



Bilkent University

Department of Computer Engineering

---

# CS 319 Course Project

*Group: 2C*

## Final 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

<b>1. Implementation</b>	<b>3</b>
1.1 Main Menu	5
1.2 Options Menu	6
1.3 Create or Select Room	7
1.4 Create Room Menu	8
1.5 Join Room Menu	9
1.6 Game Lobby	10
1.7 Owned Properties	11
1.8 Paying Rent	12
1.9 Buy Property Confirmation	13
1.10 Auction Bid	14
1.11 Regular Turn	15
1.12 Jail Condition	16
1.13 Scoreboard	17
<b>2. User Manual</b>	<b>20</b>
2.1 General Information	20
2.2 Build Instructions	20
2.4 How to Play	21
<b>2. Conclusion</b>	<b>22</b>

# 1. Implementation

After the first iteration, we decided on the first version of the general hierarchy of the game and created model and manager classes accordingly. Since we wanted to implement a multiplayer version of the game which requires a server, we used JavaScript for the programming language in order to make things easier on the server-side. In addition to that in order to provide a better-looking and enhanced UI to players, we used React.js and Pixi.js. As a development environment, all members used IntelliJ WebStorm or IntelliJ Idea, depending on each individual's choice. The code is written is pushed onto a GitHub repository, in order to follow the code state and update local versions of the project accordingly. For different parts of the game, we created new branches in order to not to intervene in the implementation process and we merged them into the master branch when it is completed.

During the implementation, each member was distributed in some different parts of the project.

- Murat was in charge of implementing the server-side and the communication between the server and the game, as well as some parts of view such as the board, tiles, etc. He also implemented some of the manager classes.

- Ümit was responsible for implementing the game UI which uses React.js. He also took responsibilities in the backend and server part of the game.
- Emre has written the UI of the game and views for properties, their cards, board, and etc. with Pixi.js, and he also contributed to game logic.
- Furkan implemented the model part of the game, and also took care of implementing some parts of the game logic such as buying a city, erecting houses, and hotels.
- Can also be responsible for implementing models and contributed to the backend part of the game. He also helped Ümit in merging chat functionality into the React screen.

It was to some extent easier for us to start implementing and move on since we had been writing reports from the beginning of the semester. The design report helped us considerably in creating classes and provide associations between those classes in order to make the whole game a working piece. For the meetings we generally used Zoom or Discord, however, some of us were able to meet in person because they were staying at the dormitories in the campus. At specific times we were not able to focus on the project as much as we wanted since all of us had another project due the same day for different courses.

We were able to implement most of the game, however, there are still some parts to be implemented after the due date. Below are the things that are not fully completed before the due date:

- Credits are not fully functional
- The game music system is not functional
- Quest feature is not implemented

## 1.1 Main Menu



Figure 1: Main Menu

Above image shows the main menu screen. When the app is executed by the user, this screen shows up. Here the user has three options, starting a new game by clicking on “START GAME” button, navigating into options menu by clicking on “OPTIONS” button, exit the game clicking the “EXIT” button.

## 1.2 Options Menu

← ROOM OPTION PAGE

---

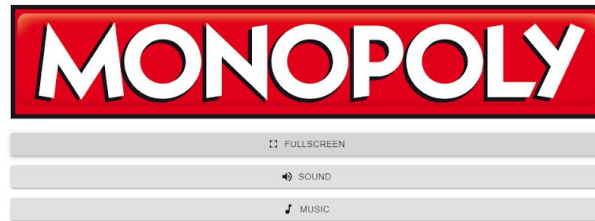


Figure 2: Options Menu

By clicking on the “OPTIONS” button on the main menu of the game, the user is redirected into this menu where s/he makes the game fullscreen by clicking on “FULLSCREEN” button, or adjusts the sound and music by clicking on the appropriate buttons. S/he can also go back to the main menu by clicking on the “ROOM OPTION PAGE” button on the upper left side of the screen.

