three times, they can use a bail out of jail card, or they can pay the fee for getting out of jail.
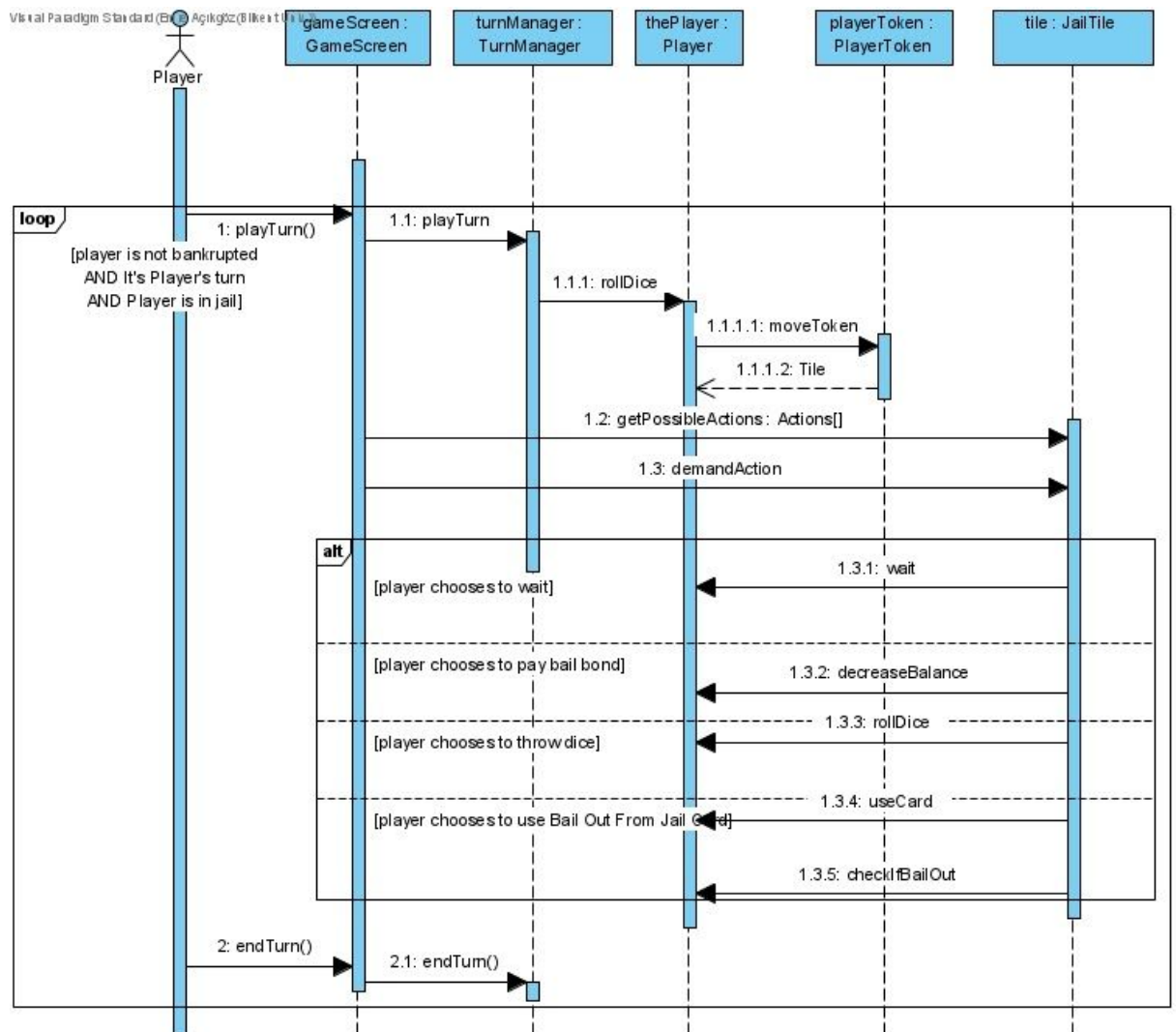
5.2.1.6. Sequence Diagram for Property Actions

Scenario Player displays its own properties. Then, Player chooses a title deed card. Then mortgages a property. The system calculates the mortgage value and downgrades the property and then adds this value to Player's balance.
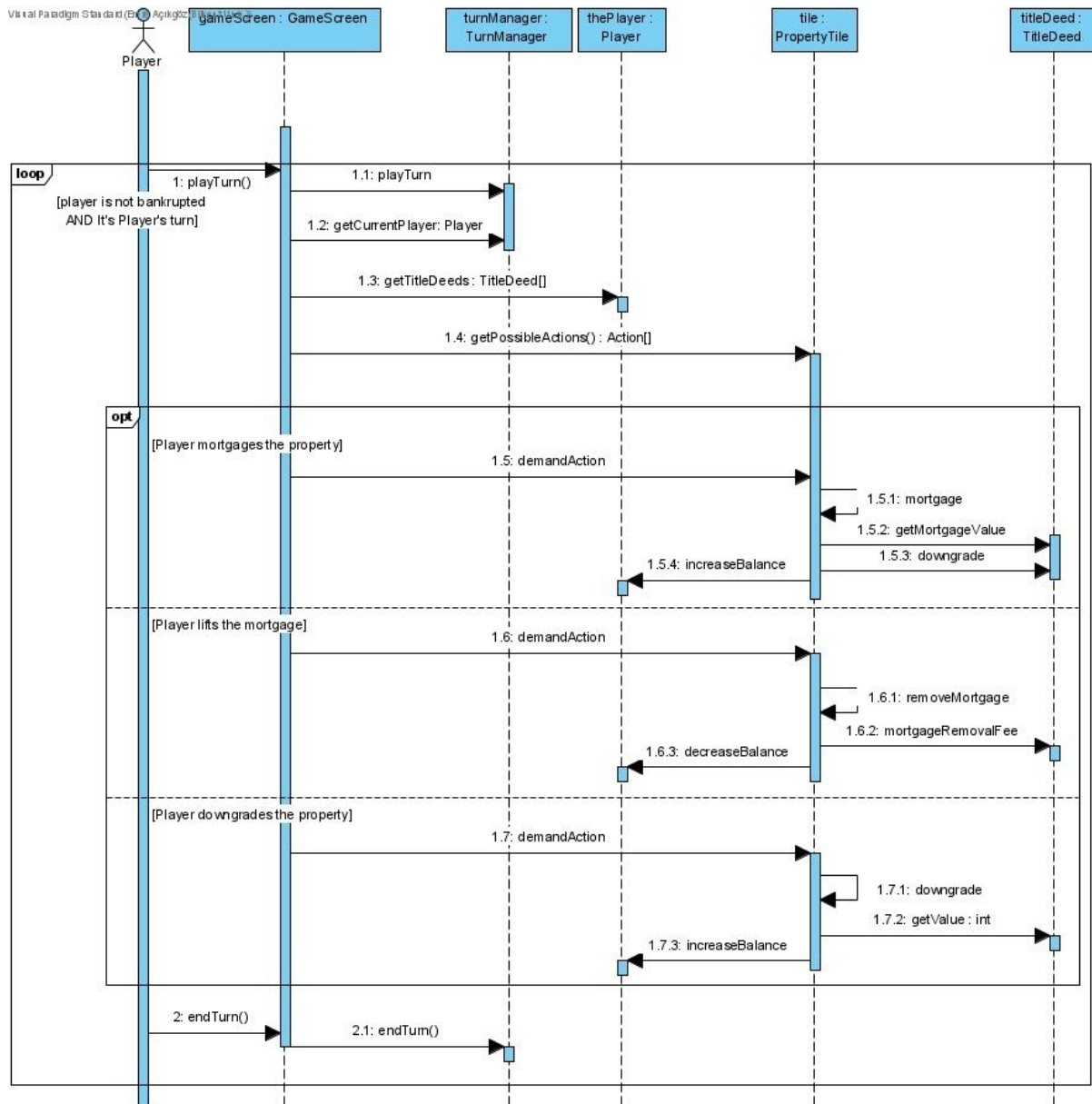
5.2.1.7. Sequence Diagram for a Player Landing on a PropertyTile

Scenario: Player lands on a PropertyTile.
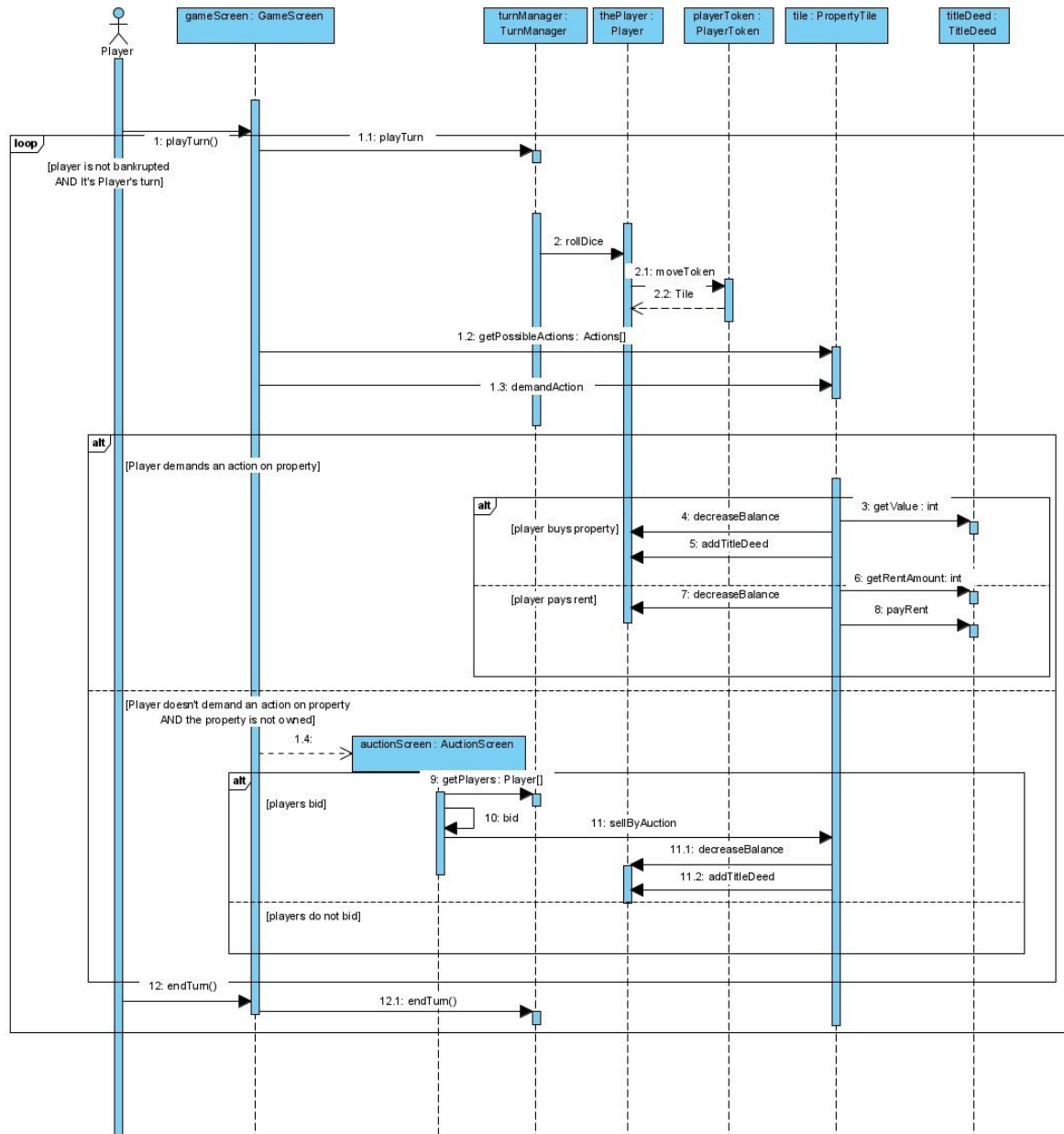
5.2.1.8. Sequence Diagram for Player Landing on a TeleportTile

