Backgammon++ specification

December 30, 2023

1 Game summary

Backgammon is a strategic game that includes dice rolls and moving tokens across the board. The game lasts until one of the players moves all of their tokens to the goal. Games can be played between bots and/or human players. A played match can be saved for later analysis.

2 Build instructions

2.1 Building instructions

2.1.1 Using QT Creator

Install QT for Open Source Development from qt.io[1]. While installing, make sure to select **Custom installation** and select the following components: - QT 6.6.1 (in the QT dropdown) - QT Multimedia (under QT / Additional Libraries) - QT Creator (under Development and Designer Tools) - CMake (under Development and Designer Tools) - Ninja (under Development and Designer Tools)

Before building, you will need to clone[2] or download[3] the repository.

Now open QT Creator, and click *Open Project*. Navigate to the project folder and inside into the *BackgammonPP* directory, and open *CMakeLists.txt*.

Building and running the project now is as simple as pressing the big green button on the bottom left of the screen.

2.1.2 Using CMake without QT Creator (on Linux)

You will need `git`, `cmake`, and `g++` in order to clone and build the project, as well as two QT dependencies: `qt6-base-dev` and `qt6-multimedia-dev`. For example, on Debian you may run the following command to install the above packages:

```
sudo apt-get install git cmake g++ qt6-base-dev qt6-multimedia-dev
```

Next you will need to to clone the repository, and position yourself inside of it:

```
git clone https://gitlab.com/matf-bg-ac-rs/course-rs/projects-2023-2024/backgammon.git && cd
↪  ./backgammon
```

Now build the project by running the following command:

```
cmake -S BackgammonPP -B ./build && make --directory=build
```

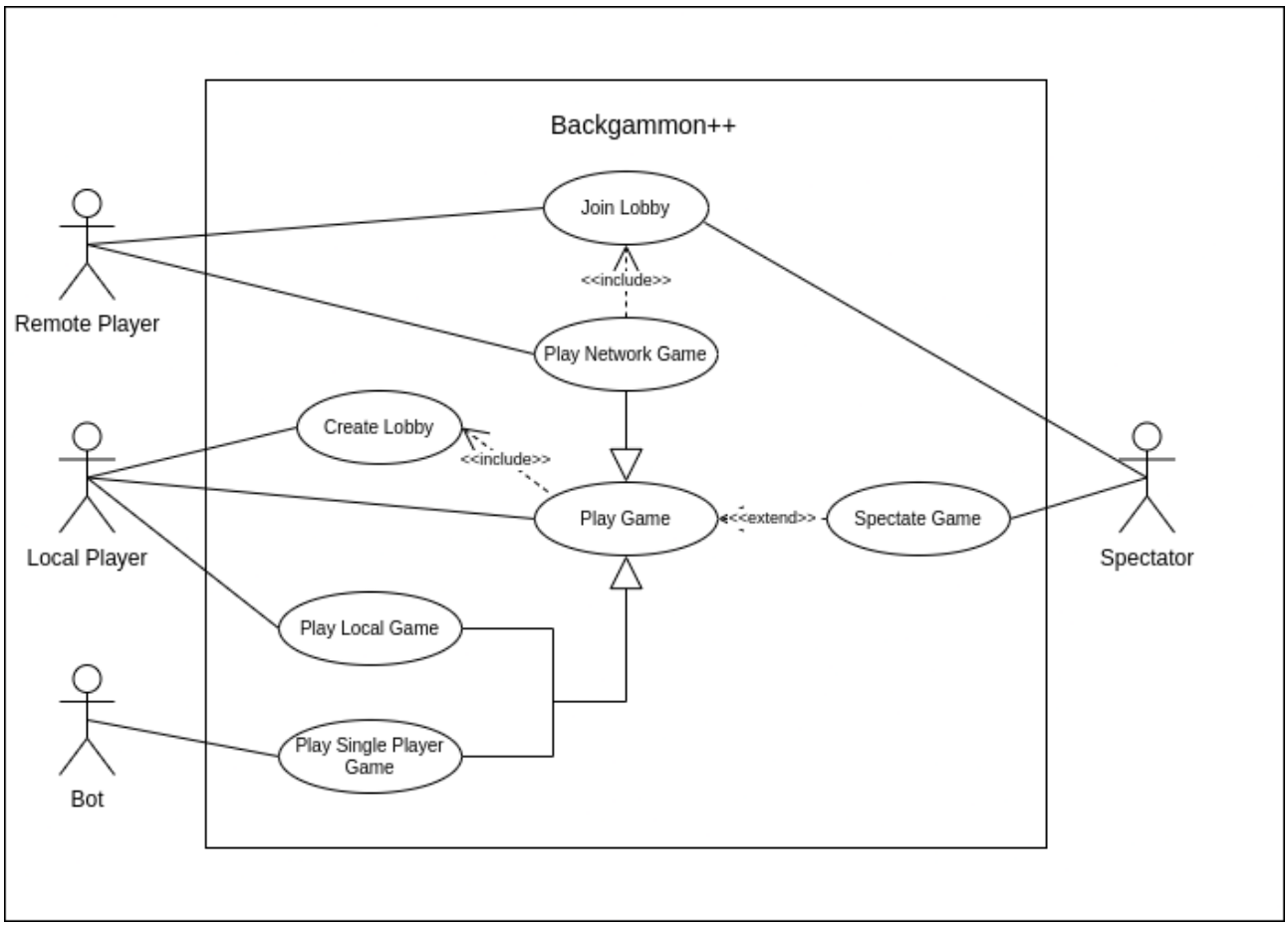
The binary is located inside the build directory, and can be launched from the terminal like this:

```
./build/BackgammonPP
```

[1] <https://www.qt.io/download-open-source>
[2] <https://docs.gitlab.com/ee/gitlab-basics/start-using-git.html#choose-a-repository>
[3] <https://gitlab.com/matf-bg-ac-rs/course-rs/projects-2023-2024/backgammon/-/archive/main/backgammon-main.zip>