## 1.3 Create or Select Room

← MAIN MENU

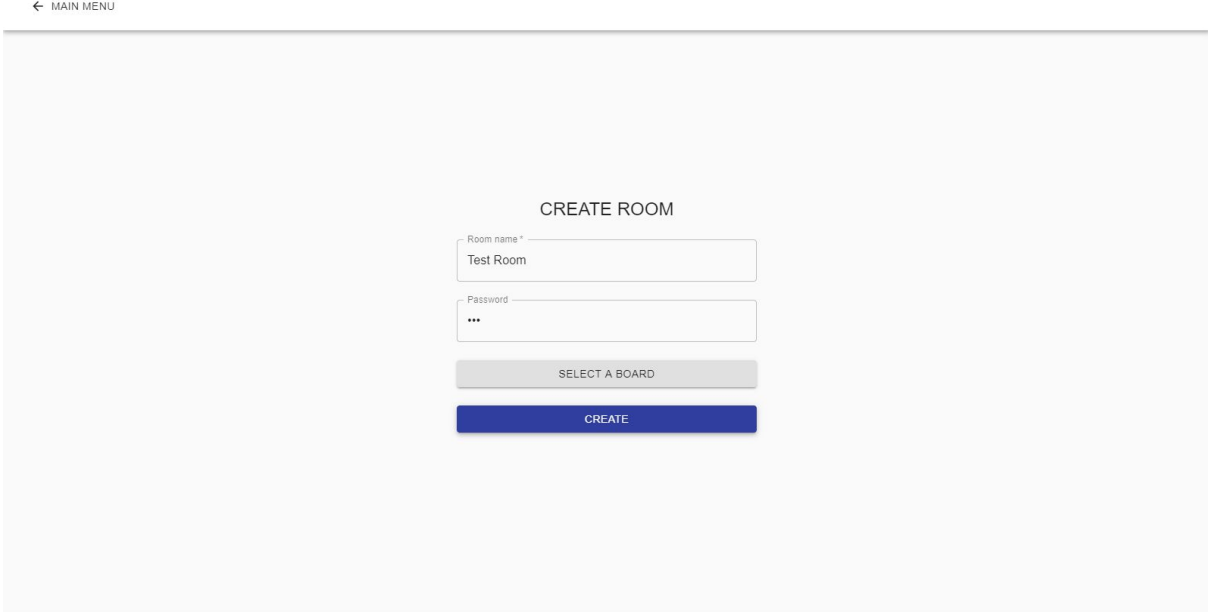
---



Figure 3: Create or Select Room page

This is the screen where users can create a new room or join an existing room in order to play with others.

## 1.4 Create Room Menu



The screenshot shows a web interface for creating a room. At the top left, there is a link labeled "← MAIN MENU". The main content area is titled "CREATE ROOM" and contains the following elements:

- A text input field labeled "Room name \*" with the text "Test Room" entered.
- A password input field labeled "Password" with three asterisks "\*\*\*" displayed.
- A light gray button labeled "SELECT A BOARD".
- A dark blue button labeled "CREATE".

Figure 4: Create Room page

Above is popped up to the user if s/he is to create a new room. Room name is required in order to display the room information and uniquely identify among all the rooms on the server. Password field can be left empty if the lobby owner does not want the lobby to be password protected. "SELECT A BOARD" button allows the player to select predefined board templates. After selecting the template, the user can start the lobby and the user is redirected into the lobby.



## 1.5 Join Room Menu

← ROOM OPTION PAGE

Select A Room			
Room name	Password	Selected Board	Join Button
Test	Required	Template - 1	JOIN
Test Room	Required	Template-2	JOIN
Rows per page: 10 1-2 of 2 < >			

Figure 5: Join Room page

You can select a room in order to play Monopoly via this page. There is a list of currently available rooms in here and you can press the blue JOIN button in the desired row in order to join the room. There will be a popup if there is a password required in the room will be joined. In both situations, whether or not require a password, the system will ask your username for this particular room and game. It will not be stored anywhere for later usages in other games.