Scenario: Player lands on a TeleportTile.
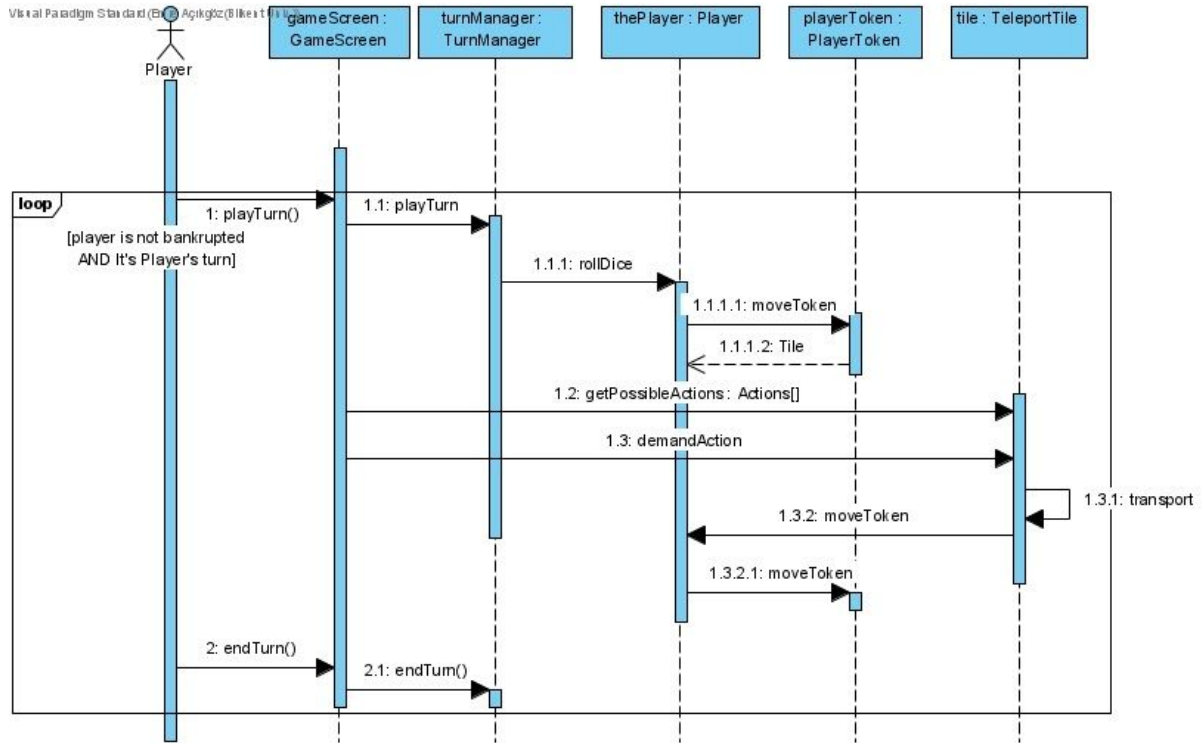
5.2.1.9. Sequence Diagram for Custom Board Building

Scenario: Player opens the custom board building screen.
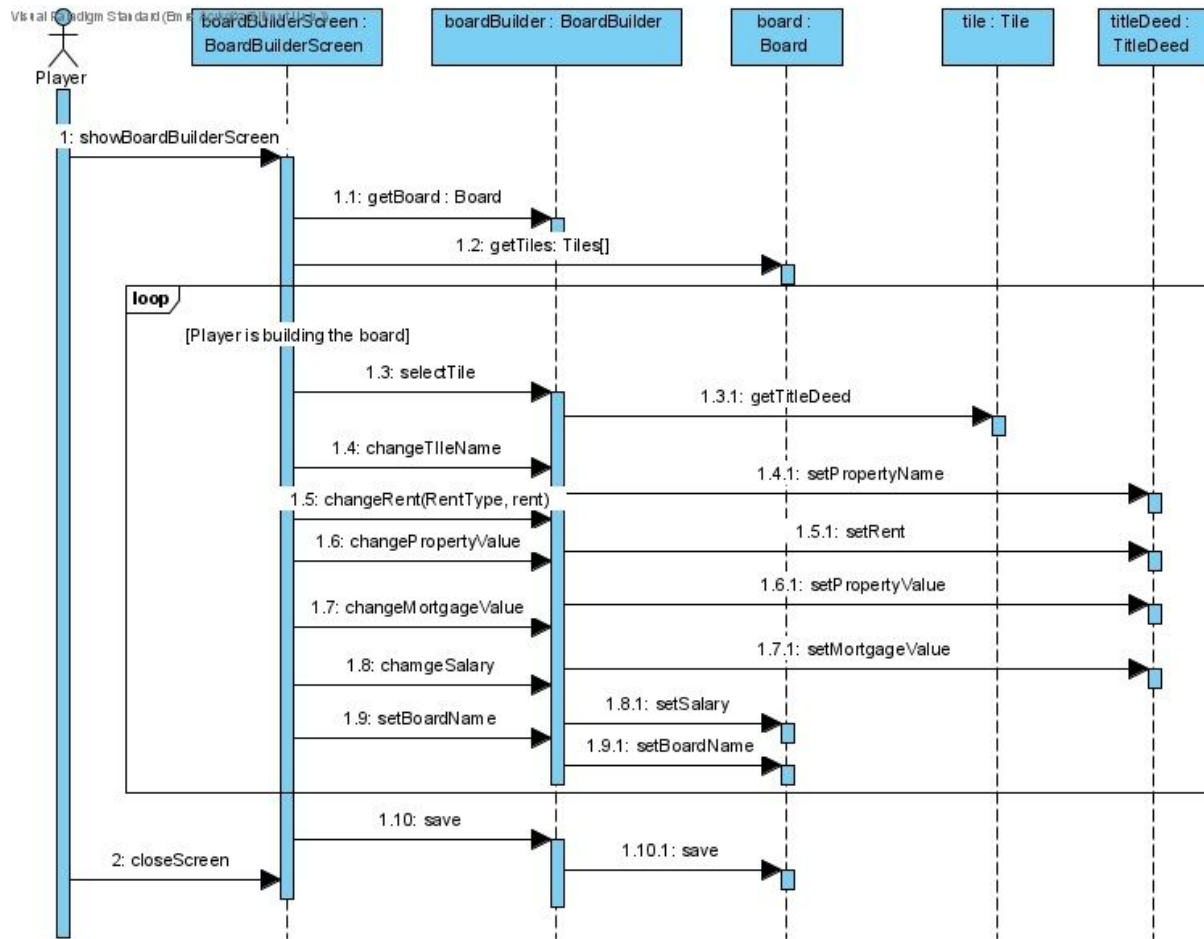
5.2.2. State Transitions Diagram
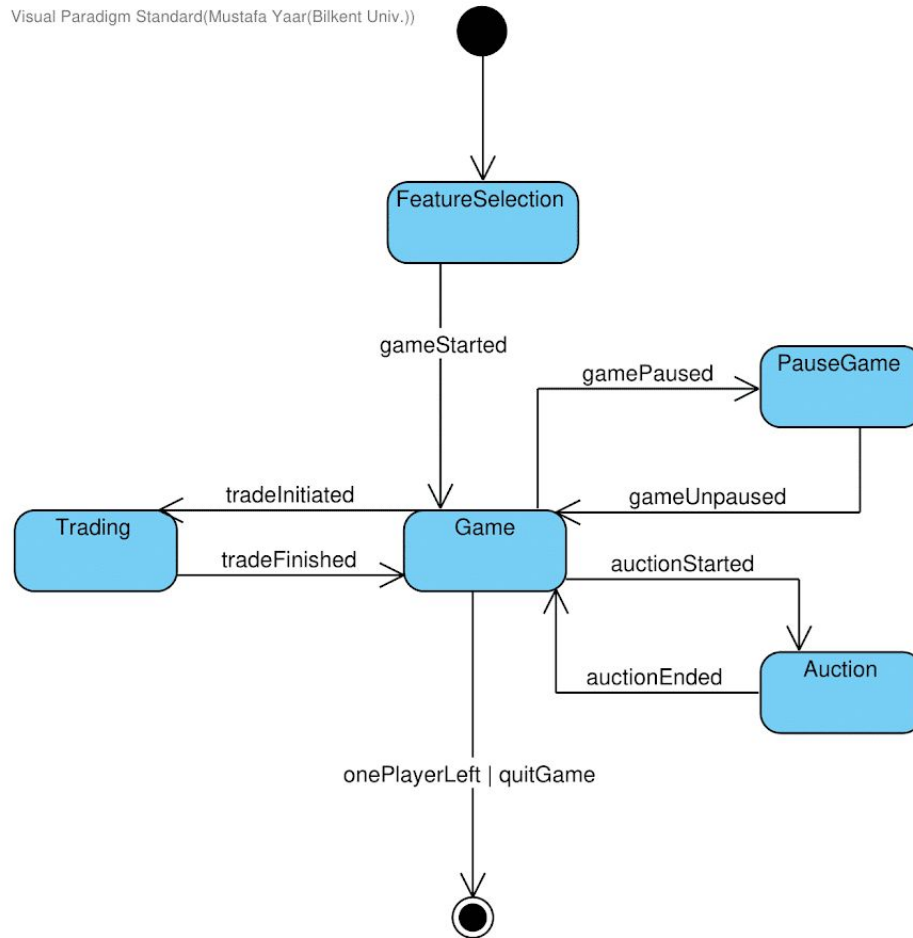
The above state diagram shows the different major states of the game. The initial state is the FeatureSelection which determines the general settings of the board and the game. FeatureSelection state directly interacts with the Game state. Game state handles the most of the logic thus, interacts with all the different states such as the Trading, the Auction, and the PauseGame state. The state diagram ends with the Game state.

5.2.3. Activity Diagram
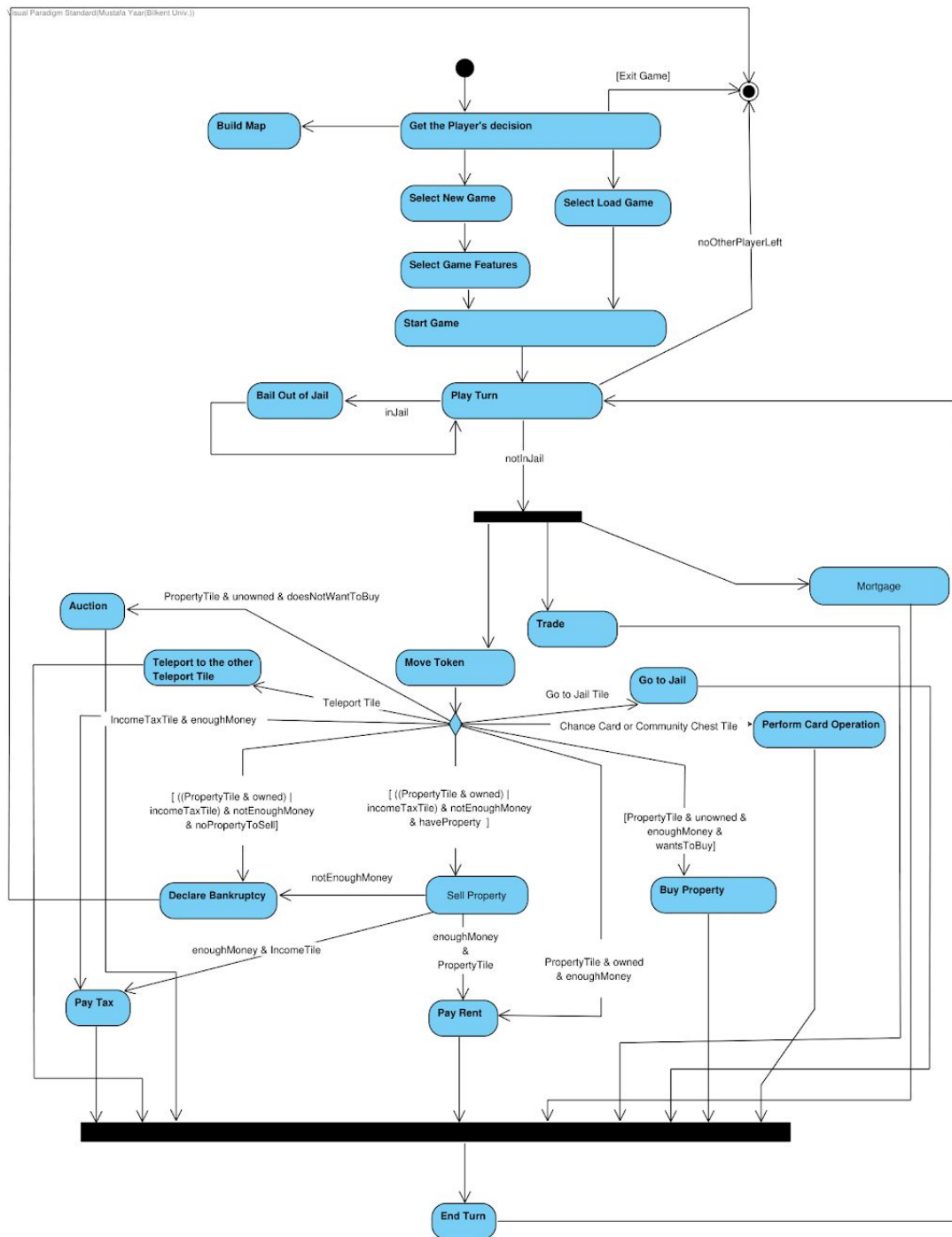
Activity Diagram Explanation:

When the player starts the Monopoly game, the system waits for the player's choice. The player can start a new game, load a game, exit the game or build a new map (board).