3 UML diagrams

4 Use case diagram



5 Sequence diagrams

6 Name: Create Lobby

6.0.1 Short description

- The Local Player creates game lobby where he chooses game options

6.0.2 Actors

- Local Player

6.0.3 Pre-Conditions

- The application is launched and displays the menu

6.0.4 Post-Conditions

- The game is started according to the selected options

6.0.5 Main Flow

```
1. The Local Player clicks 'Create Lobby' button from the menu.
2. The Local Player inputs his username.
3. The Local Player chooses type of his opponent:
    3.1. If the Local Player decides to play against another Local Player:
        3.1.1. Other Local Player inputs his username.
        3.1.2. If there are remote users connected:
            3.1.2.1. Remote users proceed to 'Spectate Game' use case.
        3.1.3. Proceed to 'Play Local Game' use case.
    3.2. If the Local Player decides to play against Bot:
        3.2.1. If there are remote users connected:
            3.2.1.1. Remote users proceed to 'Spectate Game' use case.
        3.2.2. Proceed to 'Play Single Player Game' use case.
    3.3. If the Local Player decides to play against Remote Player:
        3.3.1. The Local Player chooses his opponent from the list of remote users.
        3.3.2. If there are other remote users connected:
            3.3.2.1. Other remote users proceed to 'Spectate Game' use case:
        3.3.3. Proceed to 'Play Network Game' use case.
```

6.0.6 Alternative Flows

```
A2: The Local Player has chosen to play against a Remote Player but the Remote Player has
↪  not yet connected. The Local Player can choose to wait or proceed to alternative flow
↪  A2. End of path.

A2: The Local Player exits the Create Lobby. If there are remote users connected notify them
↪  about cancellation. End of path.
```

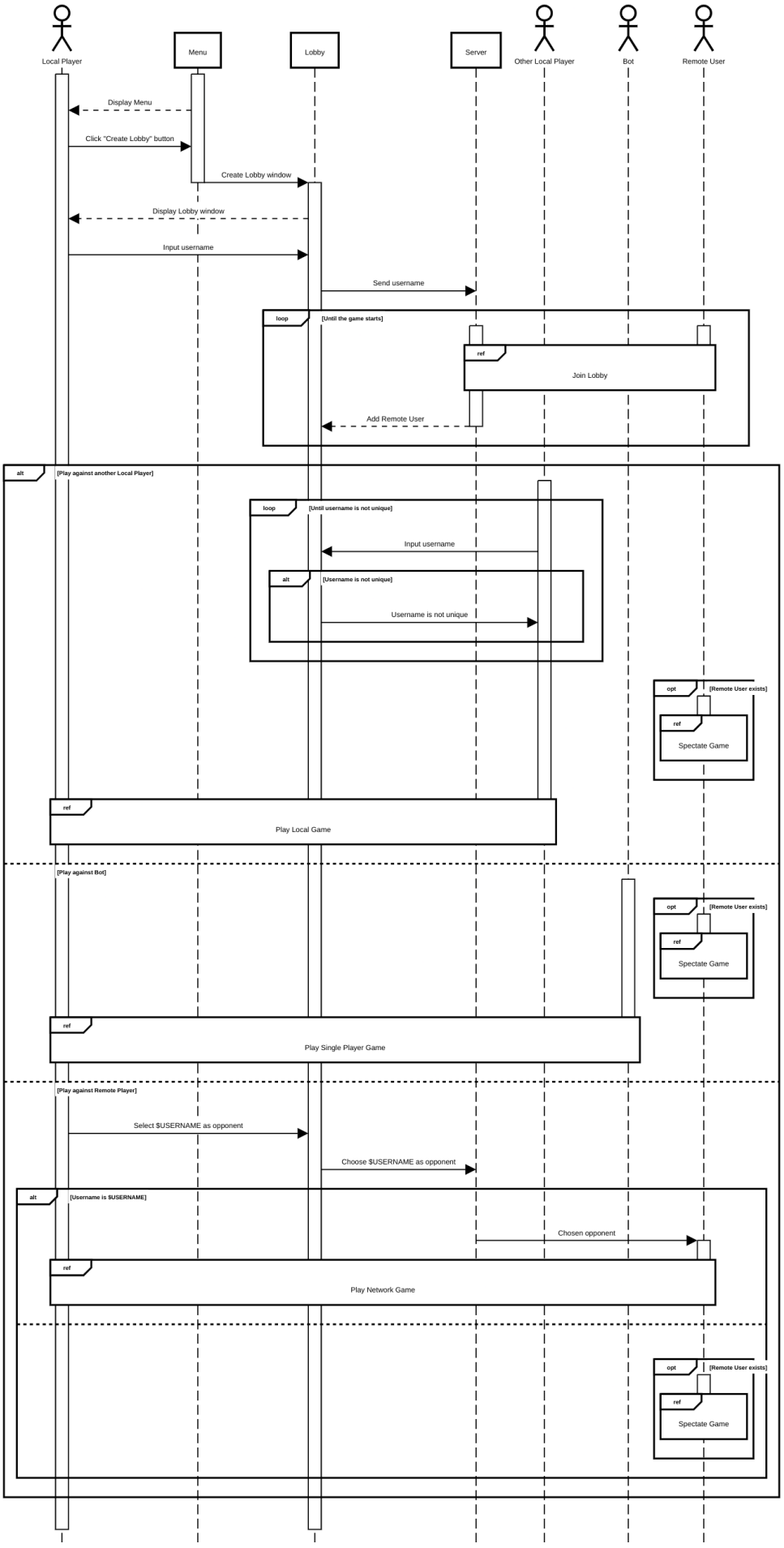
6.0.7 Subflows

- None

6.0.8 Additional info

```
Before game, player must notify remote users about his IP address through other means.
```

Create Lobby



7 Name: Join Lobby

7.0.1 Short description

- User enters nickname and host IP address.
- User connects to host.
- Depending on if the host choose this user, they can either Play Network Game or Spectate Game.

7.0.2 Actors

- Remote Player
- Spectator

7.0.3 Pre-Conditions

- Host creates lobby.
- Host selected to play network game.

7.0.4 Post-Conditions

- User is either spectator or remote player.