## 1.6 Game Lobby

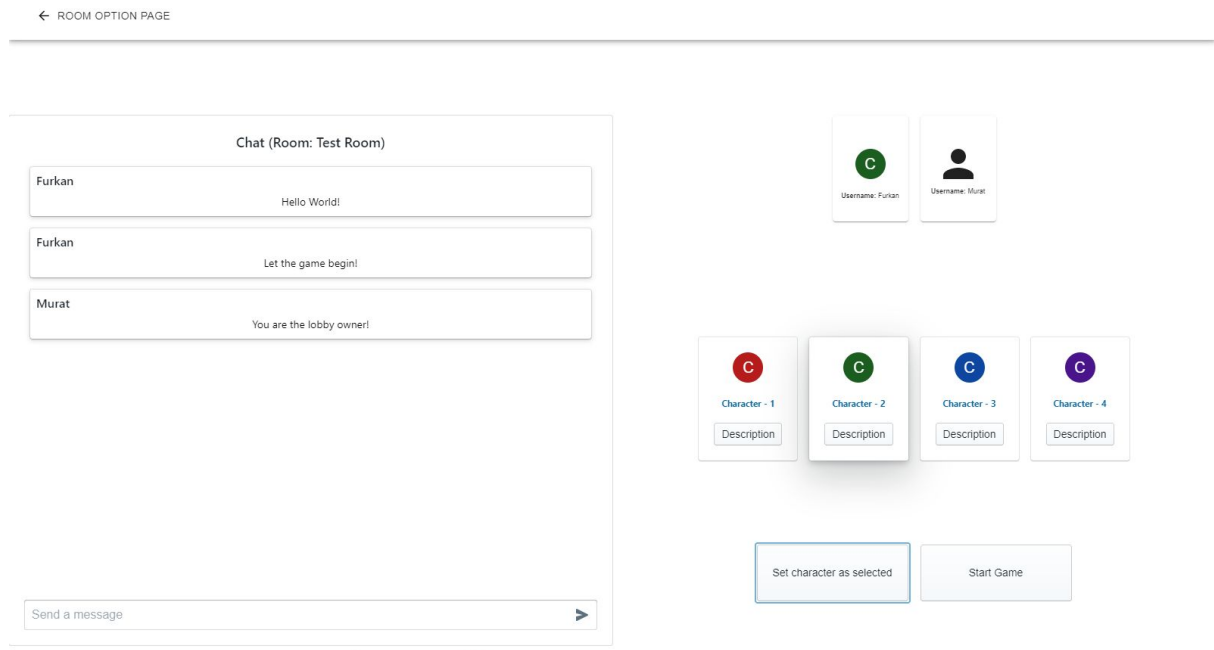


Figure 6: Game Lobby

This is the last screen before the game. In this screen players can see each others character, and set their characters. They also can chat on the chat interface shown on the left side of the screen.

## 1.7 Owned Properties

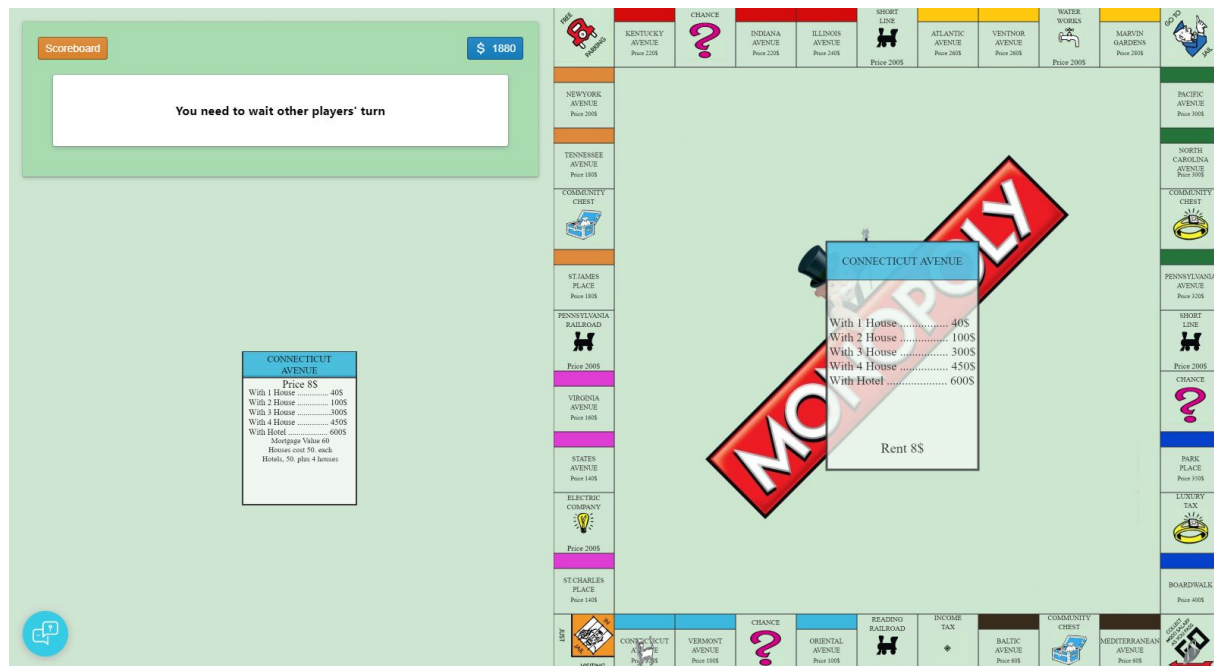


Figure 7: Owned properties

The player's owned properties will be displayed on the left. When a player clicks the card or chooses a tile from the board, the details will be displayed in the middle of the board. This will be seen only by the player who selects the tile.