If the player starts a new game, he/she should adjust the game features. He/She should select the number of players and the number of bots. If the player plays with the bots, he/she should select the characteristics of bots, and he/she should select the board theme and the game starts.

If the player selects the load game option, he/she should select from the database which saved game he/she wants to play. The game loads every feature of the saved game such as the properties and balance of the players, round number, token positions, etc. After loading the game, the game starts from where it was left off.

If the player selects the build map option, they are directed to the map builder screen where the player is able to change the names of the properties, prices of the houses and hotels and rents of these properties, and the money they get when they pass the Go tile.

If the player selects the exit game option and confirms the operation in the exit screen, the game is closed by the system and it goes to the final node.

When the game starts, players should play their turn one by one. If the player who is to play their turn is not in jail, he/she can start a trade, move their token by throwing dice or do mortgage operations on a property. The player can do these activities in any order before ending the turn. If the player wants to trade, he/she selects the player who they want to trade with, makes an offer, and waits for the other player's response. If the player wants to mortgage a property of his/hers or lift off a mortgage, then he/she selects a property to do so. The player continues to play in his/her turn by throwing dice and moving their token on board. After moving

his/her token, the player must perform other actions according to the tile on which he/she lands.

If the player lands on a teleport tile, he/she teleports to the corresponding teleport tile.

If the player lands on the Go to Jail tile, the system moves his/her token to the jail tile.

If the player lands on a Chance or Community Chest tile, the system shows the card to the player, and the player performs the operations the card says.

If the player lands on a property tile, and this tile is not owned, and the player who lands on it does not want to buy it, the auction starts. Auction is performed as explained above.

If the player lands on a property tile, and this tile is not owned, and the player wants to buy that property, if they have enough money, they buy this property from the bank.

If the player lands on an owned property or an income tax tile, and he/she does not have enough money to pay the rent, or the tax, if he/she has a property, he/she must sell that property to a player to pay the tax or the rent. After selling property, if the player still does not have enough money, he/she declares bankruptcy and exits the game. After selling property, if the player has enough money and landed on an Income Tax tile, he/she pays the tax. If the player landed on a property tile and has enough money, he/she pays the rent of the property. If the player does not have a property to sell, he/she immediately declares bankruptcy and exits the game.

If the player lands on an owned property and has enough money for the rent, they pay the rent.

If the player lands on an income tax tile and has enough money for the tax, they pay the tax.

If the turn of the player starts and there are no other players, the system goes to the final node and the game ends.

If the player who is to play their turn is in jail, he/she has to first perform bail out of jail operations. If the player has successfully gotten out of jail, he/she continues playing by throwing dice and moving his/her token.

After the player plays a turn by doing these activities, the turn ends for that player until the turn comes back to the player again.

5.3. Object and Class Models
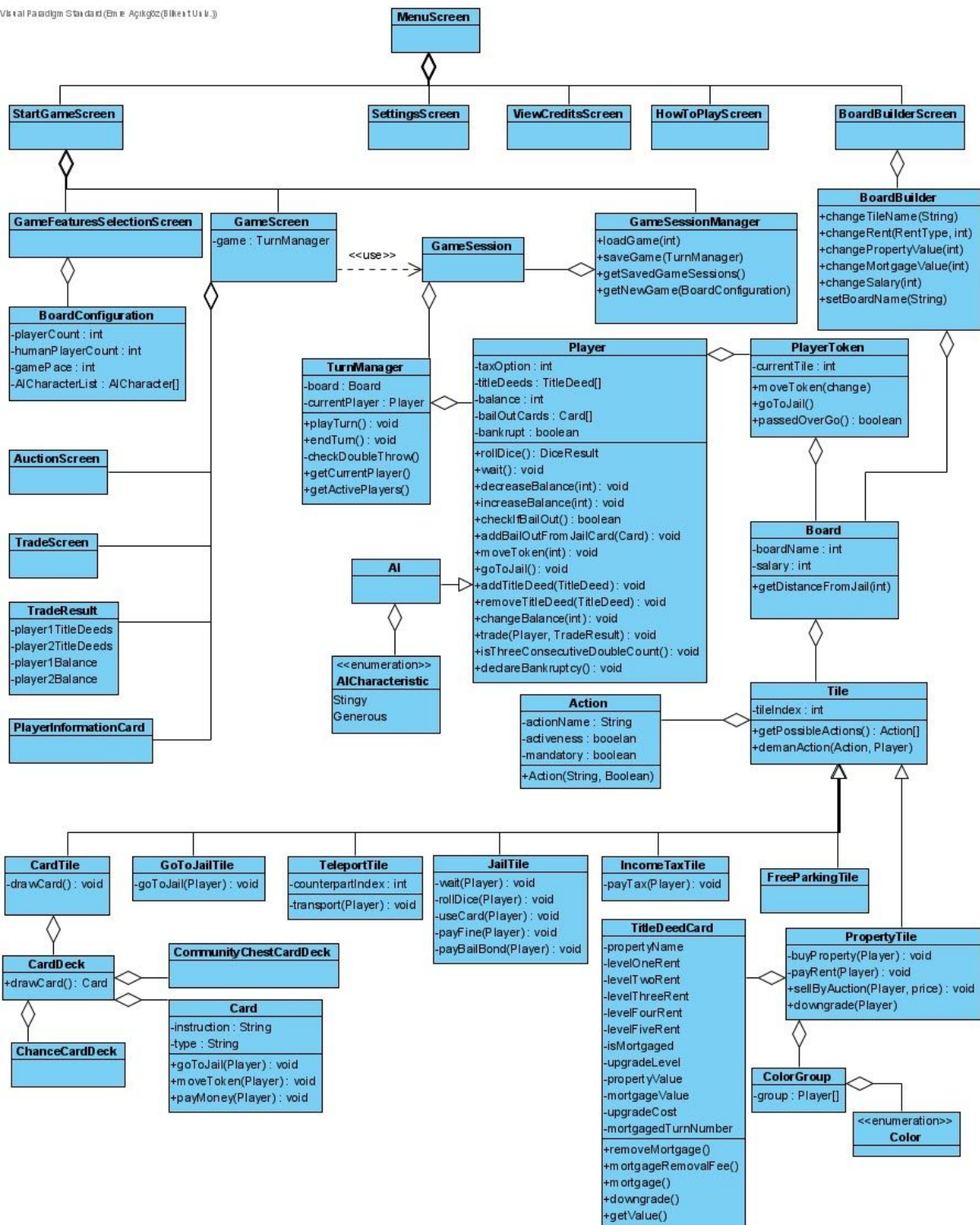
Classes related to UI are GameScreen, StartGameScreen, MenuScreen, BoardBuilderScreen, HowToPlayScreen, ViewCreditsScreen, SettingsScreen AuctionScreen, TradeScreen, PauseGameScreen. The design and the purposes of each of these classes are further expanded in the UI mockups, under section 5.4.

Application Domain Classes

TurnManager: This is the central class for game logic. It provides the communication between the *Player* and its related classes and the *GameSession* and its related classes. It manages what happens at each turn, and consequently communicates with *GameSession* accordingly for updating the game's state.

GameSession: Holds the current state of the game, how many players are still standing and such. This class is initialized using the *GameSessionManager*.

GameSessionManager: Handles save game and load game features. It takes the current state of the game from the *GameSession* class.

Player: *Player* class represents both human players and *AI* players. *Player* class holds the current state of the player themselves. To be more specific, it holds its current trade partner, balance, and properties. It also holds the current state of the player object, such as the jail status, or if the player has a get out of jail card.

PlayerToken: *PlayerToken* is the player's representation on the board. It holds the index of the current tile it is on. Players' moving on the board is represented using this class' movePlayer method.

Board: Board shows all the tiles, that include PropertyTiles, CardTiles, TaxTiles and such. Board is fundamentally a tile array that the Players move and interact on.

BoardBuilder: This class will handle the board settings, such as the tiles' rents and their names. These events will take place in the board builder menu.

DiceResult: Holds the values rolled by dice. Each turn, this class determines how many tiles the player will proceed on the board. From the perspective of usefulness, this class has a lot of use.

AI: This class will represent the non-human players in the game. This class' constructor will take different characteristics. According to those characteristics the playTurn() method will be overridden from AI's superclass *Player*.

AICharacteristic: This is an enumeration that specifies the character of the AI. The *AI* class makes decisions with a rationale based on what this class represents.

Action: This class is a representation of all of the tile or card based actions in the game, such as buying or selling property. It checks their names, whether the action is being processed and whether the users are mandated to do the particular action.

Tile: This is the class that is used by UI classes. This class gives the actions that are represented by the tile. With or without user's interaction, the tile may demand an action on the player. These actions specified in use cases. The following *CardTile*, *TeleportTile*, *TaxTile*, *PropertyTile*, *GoToJailTile*, *FreeParkingTile* classes are subclasses of *Tile* class.

CardTile: This class is a subclass of *Tile* class. This class has a reference to *CardDeck* class. When a player lands on a *CardTile*, the player draws a card. When draw card action is demanded, this class applies the instructions written on the drawn card.

TeleportTile: This class is a subclass of *Tile* class. There are 2 pairs of teleport tiles on the map. When *Player* lands on a *TeleportTile*, the *PlayerToken* index is changed to the index of the counterpart *TeleportTile* on the board.

TaxTile: This class is a subclass of *Tile* class. When *Player* lands on a *TaxTile*, the money corresponding to the amount of tax is deducted from *Player's* balance.

PropertyTile: This class is a subclass of *Tile* class. When the *Player* lands on any *PropertyTile*, the player will see the buy property pop up screen. If the player buys the property the title deed card of the corresponding property will be added to the players inventory. If the player does not have enough money then an auction will take place where each of the players will have the chance to increase their offer and buy the property. In addition to the buying functionality of the *PropertyTile* class, there will also be a sell property option where the property can be sold to another player, also said property can be mortgaged to the bank.

GoToJailTile: When *Player* lands on the *GoToJailTile* their token will be directly sent to the *Jail* tile.

FreeParkingTile: FreeParkingTile is a tile that is not connected to any actions. Player landing on this tile completes his turn without doing anything further.

JailTile: This class will be used to place the players in the ingame jail, if the players are to land on the *GoToJailTile*. There are four main ways for a player to bail out of the jail. They can wait for three turns, they can pay the fee to get out, if they throw double dice for three times they can get out or if they have a *bailOutOfJail* card they can turn it in to get out.

Card: *Player* draws a *Card* object when he lands on a *CardTile*. Each *Card* object contains one task the drawing player must fulfil, or he declares bankruptcy. There are 2 classes that hold *Card* arrays, the *ChanceCardDeck*, and the *CommunityChessCardTile*.

CardDeck: This class holds the cards of two kinds. It enables the player to draw a card from the set of cards. Manages the lifecycle of the cards. If a card is drawn, it is supposed to add it again except the "Bail out of Jail" card. Since there is not a distinct separation between community chest cards and chance cards, this separation is done using the subclasses ChanceCardDeck and CommunityChestCardDeck.

ChanceCardDeck: This class holds the array of the chance cards..

CommunityChestCardDeck: This class holds the array of the community chest cards.

ColorGroup: Properties in the same *ColorGroup* have reference to the same *ColorGroup* object, that has the same Color enumeration. One of this class' uses is to easily track whether a player has collected all the properties having the same Color enumeration, which is done in Player class but is tracked using this class.

TitleDeedCard: Stores the associated property's information such as rent or price. *PropertyTile* class performs operations by using the property information in this class.

Color: *Color* is an enumeration that is common for a particular *ColorGroup*.
TradeResult: Stores information about the exchanged items between players.