7.0.5 Main Flow

```
1. User selects Join Lobby in Main Menu.
2. User inputs username and host's IP address.
3. Check if IP address is valid
    3.1 If not, let the user know and return to step (2)
    3.2 Otherwise, proceed to step (4)
4. Check if Username is unique
    4.1 If not, let the user know and return to step (2)
    4.2 Otherwise, proceed to step (5)
5. Wait for host to choose opponent
    5.1 If host selected the user, proceed to "Play Network Game" use case
    5.2 Otherwise, proceed to "Spectate Game" use case
```

7.0.6 Alternative Flows

```
A1: Player exits the application before host chooses opponent. Remove player's name from
↪  host's opponent options. End of path.

A2: Host exits the application after user connected. Let the user know and return to main
↪  menu. End of path.
```

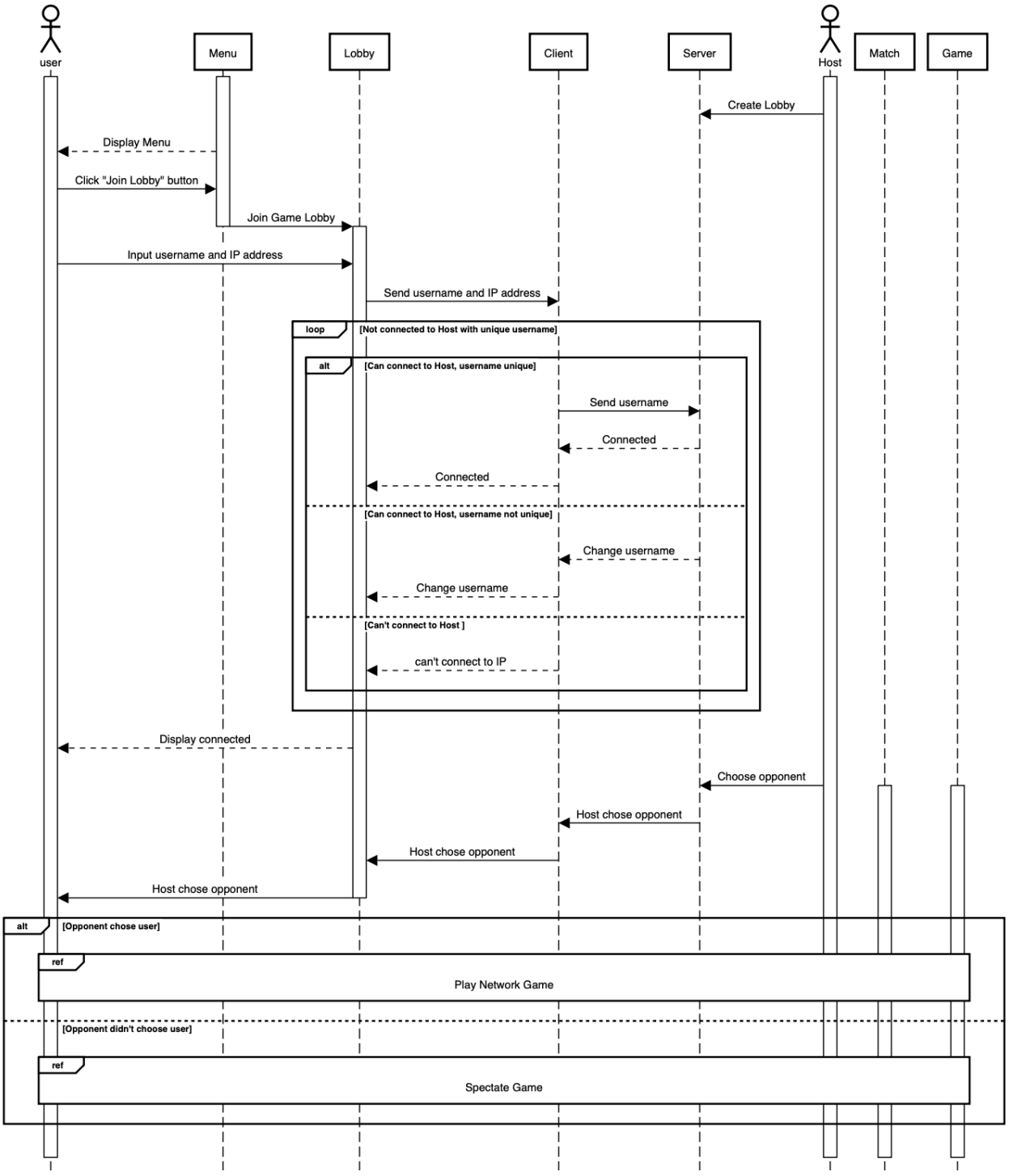
7.0.7 Subflows

- None

7.0.8 Additional info

```
Before game, player must be made aware of IP address of the host through other means.
```

Join Lobby Sequence Diagram



8 Name: Play Game

8.0.1 Short description

- Player rolls their dice, and chooses a valid move.
- Once finished, player ends turn, and their opponent continues.
- An opponent can be either the computer, or another player.

8.0.2 Actors

- Player

8.0.3 Pre-Conditions

- Game options are selected
- The match (single- or multiplayer) is initialized

8.0.4 Post-Conditions

- Game replay (log) is saved

8.0.5 Main Flow

```
1. Player color is determined
    1.1 If WHITE, skip to (2)
    1.2 If BLACK, skip to (10)
2. Display the board
3. Determine if doubling possible
4. Player decides whether to use the doubling cube
    4.1. If doubling, prompt the opponent
        4.1.1. If opponent accepts, double the stakes, and opponent takes ownership of the
        ↪  doubling cube
        4.1.2. If opponent rejects, jump to (13)
5. Roll the dice
6. Determine valid moves
7. Player plays their turn
    7.1. Player chooses a valid move
    7.2. Player decides, whether to end turn or undo the chosen move
        7.2.1. If turn ended, proceed to step (8)
        7.2.2. If turn undone, return to step (7)
8. Update the game state and board
9. Check if player satisfies winning condition
    9.1. If they do, skip to (13)
10. Wait for opponent to make a move
    10.1. If opponent offers doubling the stakes
        10.1.1. If player accepts, double the stakes, and player takes ownership of the
        ↪  doubling cube
        10.1.2. If player rejects, jump to (13)
11. Check if opponent satisfies winning condition
    11.1. If they do, skip to (13)
12. Return to step (3)
13. Determine the winner of the game
    13.1. If player won, declare player the winner
    13.2. If opponent won, declare opponent the winner
14. Proceed to the "Save Game Replay" use case, when finished, proceed to step (15)
15. Display the post-game screen
16. Prompt the player to rematch
    16.1. If player accepts, prompt the opponent to rematch
        16.1.1. If opponent is another player, wait for them to answer the prompt
            16.1.1.1. If opponent accepts, jump to step (1)
            16.1.1.2. If opponent rejects, jump to step (17)
        16.1.2. If opponent is computer, jump to step (1)
```

16.2. If player rejects, jump to step (17)
17. Return to main menu

8.0.6 Alternative Flows

A1: Player exits the application. If ingame, discard current game. End of path.