## 1.8 Paying Rent

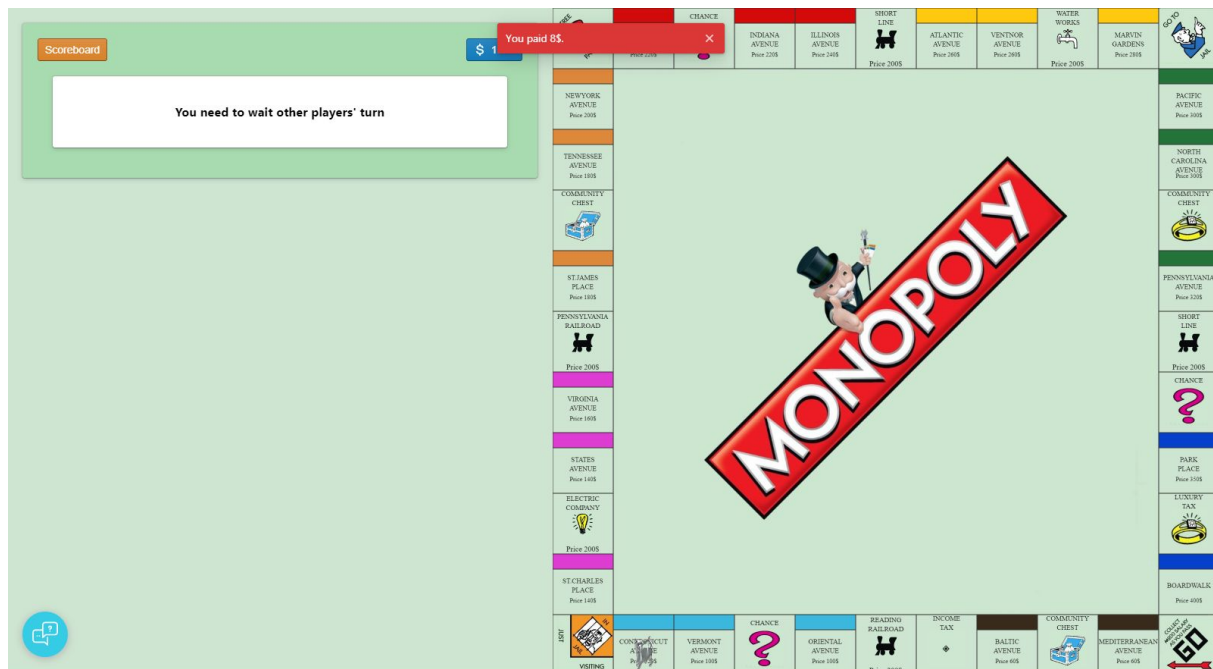


Figure 8: Paying Rent Notification

When a player loses money through paying rent or losing money from Community or Chance cards, the pop-up notification will inform the user.