5.4. User Interface

5.4.1 Navigational Path
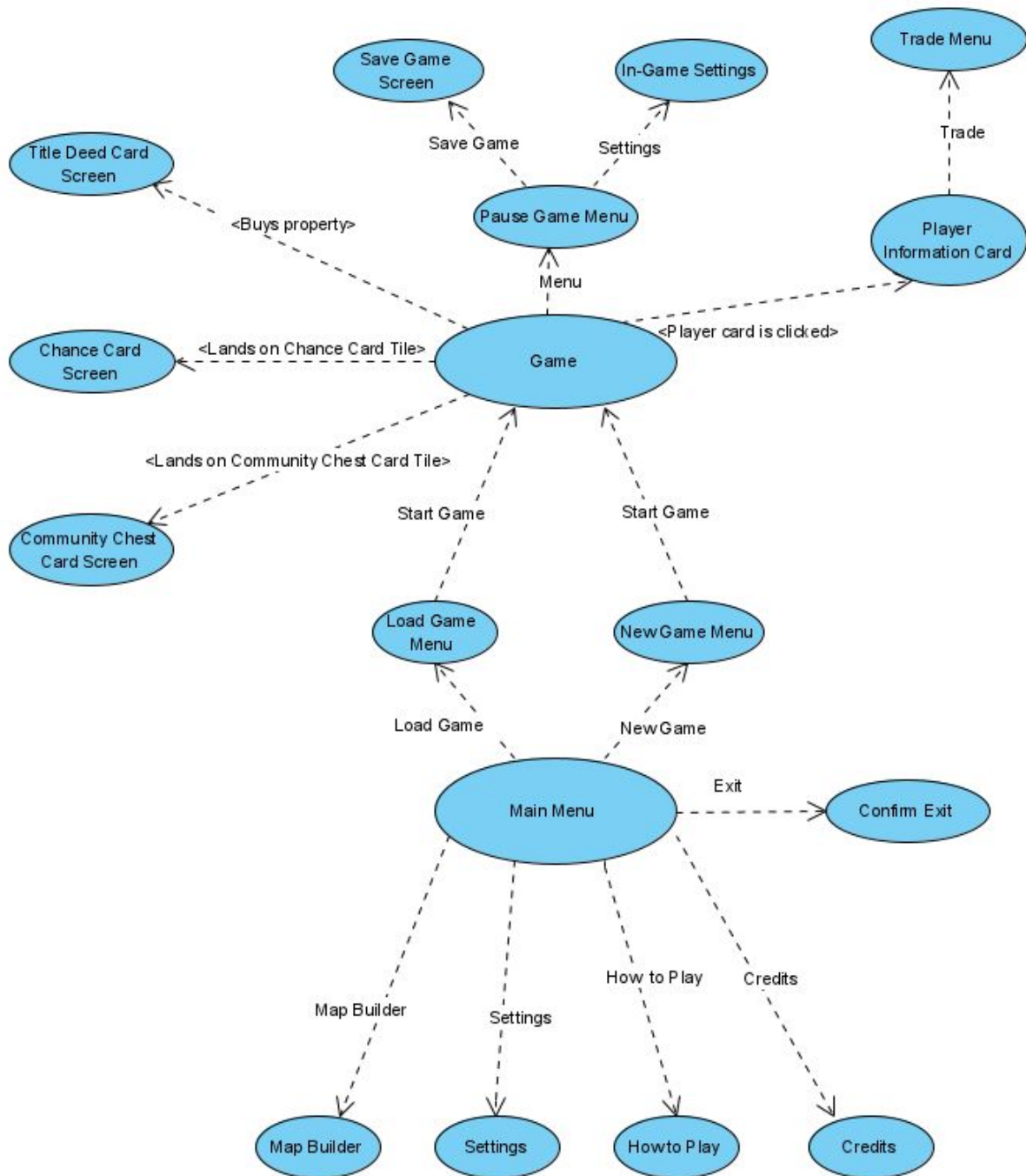
5.4.2. Screen Mockups
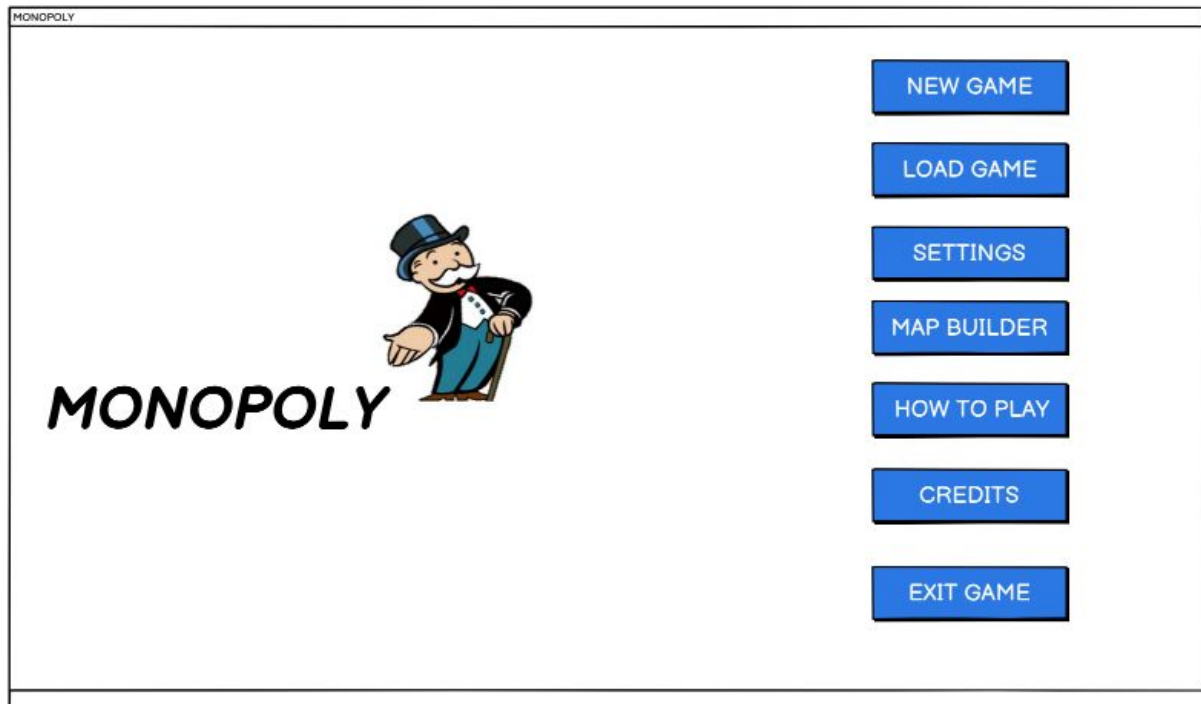
Figure 1. Main Menu

The main menu is the starting screen of the game. It contains 7 buttons. All buttons open their respective screens.

Figure 2. New Game Menu

"New Game" screen is used to construct a new game. The player can choose a previously built map to play on, the game pace which indicates how quickly the game will end, the number of players, and bot character. After specifying the features of the game, the player starts the new game using the "Start Game" button.
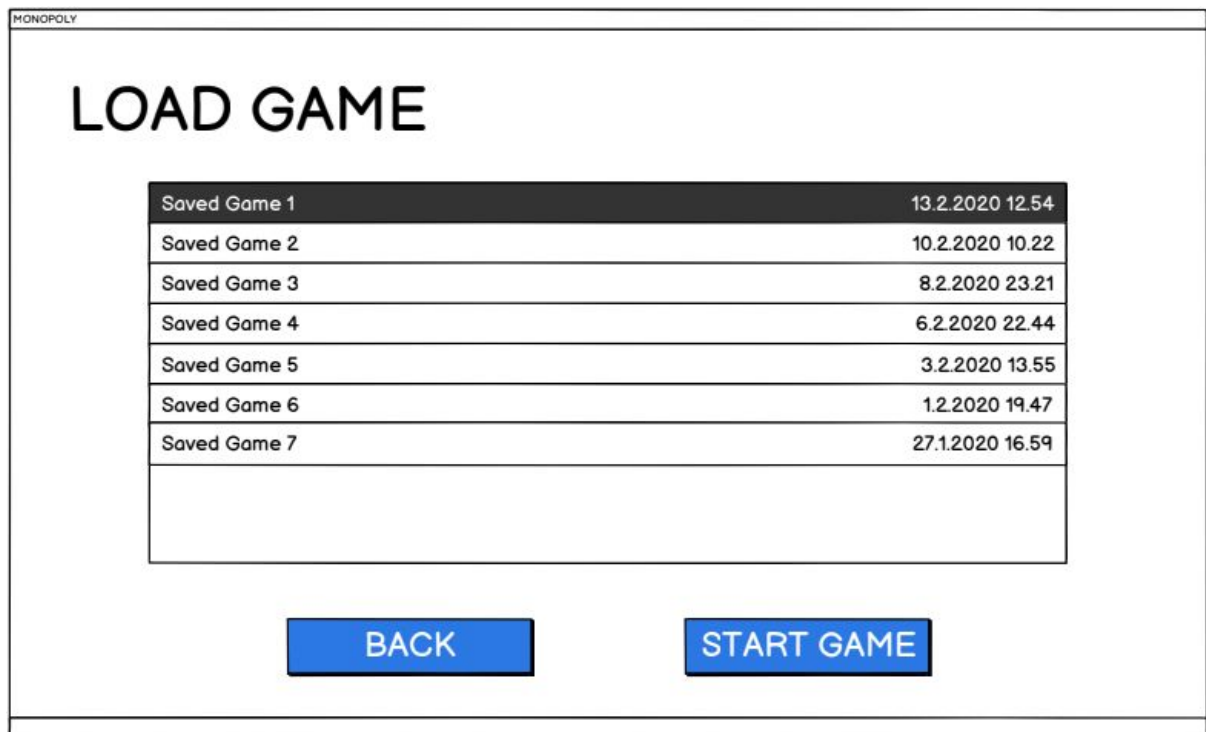
Figure 3. Load Game Menu

In the "Load Game" screen, the player chooses a previously saved game to continue playing.
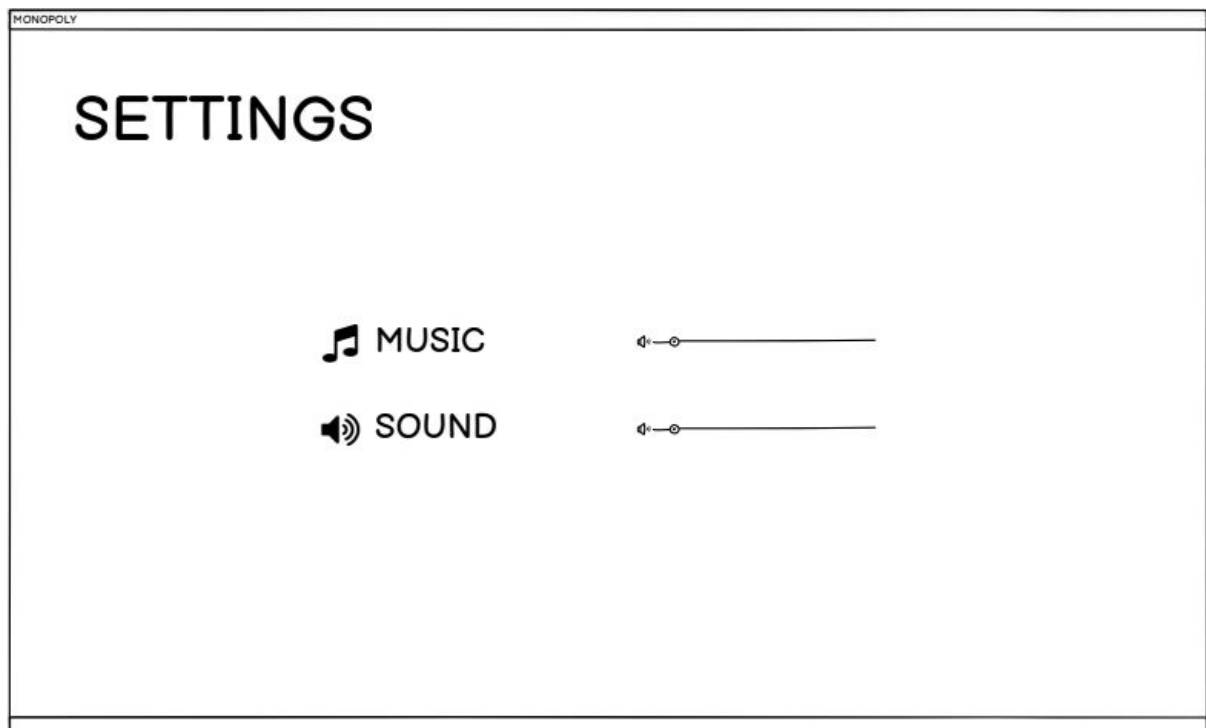
Figure 4. Settings