A2: Human opponent exits the application during a multiplayer match. Tell the player the opponent disconnected, discard the game, and return to main menu. End of path.

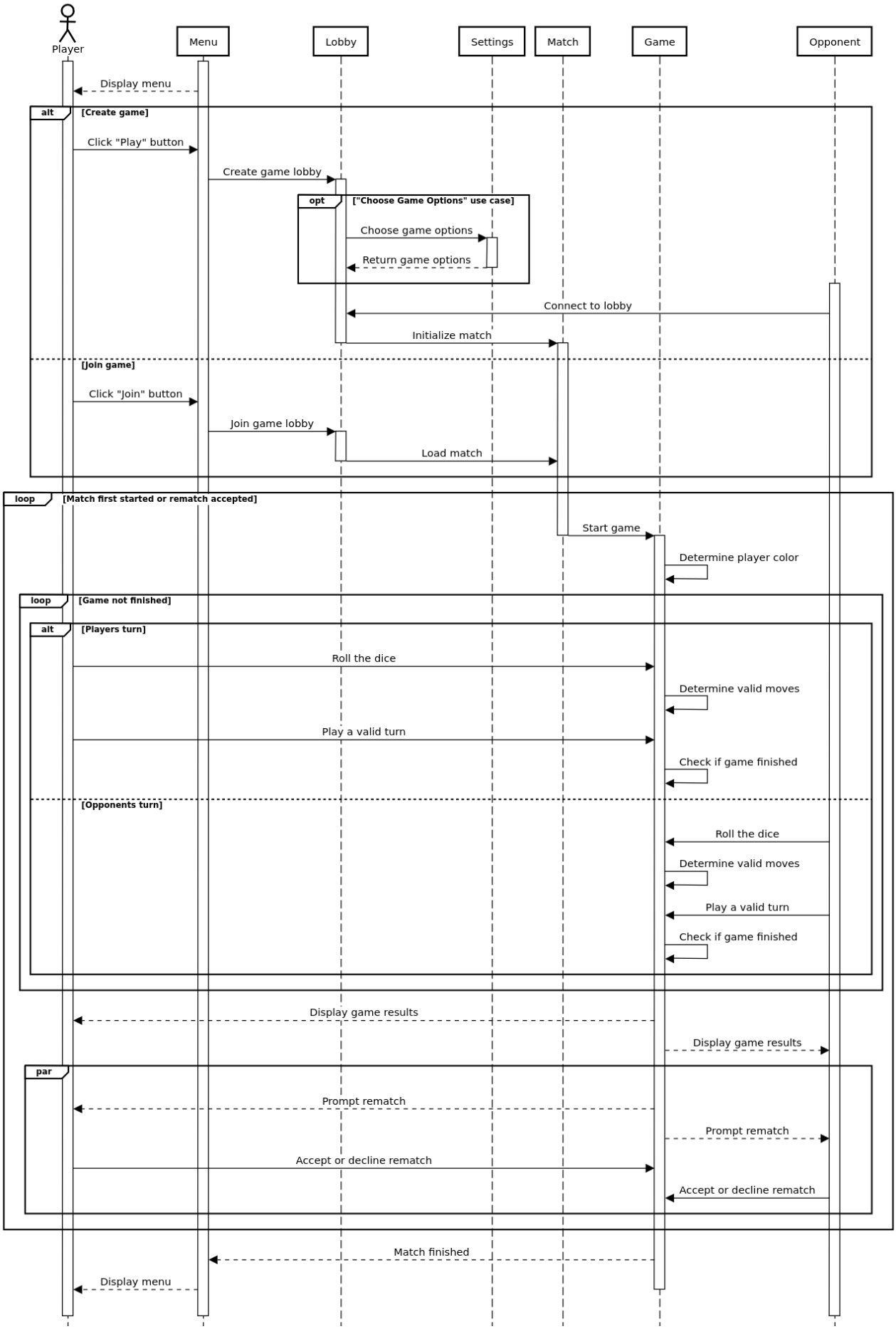
8.0.7 Subflows

- None

8.0.8 Additional info

During the game, each move is saved to persistent storage.

Play Game Sequence Diagram



9 Name: Play Single Player Game

9.0.1 Short description

- Play a game versus a bot

9.0.2 Actors

- Player
- Bot

9.0.3 Pre-Conditions

- Lobby is created
- Player selected to play versus a Bot
- Game is started.

9.0.4 Post-Conditions

- Player is offered to save the game log.

9.0.5 Main Flow

```
1. Initialize the bot from a config file.
2. Determine if the player or the bot goes first by a dice roll.
    2.2 If the player goes first, skip to (3)
    2.3 If the bot goes first, skip to (7)
3. Determine if doubling possible
4. Player decides whether to use the doubling cube
    4.1. If doubling, request a move from the bot
        4.1.1. If the bot accepts, double the stakes, and the bot takes ownership of the
        ↪  doubling cube
        4.1.2. If the bot rejects, jump to (13)
5. Follow steps (5) - (8) from the Play Game use case
6. Check if player satisfies winning condition
    9.1. If they do, skip to (13)
7. Determine if doubling possible
8. The bot decides whether to use the doubling cube.
    8.1 If the bot offers a double, prompt the player to either accept it or reject it.
        8.1.1 If the player accepts, double the stakes, and the bot takes ownership
        8.1.2 If the player rejects, skip to (13)
9. Follow steps (5) - (6) from the Play Game use case
10. The bot plays a valid turn
11. Check if the bot satisfies winning condition
    11.1 If it does, skip to (13)
12. Return to (3)
13. Determine the winner of the game
    13.1. If player won, declare player the winner
    13.2. If the bot won, declare bot the winner
14. Follow steps (14) - (17) from the Play Game use case
```