## 1.9 Buy Property Confirmation

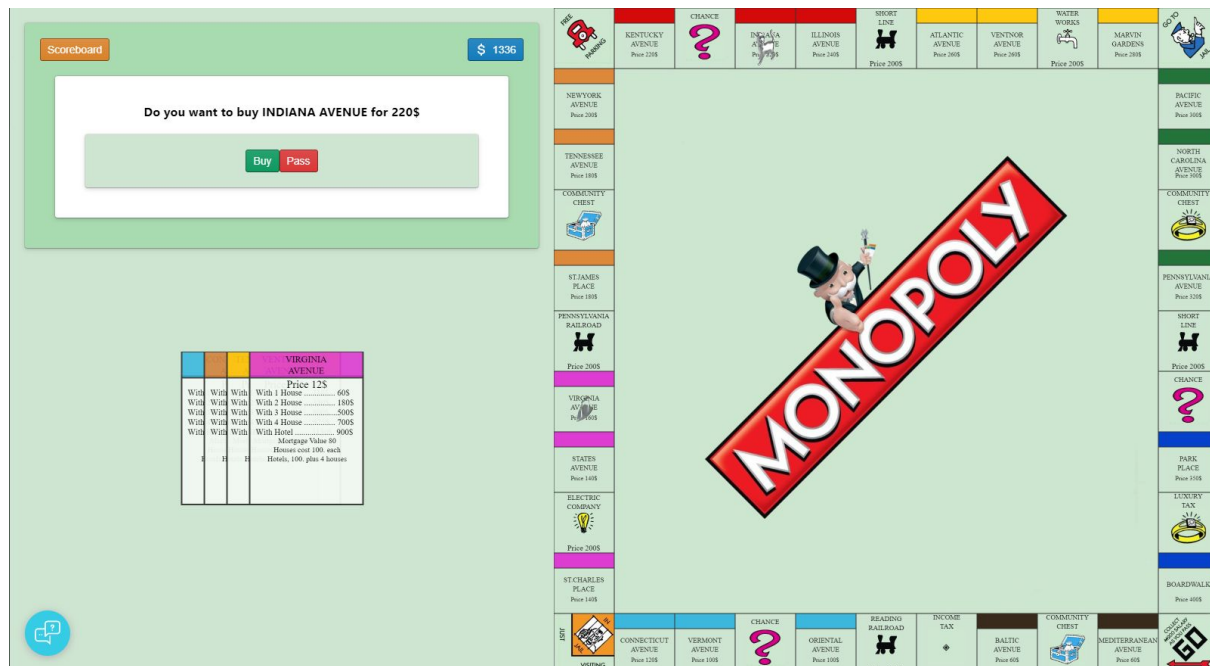


Figure 9: Buy Property

When a player comes to a tile that is not bought, the state screen comes to the buying screen. Current players can buy the tile by clicking the buy button. Otherwise, if the pass is clicked, the auction screen occurs.

## 1.10 Auction Bid

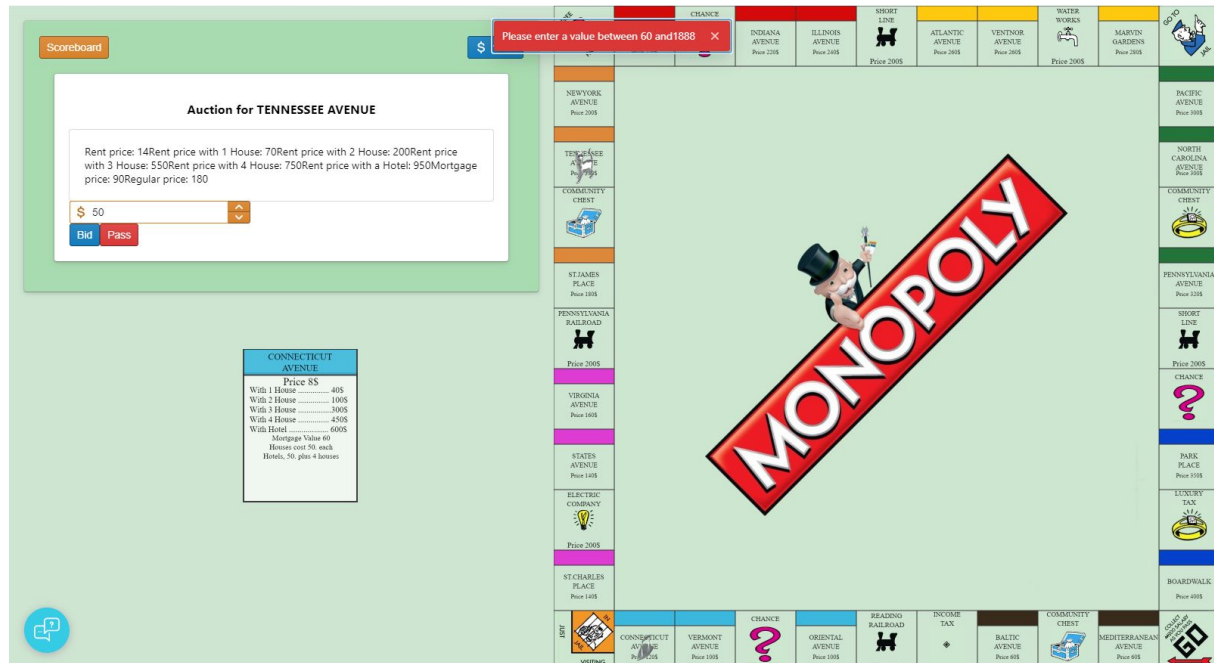


Figure 10: Auction Bid

Each player should enter a minimum bid between the higher bid and the current bid. If they don't the inform notification will pop-up and asks the user to enter a valid bid.