In the "Settings" screen, the volume of music and sound effects can be changed.
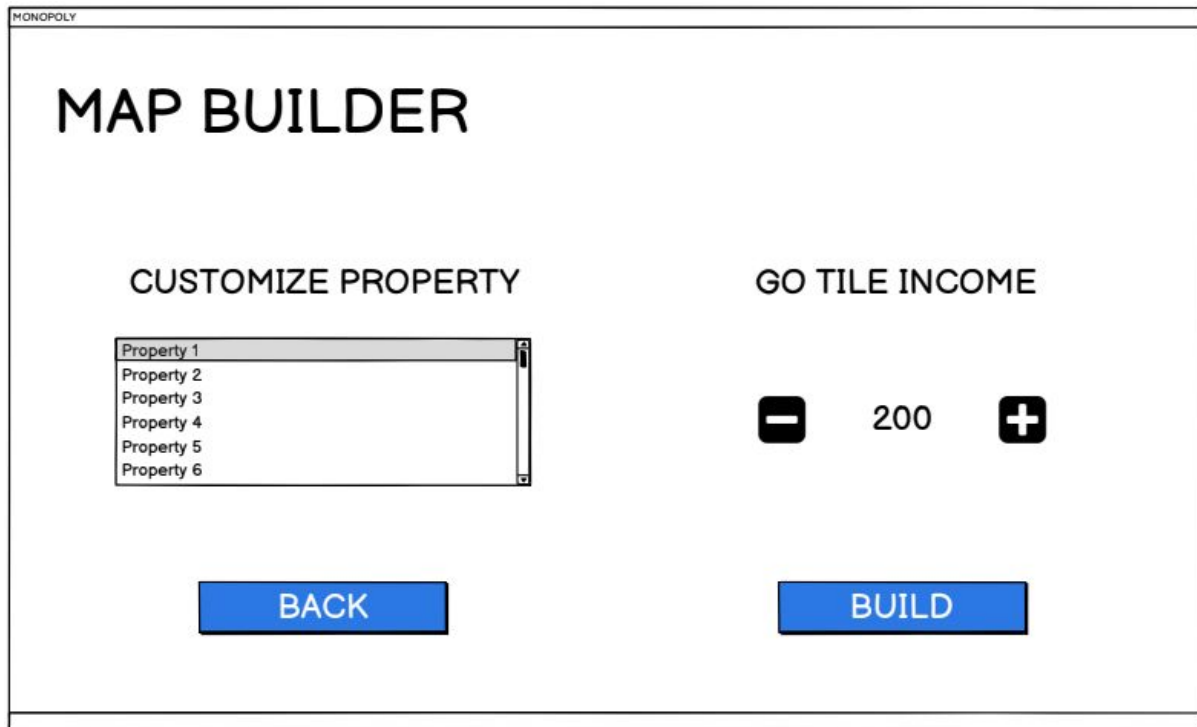
Figure 5. Map Builder

In the "Map Builder" screen, the user can create a map with custom properties and go tile income. When any property in "Customize Property" section is clicked, the following pop-up screen appears:
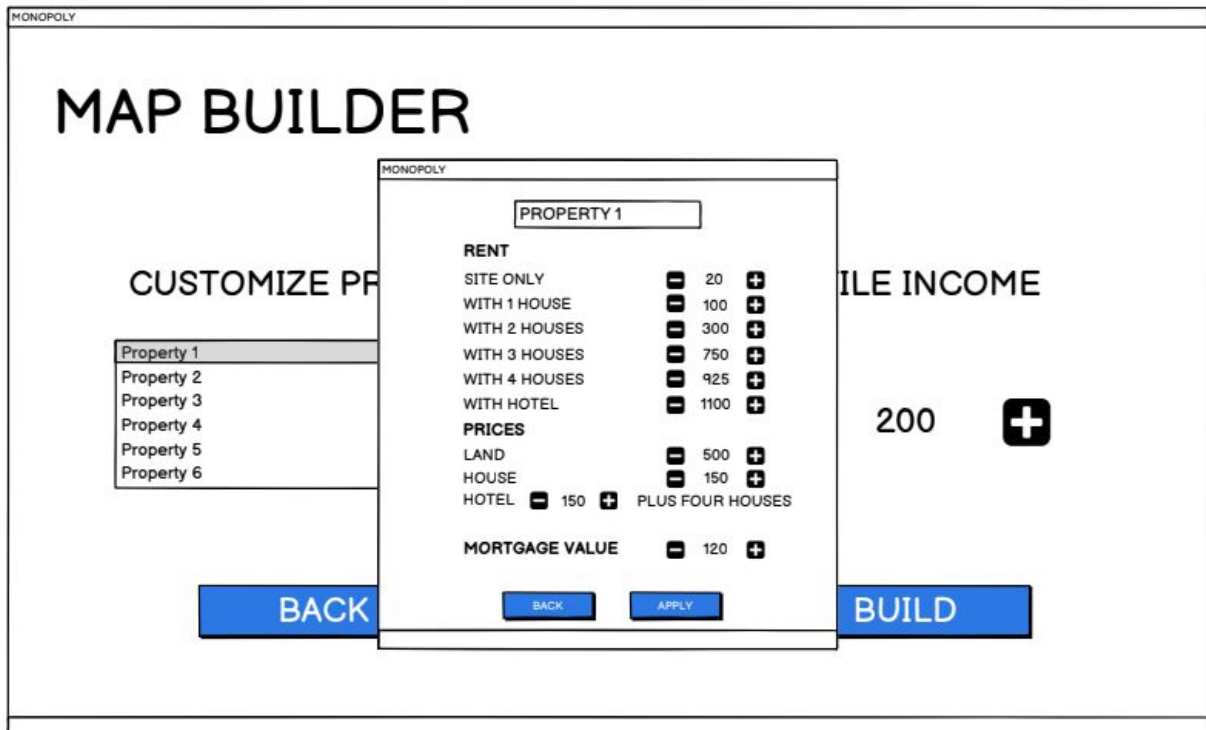
Figure 6. Map Builder Customization Popup

Using the interface shown in Figure 6, the user can change the name, rent, prices, and mortgage value of the selected property.

Figure 7. How To Play

In the "How to Play" screen, the user can get information about the rules and conventions of the classic Monopoly game, as well as the extra features that we have implemented.
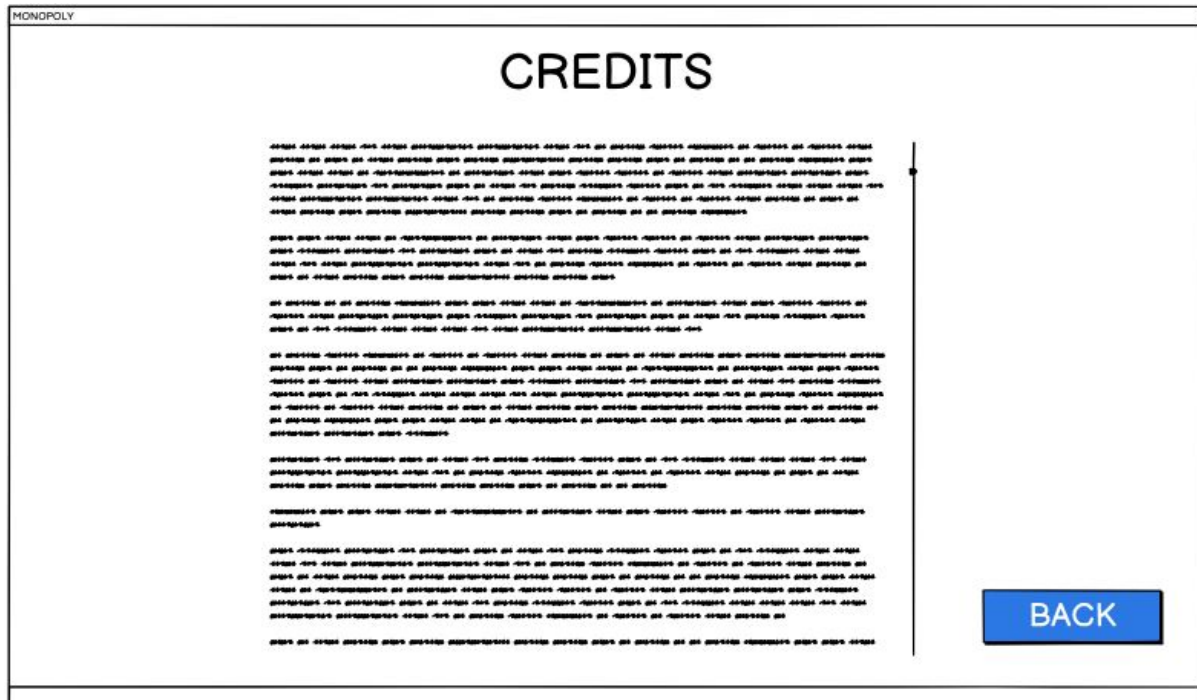
Figure 8. Credits

In the "Credits" screen, the user can view the credits.

When the user clicks "Exit Game" button, a confirmation is asked from the user on the following pop-up screen:

Figure 9. Confirm Exit

The following mockups describe in-game screens.

Figure 10. Game

This screen is displayed while the game is being played. It illustrates information regarding all players playing the game. The tokens of the players are also displayed on the tile they are on. The symbols on the game board indicate the following:

Plane and Train Icons: Indicate the teleport tiles.

Money Icon: Indicates community chest card tiles and their deck.

Question Mark Icon: Indicates chance card tiles and their deck.

Black House Icon: Indicates title deed card deck.

Arrow Icon: Indicates GO tile.

In-Jail Icon: Indicates the jail tile.

Car Icon: Indicates free parking tile.

Police Icon: Indicates go to jail tile.

The first four icons are taken from Balsamiq's preinstalled icons, and the last three icons are taken from the original Monopoly game board.

The screen contains four buttons. By clicking the "Buy Property" button, a player can buy the property on their tile, or they can start an auction. This is done on the following interface:
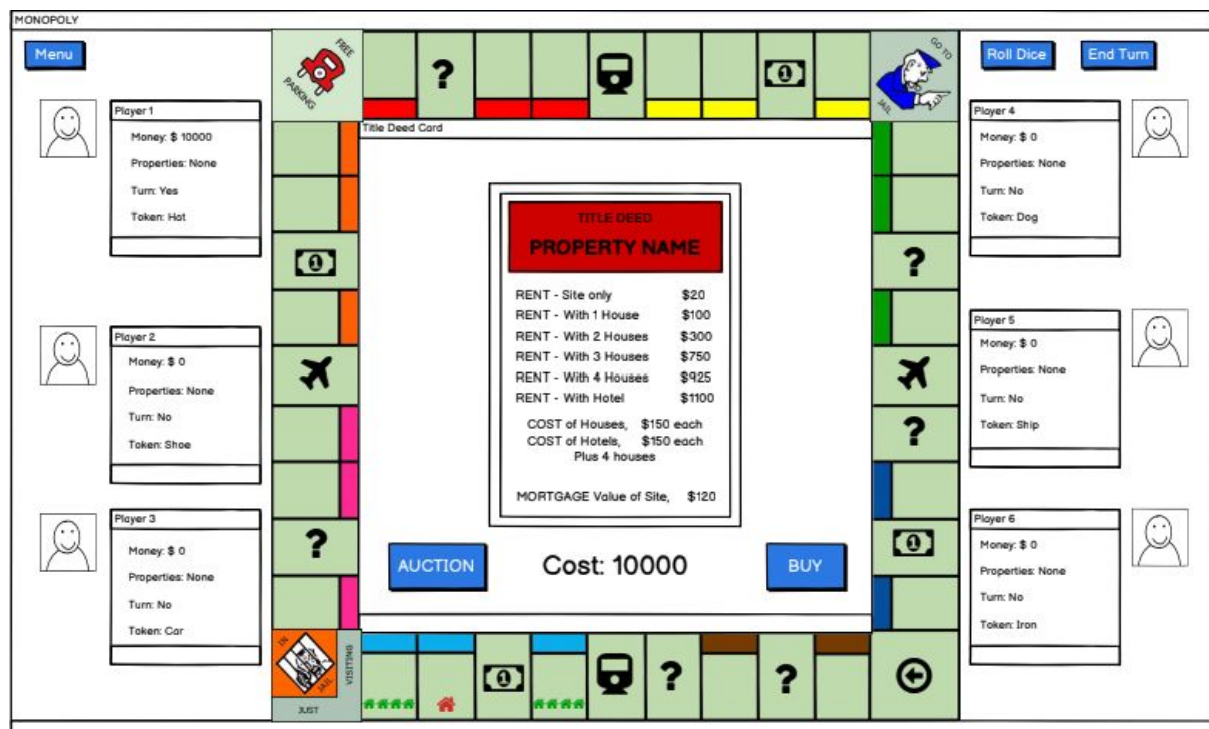
Figure 11. Title Deed Card Screen