9.0.6 Alternative Flows

```
A1: Player exits the application. If ingame, discard current game. End of path.
```

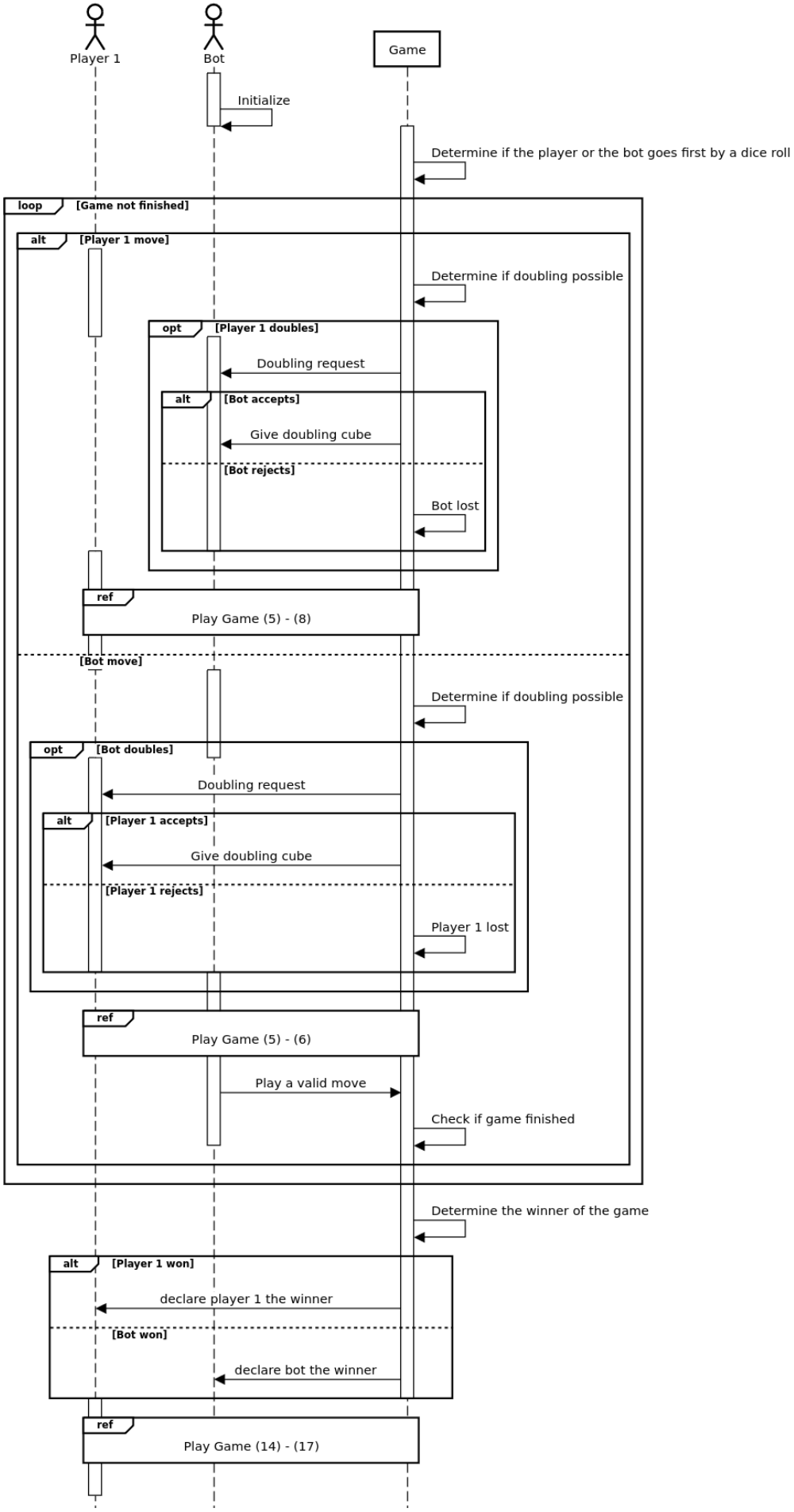
9.0.7 Subflows

- None

9.0.8 Additional info

- None

Play Single Player Game



10 Name: Play Local Game

10.0.1 Short description

- Play a game versus another human opponent on the same computer

10.0.2 Actors

- Player 1
- Player 2

10.0.3 Pre-Conditions

- Lobby is created
- Player 1 selected to play local game
- Game is started

10.0.4 Post-Conditions

- Player is offered to save the game log.

10.0.5 Main Flow

```
1. Determine which player goes first by a dice roll.
    1.1 If player 1 goes first, skip to (2)
    1.2 If player 2 goes first, skip to (6)
2. Determine if doubling possible
3. Player 1 decides whether to use the doubling cube
    3.1. If doubling, request a move from player 2
        3.1.1. If player 2 accepts, double the stakes, and player 2 takes ownership of the
        ↪  doubling cube
        3.1.2. If player 2 rejects, jump to (12)
4. Follow steps (5) - (8) from the Play Game use case
5. Check if player 1 satisfies winning condition
    5.1. If they do, skip to (12)
6. Determine if doubling possible
7. Player 2 decides whether to use the doubling cube.
    7.1 If doubling, request a move from player 1
        7.1.1 If player 1 accepts, double the stakes, and player 1 takes ownership of the
        ↪  doubling cube
        7.1.2 If player 1 rejects, skip to (12)
8. Follow steps (5) - (6) from the Play Game use case
9. Player 2 plays a valid turn
10. Check if player 2 satisfies winning condition
    11.1 If he does, skip to (12)
11. Return to (2)
12. Determine the winner of the game
    12.1. If player 1 won, declare player 1 the winner
    12.2. If player 2 won, declare player 2 the winner
13. Follow steps (14) - (17) from the Play Game use case
```