## 1.11 Regular Turn

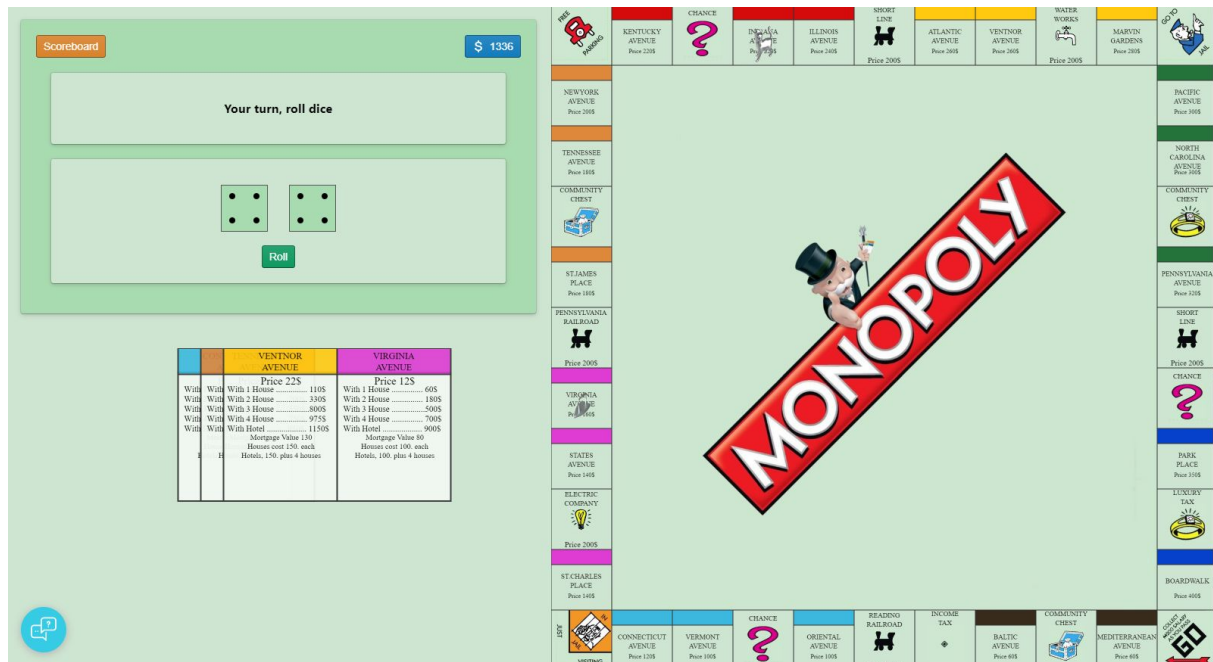


Figure 11: Regular Turn

When a player's turn comes, the screen will be displayed like in the picture. The player can roll a dice and move. When players hover their mouse over their cards, the card is turned up and the full card is shown.