When a player refuses to buy a property, that property is put up for auction on the following screen:
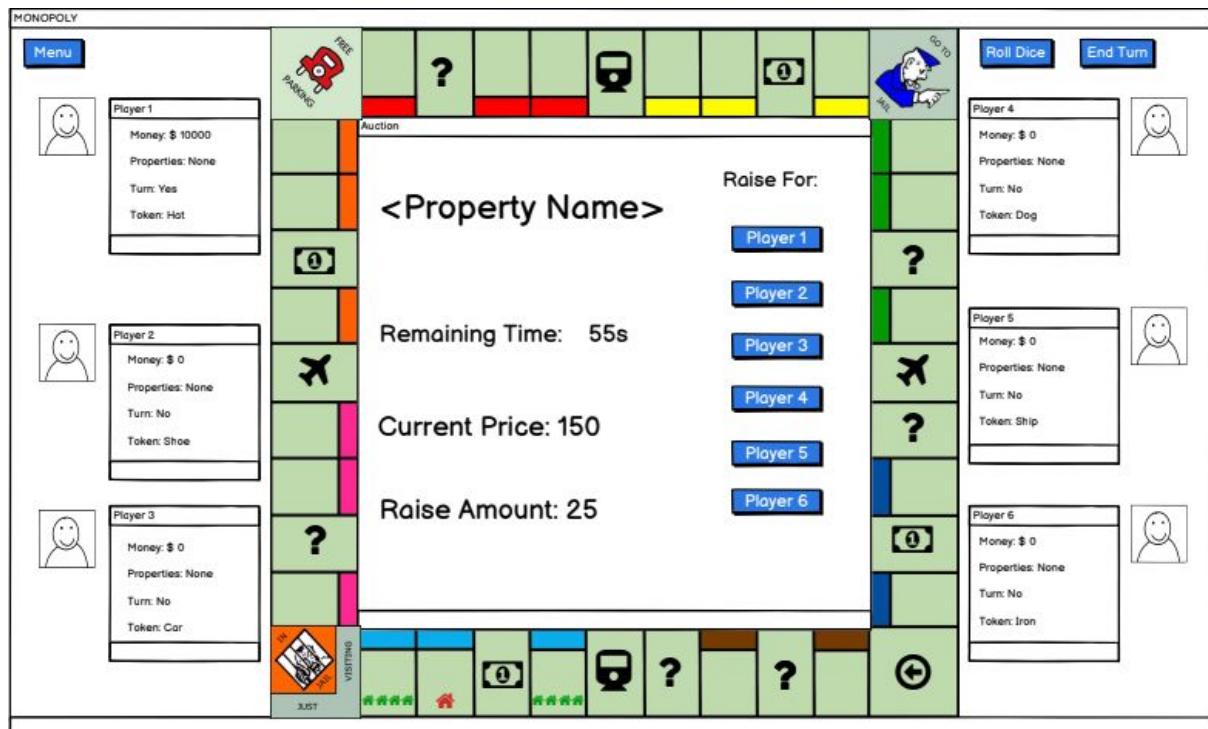
Figure 12. Auction Screen

By clicking the "Roll Dice" button, the player rolls dice and moves their tokens according to the result of the dice.

By clicking the "End Turn" button, the player ends the turn. Then, the game continues with the following player's turn.

Several additional screens are displayed based on the location of the currently playing player. When a player lands on a chance card or community chest card tile, the following pop-ups are displayed:
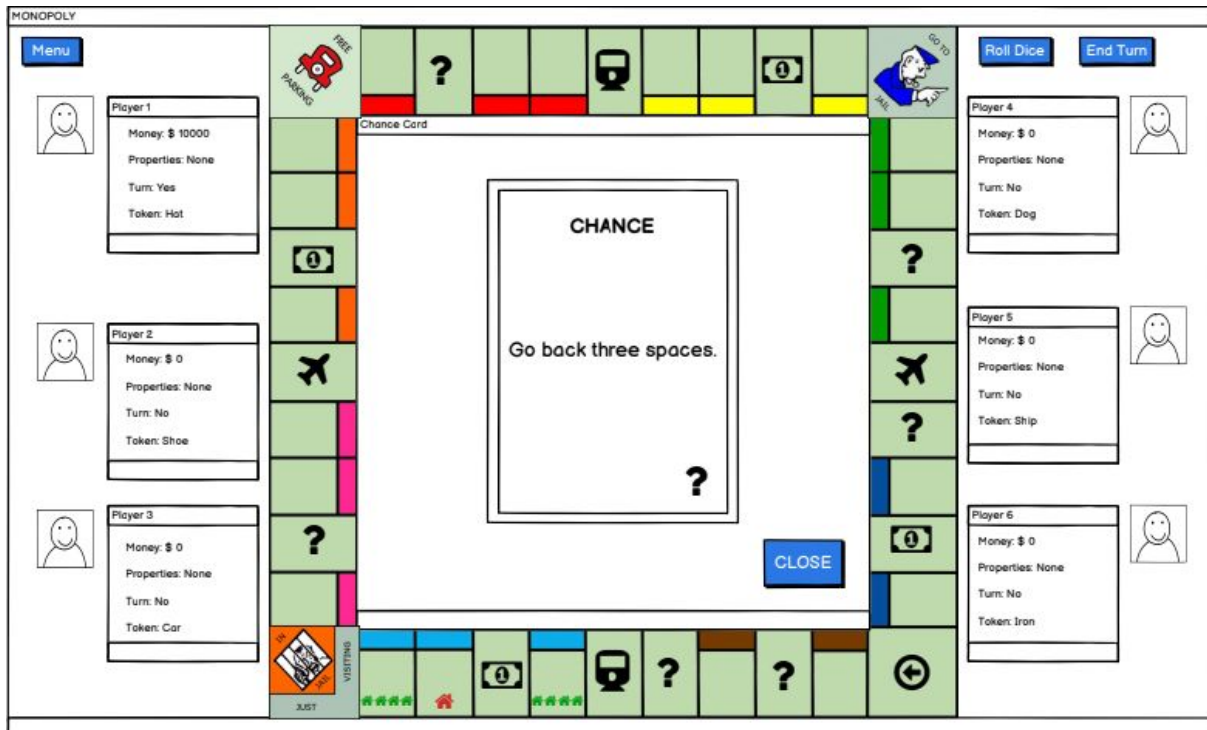
Figure 13. Chance Card Screen
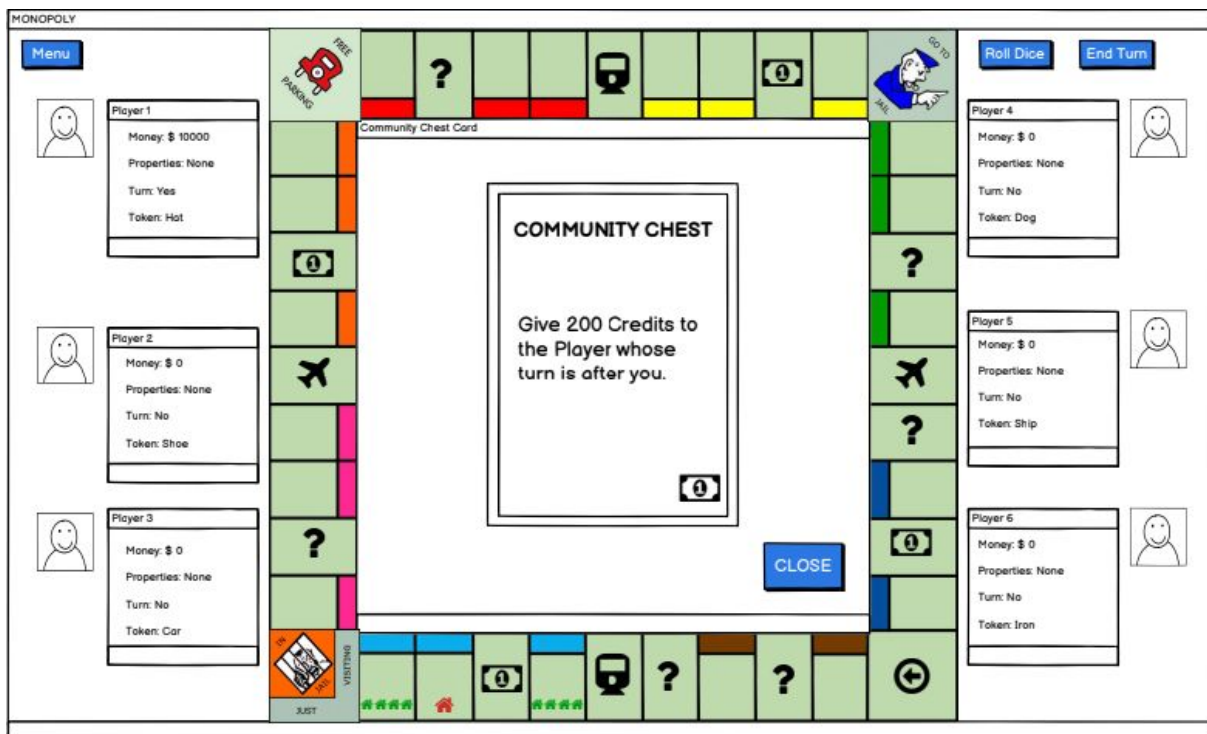
Figure 14. Community Chest Card Screen