10.0.6 Alternative Flows

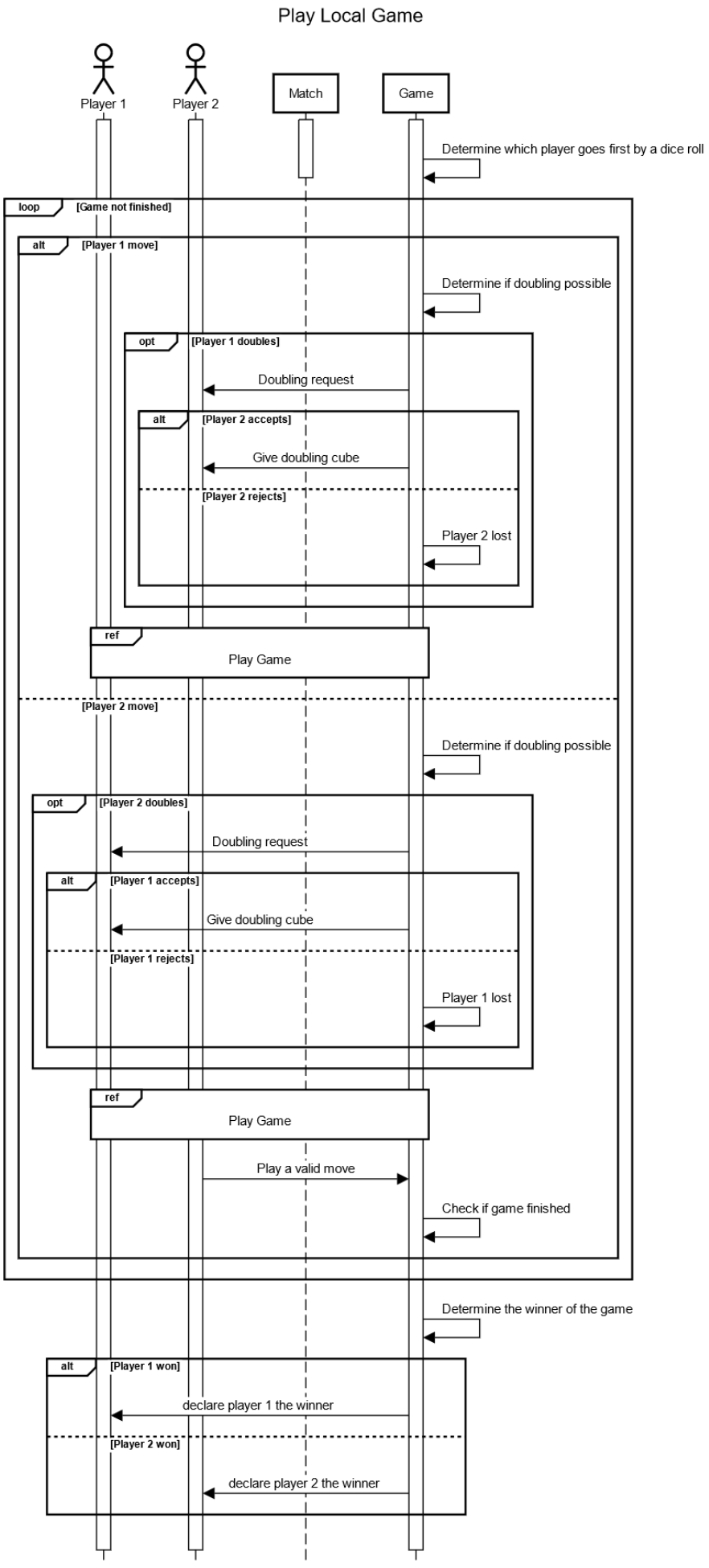
```
A1: Player exits the application. If ingame, discard current game. End of path.
```

10.0.7 Subflows

- None

10.0.8 Additional info

- None


Play Local Game
Player 1
Player 2
Match
Game
Determine which player goes first by a dice roll
loop
[Game not finished]
alt
[Player 1 move]
Determine if doubling possible
opt
[Player 1 doubles]
Doubling request
alt
[Player 2 accepts]
Give doubling cube
[Player 2 rejects]
Player 2 lost
ref
Play Game
[Player 2 move]
Determine if doubling possible
opt
[Player 2 doubles]
Doubling request
alt
[Player 1 accepts]
Give doubling cube
[Player 1 rejects]
Player 1 lost
ref
Play Game
Play a valid move
Check if game finished
Determine the winner of the game
alt
[Player 1 won]
declare player 1 the winner
[Player 2 won]
declare player 2 the winner

11 Name: Play Network Game

11.0.1 Short description

- Play a game versus another human opponent on the different computer

11.0.2 Actors

- Our Player
- Hosting Player

11.0.3 Pre-Conditions

- Game options are selected
- Our player joins the lobby, waiting for hosted game
- Match between them is found

11.0.4 Post-Conditions

- Game replay (log) is saved

11.0.5 Main Flow

```
1. Player color is determined
    1.1 If white, skip to (2)
    1.2 If black, skip to (10)
2. Display the board
3. Determine if doubling possible
4. Player decides whether to use the doubling cube
    4.1. If doubling, prompt the opponent
        4.1.1. If opponent accepts, double the stakes, and opponent takes ownership of the
        ↪  doubling cube
        4.1.2. If opponent rejects, jump to (13)
5. Roll the dice
6. Determine valid moves
7. Player plays their turn
    7.1. Player chooses a valid move
    7.2. Player decides, whether to end turn or undo the chosen move
        7.2.1. If turn ended, proceed to step (8)
        7.2.2. If turn undone, return to step (7)
8. Update the game state and board
9. Check if player satisfies winning condition
    9.1. If they do, skip to (13)
10. Wait for opponent to make a move
    10.1. If opponent offers doubling the stakes
        10.1.1. If player accepts, double the stakes, and player takes ownership of the
        ↪  doubling cube
        10.1.2. If player rejects, jump to (13)
11. Check if opponent satisfies winning condition
    11.1. If they do, skip to (13)
12. Return to step (3)
13. Determine the winner of the game
    13.1. If player won, declare player the winner
    13.2. If opponent won, declare opponent the winner
14. Proceed to the "Save Game Replay" use case, when finished, proceed to step (15)
15. Display the post-game screen
16. Prompt the player to rematch
    16.1. If player accepts, prompt the opponent to rematch
        16.1.1. If opponent is another player, wait for them to answer the prompt
            16.1.1.1. If opponent accepts, jump to step (1)
            16.1.1.2. If opponent rejects, jump to step (17)
        16.1.2. If opponent is computer, jump to step (1)
```

    16.2. If player rejects, jump to step (17)
17. Return to main menu

11.0.6 Alternative Flows

A1: Player can't find hosted game. Wait and after some time, return to the main menu. End of path.