## 1.12 Jail Condition

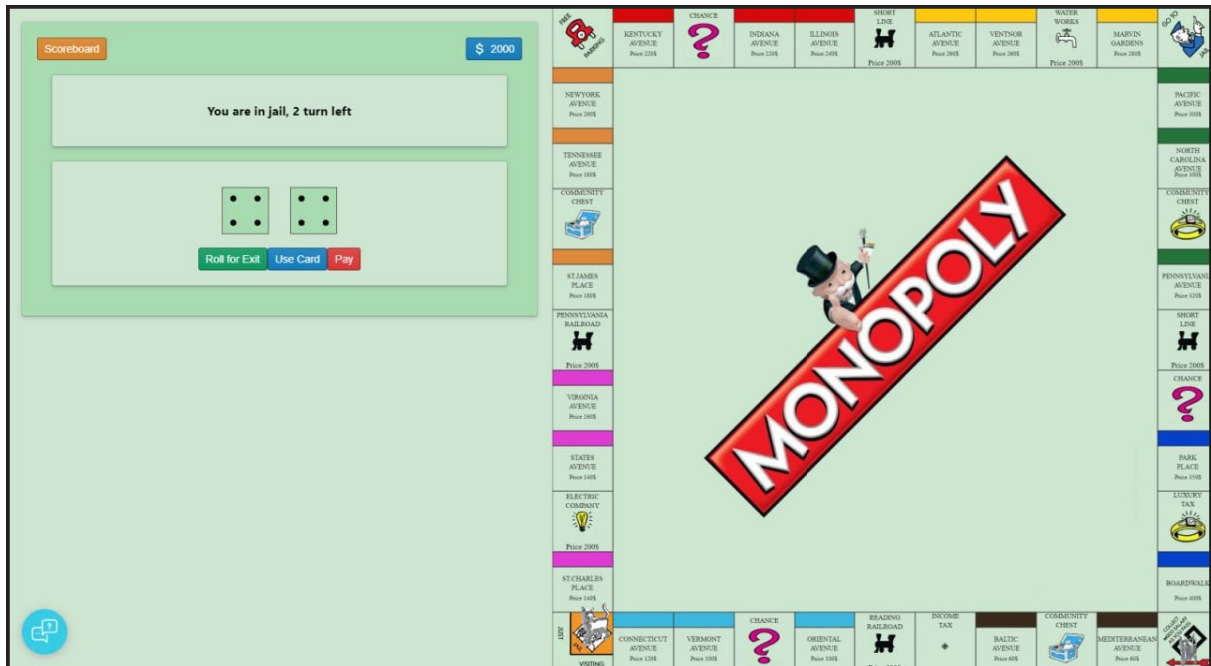


Figure 12: Jail Condition

When a user is in jail, Roll to Exit, Use exit from jail card and Pay buttons appear in order to exit from jail. In addition, the player can see how many turns left.



## 1.13 Scoreboard












Scoreboard		✕	
Username: umityigitbsrn		Money: 200 \$	
City	 2/3	 1/3	
Station	 2/4		
Utility	 2/2		
			
Username: murata42		Money: 200 \$	
City	 1/3	 2/3	 1/3
Station	 2/4		
Utility	 2/2		
			

Figure 13: Score Board

When the user presses the scoreboard button inside the game screen, this screen shows up on the screen. It expands from the left part of the screen. Next to “City” there are tags and numbers for this user. For example, for user with username “umityigitbsrn”, he has 2 cities which are labeled with red and 1 city which is labeled with blue. Next to the station, it shows the number of station properties and next to utility it shows the number of utility properties.

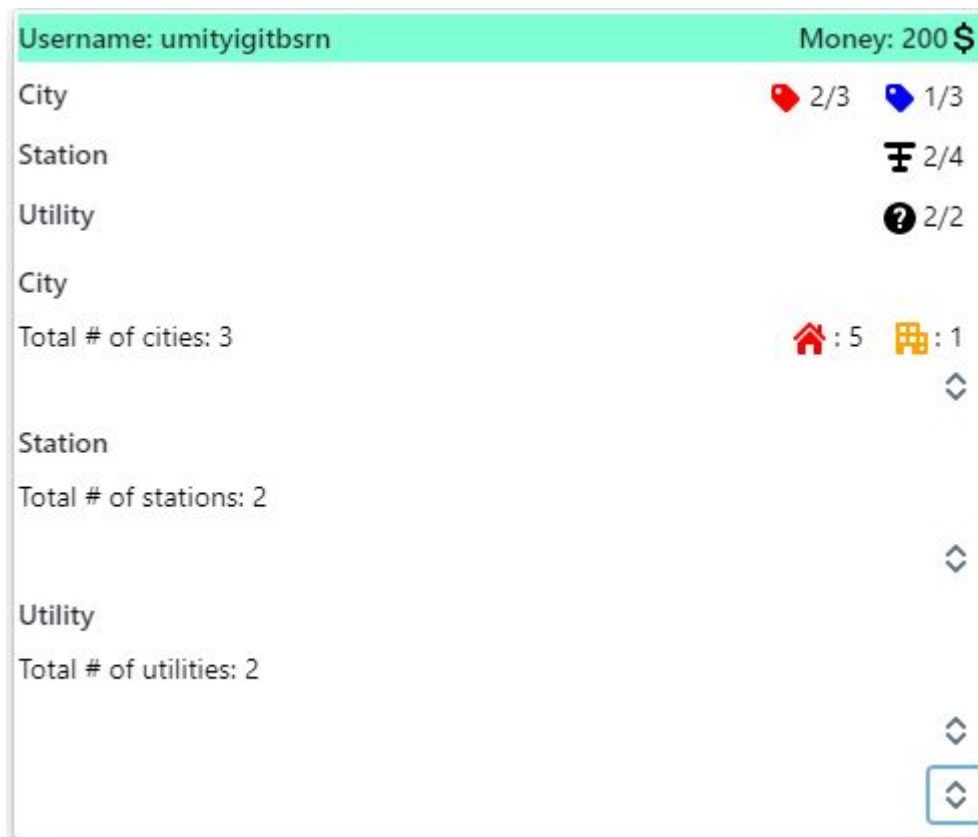


Figure 14: Extended Scoreboard

When the user presses the expand button from one of the cards, that card expands. Below the “City” part it shows total number of cities, total number of houses and total number of hostels. Below the “Station” part, it shows the total number of stations. Below the “Utility” part it shows the total number of utilities.