When "Menu" button is clicked during the game, the following menu screen appears:
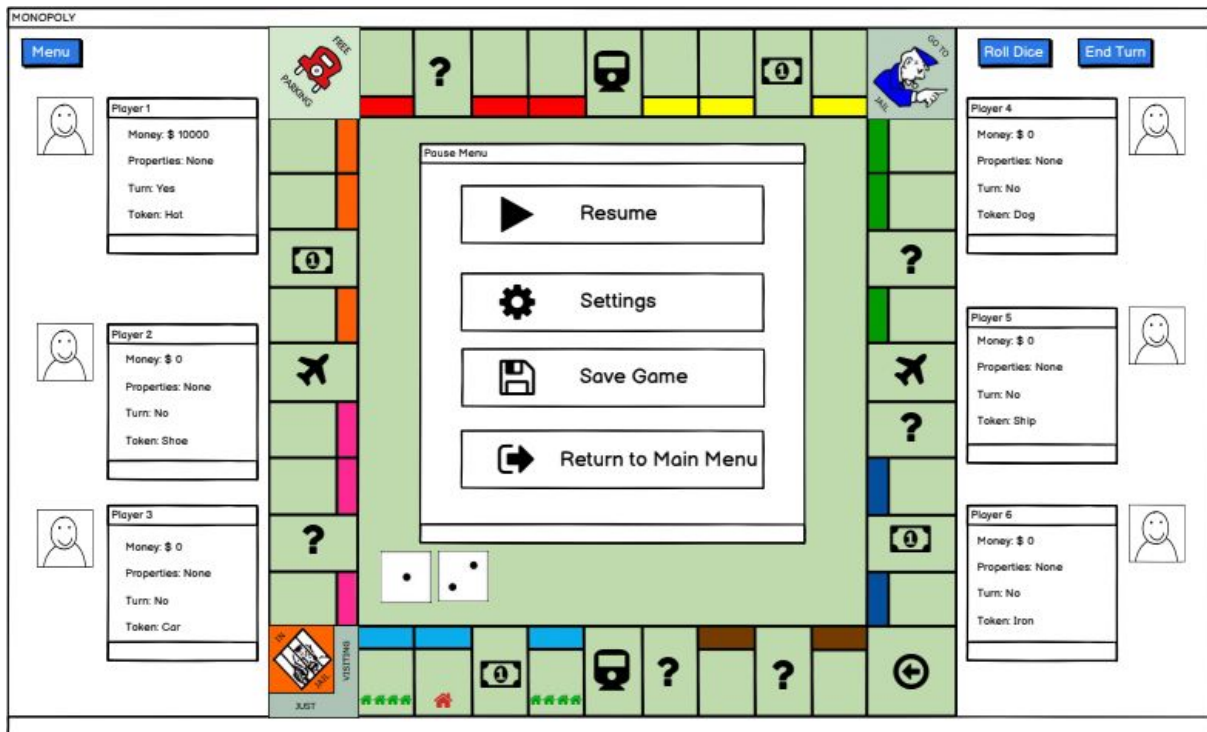
Figure 15. Pause Game Screen
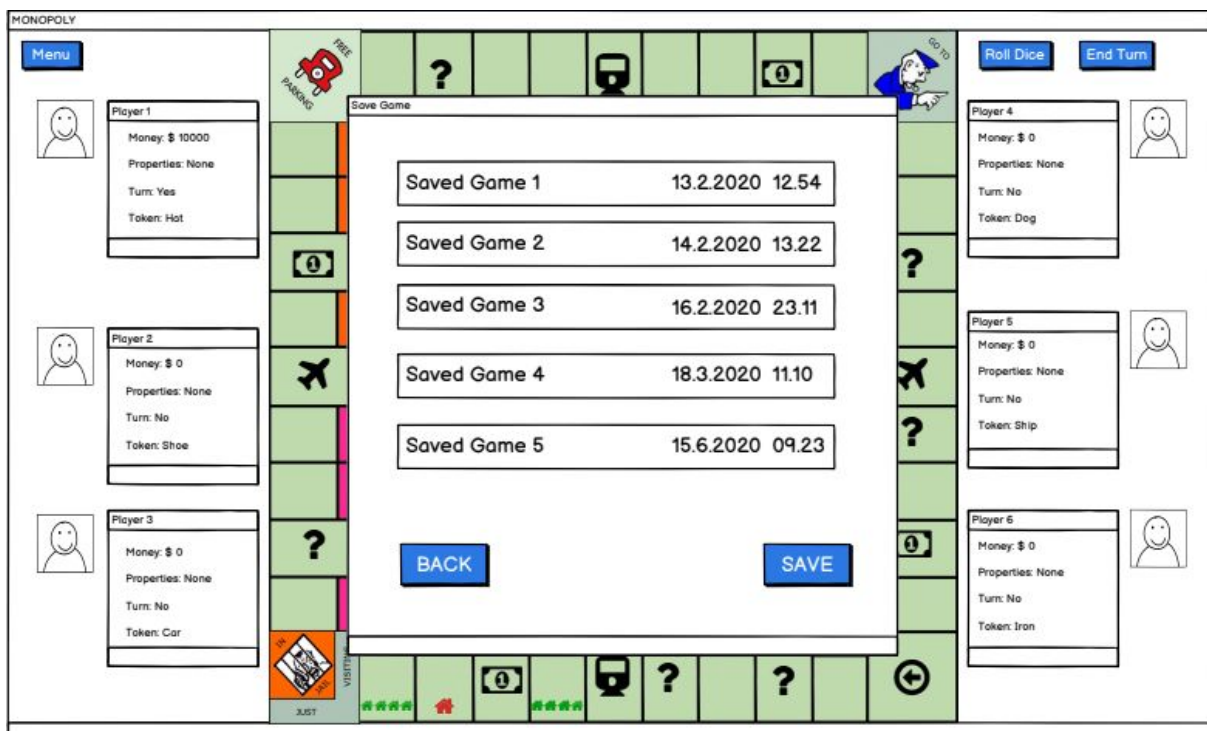
Figure 16. Save Game Screen

The Save Game screen is given as in Figure 16. The player may choose to override or save the game to an empty slot. In this example all 5 save slots are full, so the player has to override one of the save slots.
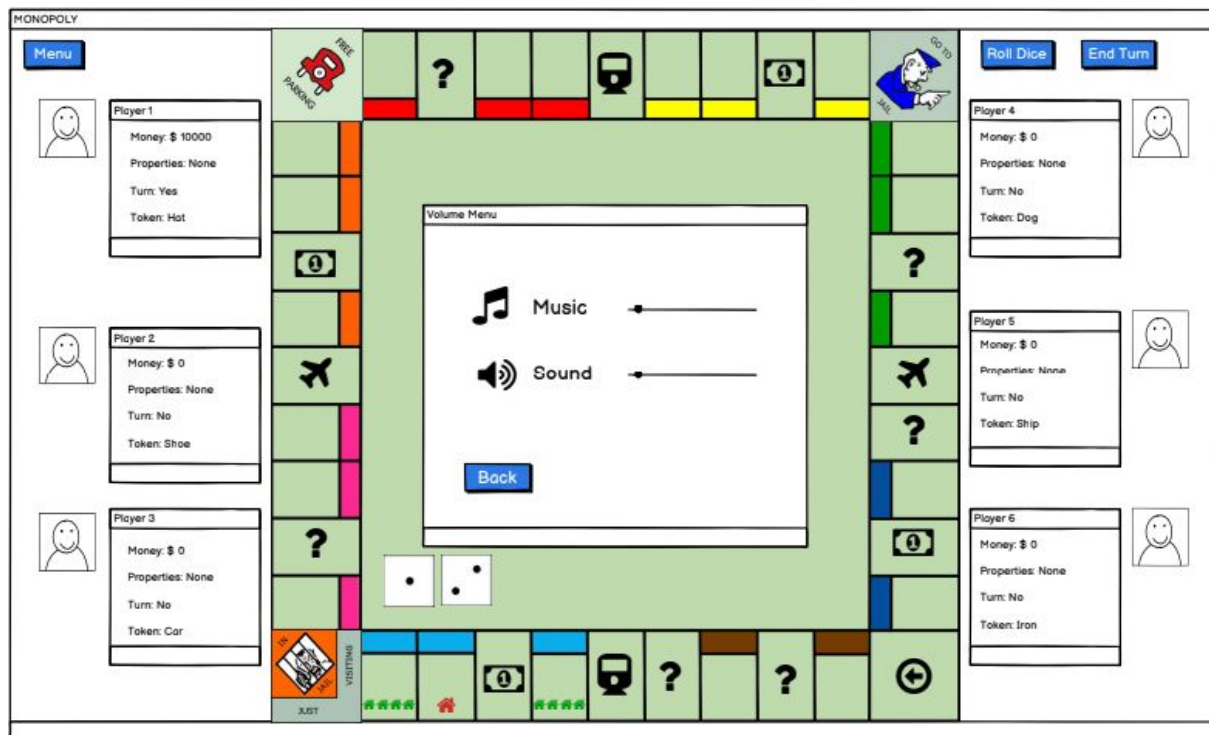
Figure 17. In-Game Settings

When a currently playing player clicks on their own icon, the following information card appears:
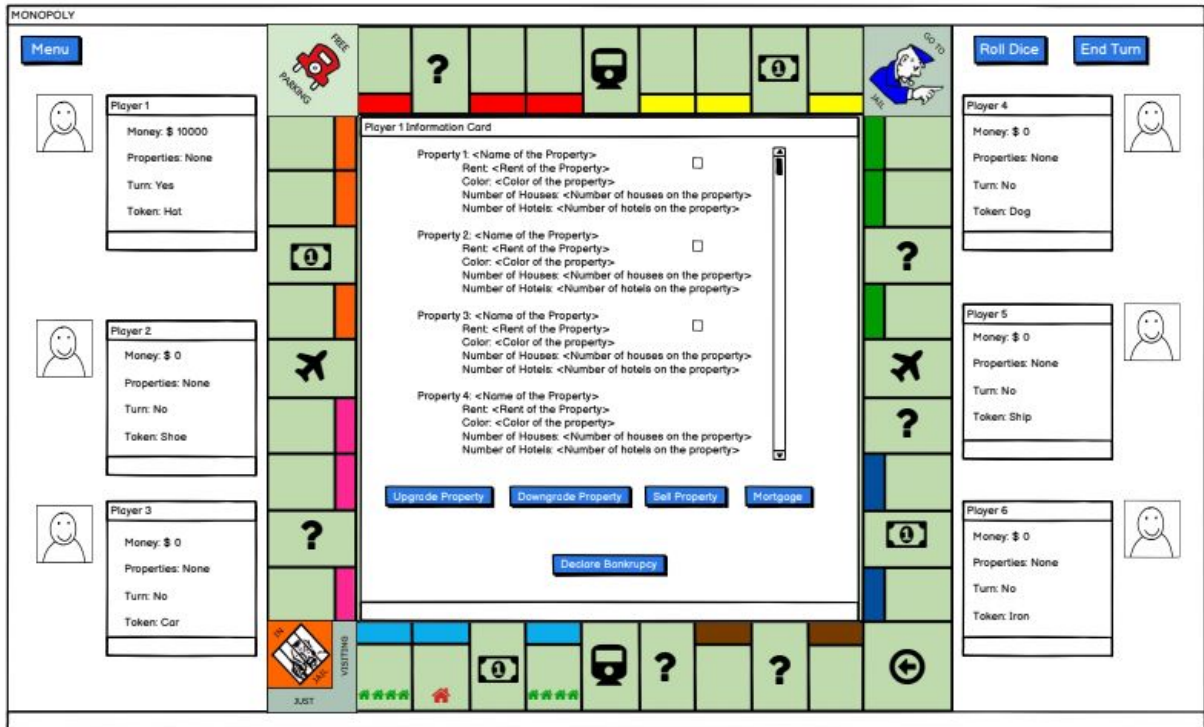
Figure 18. Player Information Card (As seen from player's POV)

When a currently playing player clicks on another player's icon, the following information card appears:
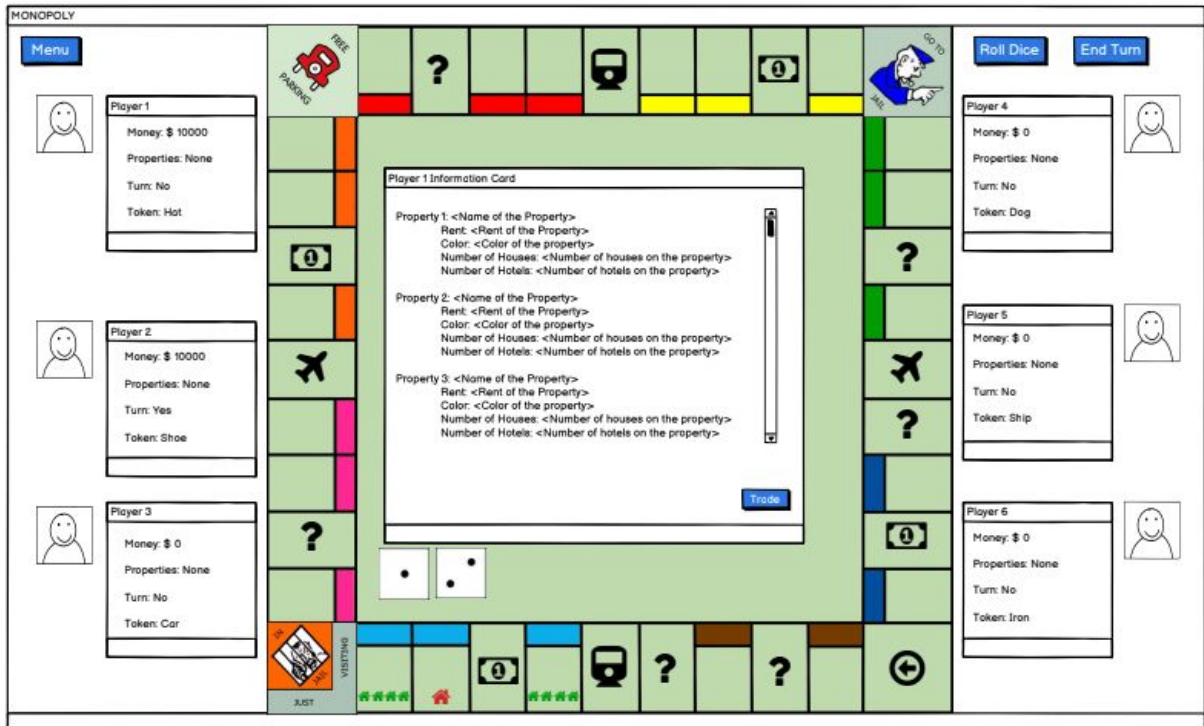
Figure 19. Player Information Card (As seen from another player's POV)

A currently playing player can perform trade with an arbitrary player on the following screen:

Figure 20. Trade Menu

Player 1 proposes a trade offer to Player 2 by selecting his offerings and the items wanted from Player 2's inventory. If Player 1 selects "Propose", an offer is sent. If "Back" is selected, the game screen is opened again.
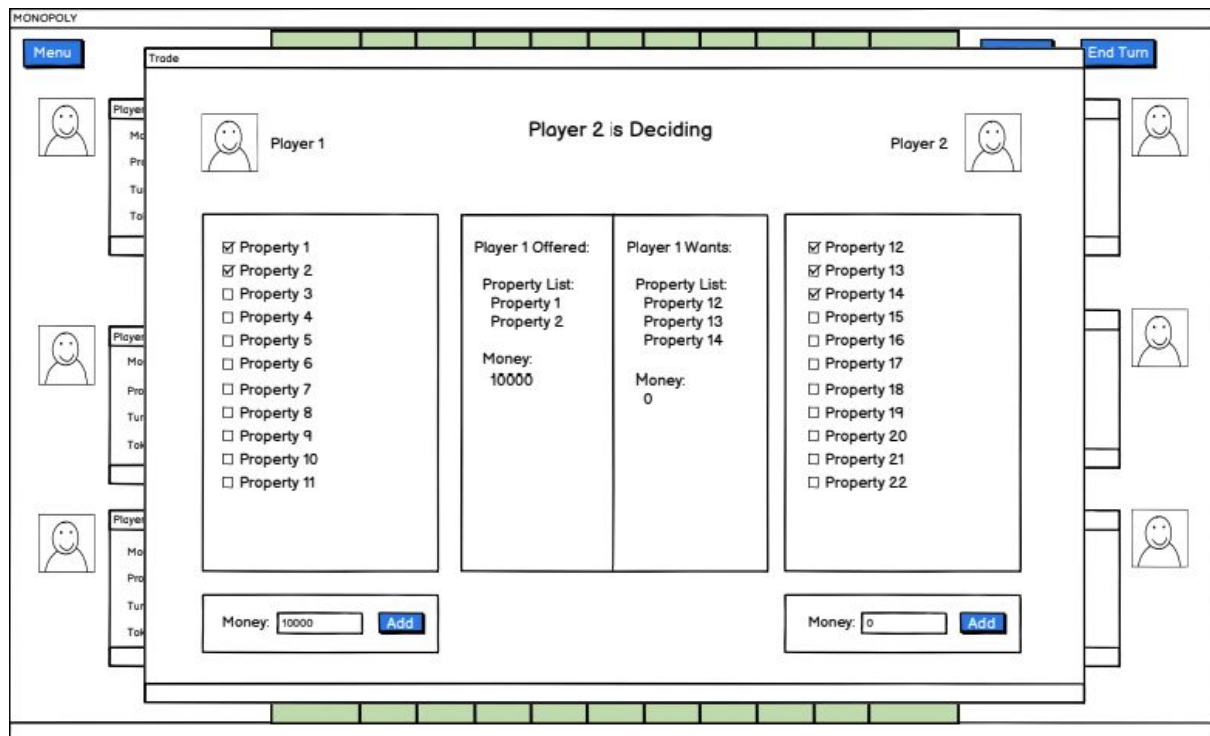
Figure 21. Trade Menu

If Player 2 is an AI player, the screen will not include "Accept" and Reject "options". The AI player will respond to the trade in correspondence to its character. After the trade is complete, the game screen is opened again.
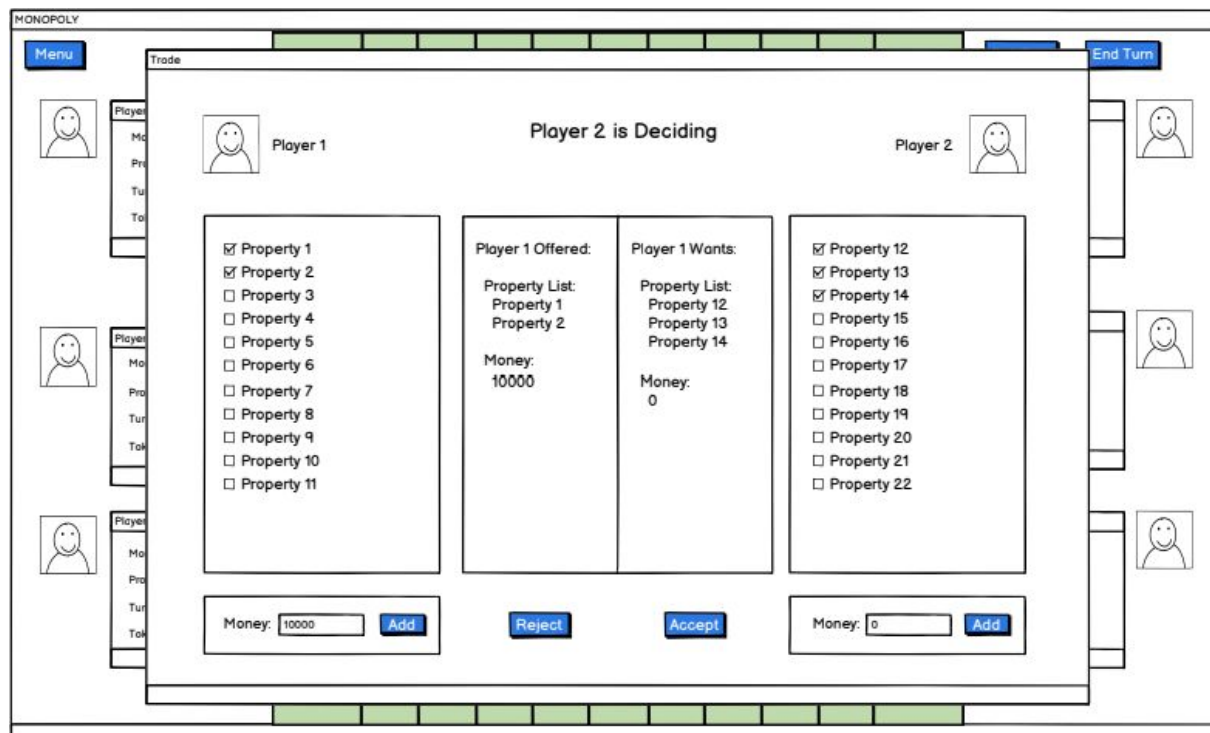

Figure 22. Trade Menu

If Player 2 is a human player, "Accept" and "Reject" options will be available. The offer values set by Player 1 cannot be changed by Player 2. Accept and Reject options will be pressed by Player 2 according to his or her preferences. After the trade is complete, the game screen will be opened.

5.4.3. Game Tokens

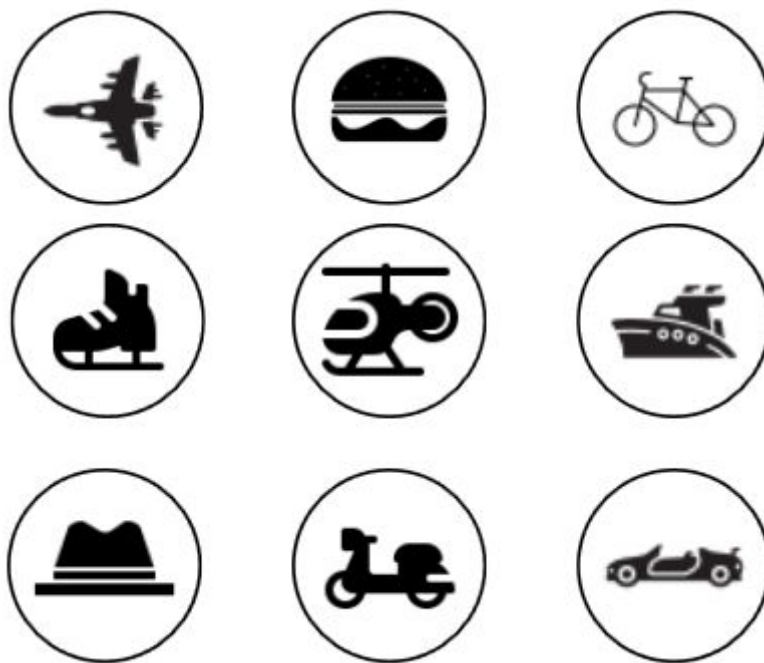Some sample tokens that will be used in game are shown in Figure 23.

Figure 23. *Game Tokens*

6. Conclusion

In this requirement elicitation and analysis report, we have taken the classic board game Monopoly, added extra features to it and created a design for a modernized digital version of the classic board game.

As extra features to the classic board game, we added teleportation tiles, a customization tool for the board, selections for changing the behaviour of the AI players as well as the pace of the game. Furthermore, we designed complex systems to carry the classic Monopoly mechanics to the digital system so that players can have smooth trade transactions with each other, or auctions.

All of these features will be implemented in accordance with the constraints and requirements stated earlier, using the designs as shown on the various diagrams.

7. Glossary & References

7.1. Glossary

AI Characteristics: AI characteristics are a special feature in this digital version of Monopoly. Players can customize the behaviour and decision-making of the AI by selecting a characteristic for the AI. Further discussion about customizing the AI is under section 2.14.

Auction: An auction is a system of selling property in Monopoly. The money paid by the player goes to the bank.

Bank: Bank is the main entity in the Monopoly game controlling financial actions. It has unlimited money and handles the first-time selling of the properties, as well as auctions. Other aspects of the bank is further discussed under section 2.5.

Bankruptcy: When a player owes to the bank or a player more than he can pay, he is forced to resign by declaring bankruptcy. Bankruptcy is handled according to the terms discussed under section 2.15.

Board Salary: the amount of money which the player get when they pass over or land on "GO Tile" defined by the board itself

Chance Cards: Player draws chance cards when he lands on a Chance Card Tile. After that, the player must perform the actions stated in the chance card. Further discussion of chance cards is under section 2.6.

Color Group: Every property is in a color group. A color group consists of 2 to 3 properties. A player must collect all properties in a color group to upgrade a property.

Community Chest Cards: Player draws chance cards when he lands on a Community Chest Card Tile. Players must then do what is asked on the community chest card, which is usually paying a certain sum to the bank, depending on the particular card. Further discussion of community chest cards is under section 2.6.

Dice: Dice is one of the main elements in the monopoly game. The game is played by rolling a pair of 6 sided dice, and moving according to the sum every turn.

Double Dice: Throwing a double dice is having the same side come up in both of the dice thrown. When a player throws a double dice, the player who threw the double dice plays another turn. Throwing 3 double dice in a row results in the player rolling the dice sent to jail.

Downgrade Property: Selling an upgrade on a particular property. For example, downgrading the property from 4 houses to 3 houses.

GO Tile: Every player starts the game on this tile. Everytime a player passes through this tile, they receive their wage.

Hotel: Hotel is the final upgrade for a property, and is unlocked after building 4 houses to every property in a color group.

House: House is an upgrade for a property that increases its rent significantly. Building of houses must follow the rules explained in 2.10.

Jail: If any of the conditions stated in 2.12, such as landing on the go to jail tile, happen, the player is moved to the jail tile, where he cannot leave until he bails out of jail.

Mortgage: When deemed necessary, a player might choose to mortgage a particular property. When mortgaging, the player receives the sum stated on the title deed card and other players who land on the mortgaged property of the other players do not pay rent to the owner. To lift the mortgage, the player has to pay the price of the mortgage plus the interest to the bank. Mortgaging is discussed further under section 2.11.

Player Token: Player token is the representation of the player on the board, i. e. when the player throws the dice, his player token is moved on the board.

Rent: When a player lands on another player's property, he must pay the rent according to the upgrade level of the property.

Total Worth: Sum of the values of every property, every upgrade and the balance of a player is that player's total worth.

Trade Offer: Players can exchange properties, money or other tradables by making a trade offer to another player during their turn. Trade offers are further discussed under section 2.8.

Upgrade Level: It's the upgrade level of a particular property. Once purchased, a property has an upgrade level of 0. Building one house is upgrade level 1, 2 houses are level 2, 3 houses are level 3, 4 houses are level 4 and a hotel is upgrade level 5.

Upgrading A Property: A player can upgrade a property by building a house or a hotel on it. Upgraded properties yield better rent for their owner.

7.2. References

Monopoly, Parker Brothers Real Estate Trading Game [PDF]. (n.d.). Parker Brothers.

E. Flynn, *Monopoly Board*. 2015

Balsamiq. (2020), Balsamiq Studios LLC. [Application]. Available: https://balsamiq.com/