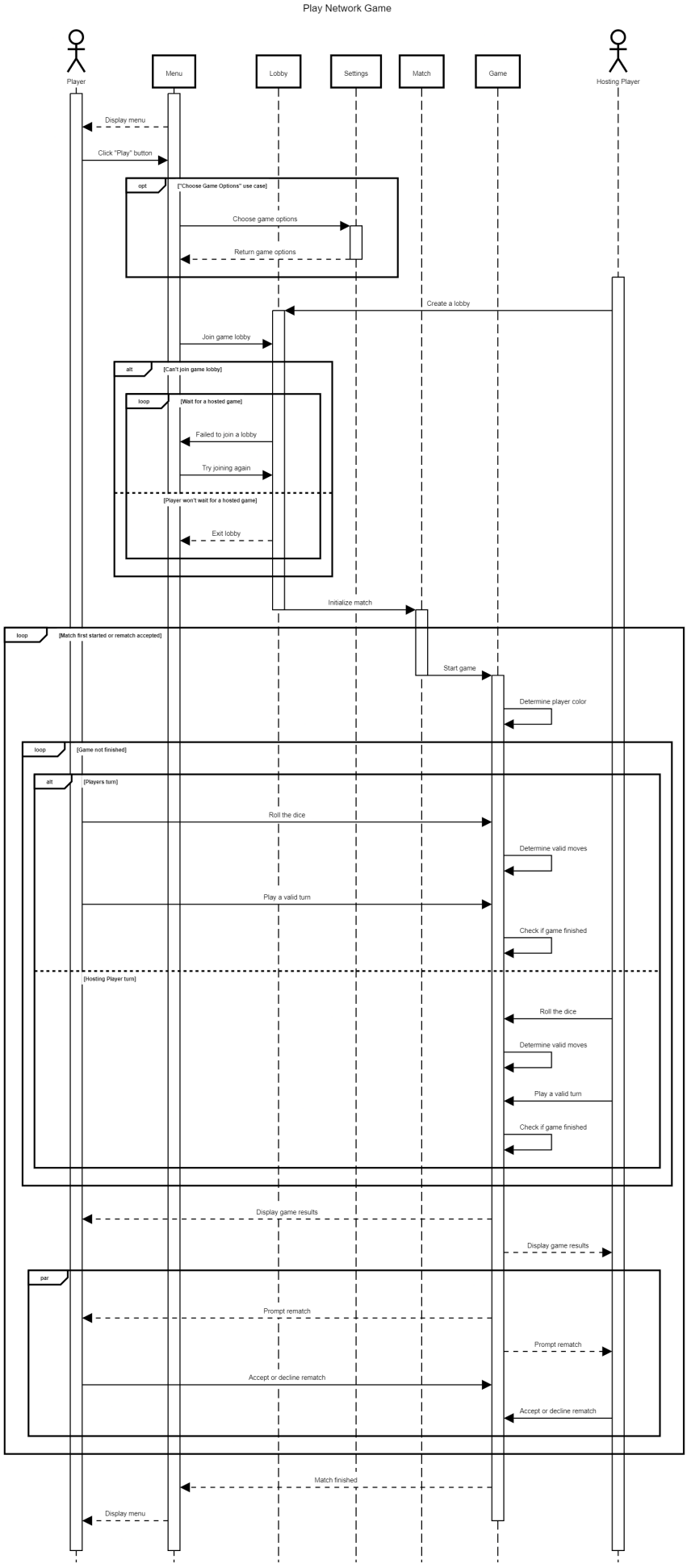
A2: Player exits the application. If ingame, discard current game. Tell the player that the opponent disconnected and that he won the game, discard the game and return to main menu. End of path.

11.0.7 Subflows

- None

11.0.8 Additional info

During the game, each move is saved to persistent storage.


Play Network Game
Player
Menu
Lobby
Settings
Match
Game
Hosting Player
Display menu
Click "Play" button
opt
["Choose Game Options" use case]
Choose game options
Return game options
Create a lobby
Join game lobby
alt
[Can't join game lobby]
loop
[Wait for a hosted game]
Failed to join a lobby
Try joining again
[Player won't wait for a hosted game]
Exit lobby
Initialize match
loop
[Match first started or rematch accepted]
Start game
Determine player color
loop
[Game not finished]
alt
[Players turn]
Roll the dice
Determine valid moves
Play a valid turn
Check if game finished
[Hosting Player turn]
Roll the dice
Determine valid moves
Play a valid turn
Check if game finished
Display game results
Display game results
par
Prompt rematch
Prompt rematch
Accept or decline rematch
Accept or decline rematch
Match finished
Display menu

12 Name: Spectate Game

12.0.1 Short description

- Spectate a joined match
- Chat with other spectators

12.0.2 Actors

- Spectator

12.0.3 Pre-Conditions

- The lobby has been created.
- The user joined the lobby before the match started.
- Other players have been chosen as match participants.
- The match has started.

12.0.4 Post-Conditions

- The user is offered to save the game log.

12.0.5 Main Flow

```
1. The app window displays the spectated game and the chat with other spectators.
2. While the match is still going on, the user can choose to do one of the following:
    1.1 Go backwards/forwards one move in current game.
    1.2 Send message to other spectators, optionally choosing a receiver.
        1.2.1 If the user has chosen a specific receiver, only they will receive the
        ↪  message.
        1.2.2 Otherwise, all other spectators will receive the message.
```