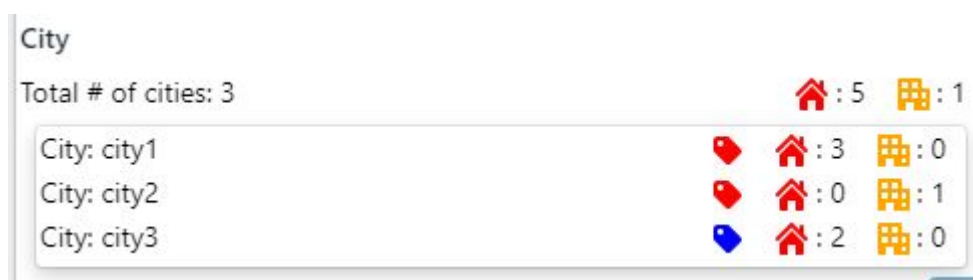


Figure 15: Extended City Part



The image shows a UI component titled "Station". Below the title, it says "Total # of stations: 2". There is a large rectangular text area containing two lines of text: "Station: station1" and "Station: station2". To the right of this text area is a small blue square button with a white double-headed vertical arrow icon.

Figure 16: Extended Station Part



The image shows a UI component titled "Utility". Below the title, it says "Total # of utilities: 2". There is a large rectangular text area containing two lines of text: "Utility: utility1" and "Utility: utility2".

Figure 17: Extended Utility Part

When the user presses extend buttons, then for each part it expands and the user can get detailed information for each part.

## 2. User Manual

### 2.1 General Information

Our current version of the project is the digital version of the Monopoly game standart (USA) version. It is a board game played by 2-6 players.

### 2.2 Build Instructions

You need to install git on your computer in order to clone the repository, or you can download files via a browser. You can install git client for your computer by instructions pointed [here](#).

You need to open your console or terminal and type:

**git clone <https://github.com/muratangin187/CS319-2C-Monopoly>**

After finishing this instructions execution, you will see a folder called CS319-2C-Monopoly in the directory you called git clone instruction. You need to enter CS319-2C-Monopoly/monopoly folder and open a console in order to install all the dependencies needed with:

**npm install**

After this command you can successfully build and run our application via:

**npm run start**

It will bundle all the js and css files and opens the game in development mode. In order create a final executable and run it you need to perform:

**npm run postpackage**

After this command finishes, it will create an executable inside CS319-2C-Monopoly/monopoly/builds/ folder. You can execute the game in production mode via the executable generated in the builds folder.

You need to connect a server in order to play and you can build your own server in the local network via scripts provided in the CS319-2C-Monopoly/server folder. You need to enter here and execute

**npm install**

for the first time. After that you can start your server via:

**npm run start**

We are currently working on the movement and deployment process of a general server for our Monopoly game.

## 2.4 How to Play

After executing the game file, one can start a game, adjust their options or exit the game using the three buttons on the main screen. Their names are given according to their functionality as described in the previous sentence.

In order to start a game, you either need to join an existing room or create a new room. In order to create a new room, required parameters, such as the name of the room, need to be specified in the given text areas. In order to join an existing room, one must also type the password for the room and username.

Options are found in the options menu and can be adjusted depending on players' choices.

The room owner needs to start the game after each player decides on their characters to be used on the board. Starting the game is done by clicking on the start game button on the below side of the character avatars. Meanwhile, players can chat with each other using the chat interface on the left side of the screen.

The game initializes the game and asks for each player to roll dice in order to decide the playing order. Depending on the playing order, each player rolls dice in each turn except if

they did not bankrupt or are in jail. However, players in jail can get out if they roll dice with the same numbers when their turn has come.

Buying and selling houses are done by using the appropriate buttons on the screen. Properties each player owns are shown on their own screens individually and the general scoreboard is shown on the left side of the screen.

Game continues until each player except one, the winner, has money to continue the game. A player is to lose the game if s/he is not the winner and loses all their money.

## **2. Conclusion**

We think that as a group, we worked well and implemented mostly what we wanted. There were some missing parts such as a customizable board and quest system in the game which I thought to implement before the implementation process, however, even though they were missing it was a good product for this scope. We tried to use the engineering and software developer techniques we learned in the course. We tried to use Design Patterns in the implementation process, for instance, we used Façade and Singleton design patterns in manager classes. For instance there is one instance of GameManager class in the runtime, which is an example use of Singleton design pattern.

As a conclusion, we tried our best in the project and did what we could. We are proud of the product we got at the end.

For a final note, we planned to implement an online game and therefore we work with backend, frontend and the server. Even though we implemented the server as we desired, we cannot connect to an online server provider due to time constraints.