12.0.6 Alternative Flows

```
A1: User exits the application. End of path.
```

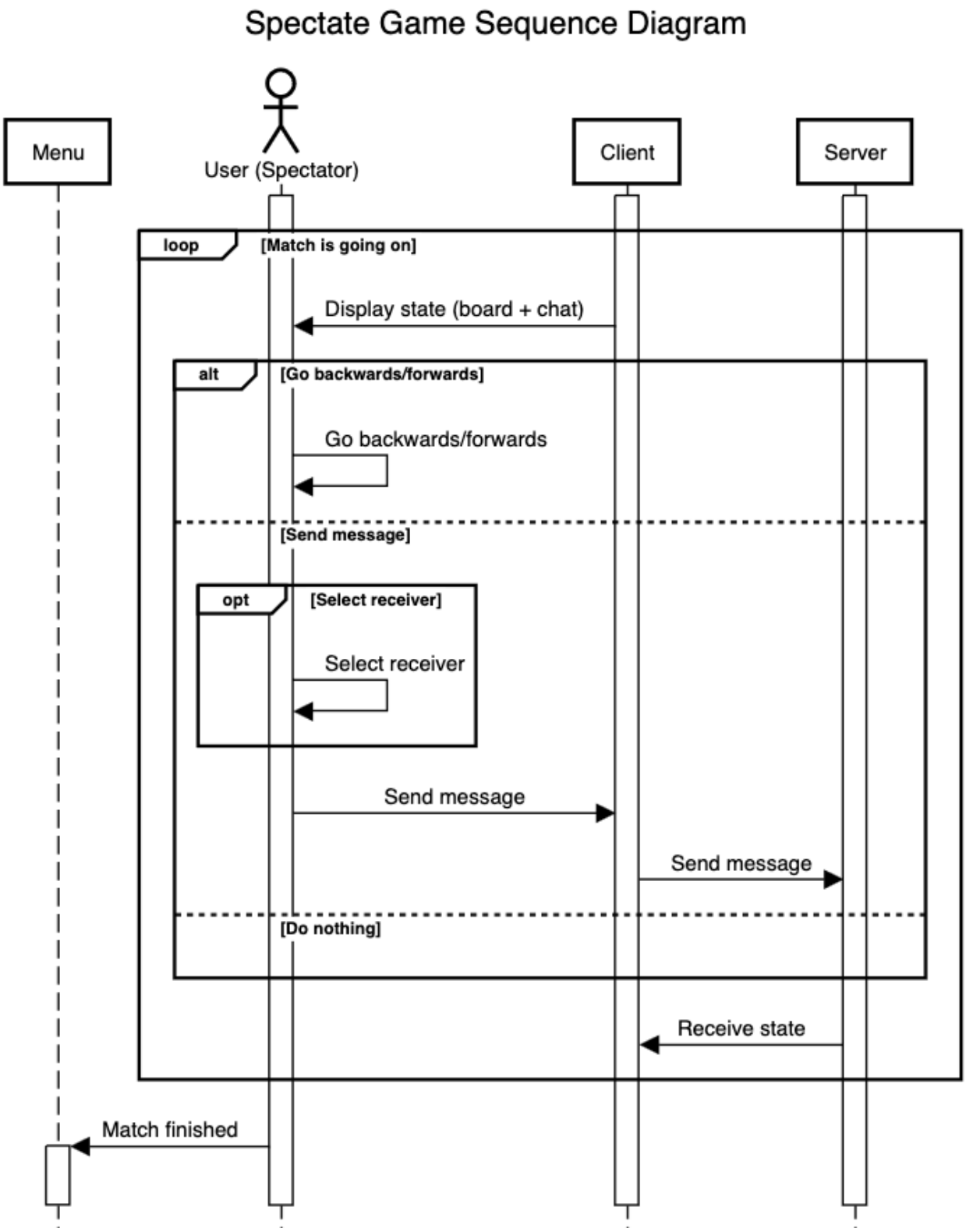
```
A2: Host exits the application after user connected. Let the user know and return to main
↪  menu. End of path.
```

12.0.7 Subflows

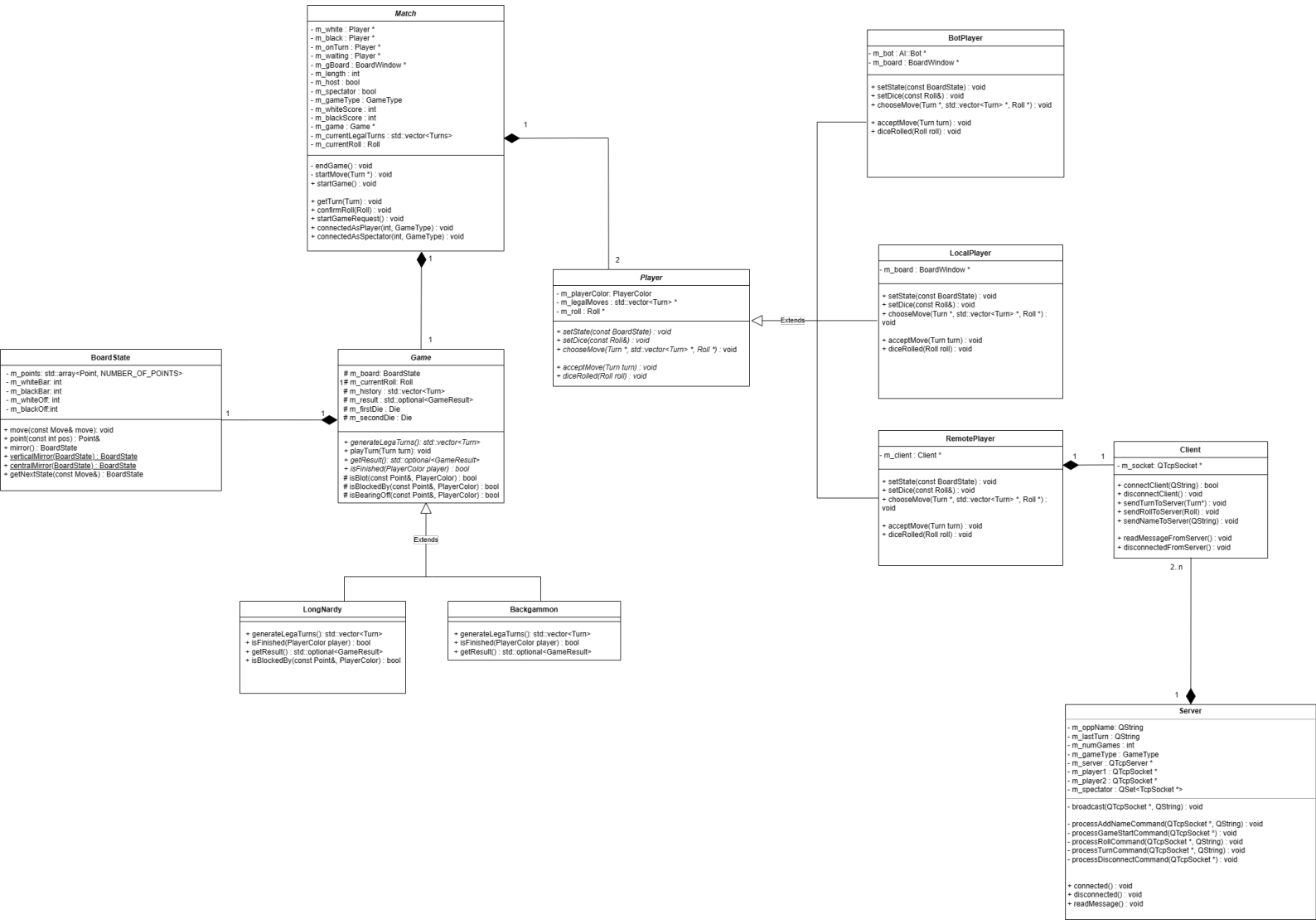
- None

12.0.8 Additional info

- None


Spectate Game Sequence Diagram
Menu
User (Spectator)
Client
Server
loop
[Match is going on]
Display state (board + chat)
alt
[Go backwards/forwards]
Go backwards/forwards
[Send message]
opt
[Select receiver]
Select receiver
Send message
Send message
[Do nothing]
Receive state
Match finished

13 Class diagram



14 Component diagram