**Functional Requirements (Ass 0)**

**Users**

● Register to the system.

● Login and logout from the system.

● Edit user profile which should include an avatar (image of their choice).

● Create Texas Hold'em games.

● Join existing games.

● Spectate active game.

● Leave a game.

● Replay games that are no longer active.

● Find all active games which the user can join.

**Game Center (System)**

● Store all the information from a game, such as: actions performed by all players in

the game, the cards dealt at each round, round beginning and end, etc.

● List all active games which are available for spectating.

● Maintain leagues, managing which users are in which league at any given moment.

**Game**

● support playing a Texas Hold'em game: dealing cards, placing blind bets for players,

allowing players to check (NOP), fold or bet according to the game rules, etc.

**Glossary**

**GameCenter:**

the main Manager of all – contains list of all leagues , default league for new players.

**League**:   
every league has list of Rooms and list of players that belong to that league. Identified by id and name.

**Room:**

Can contain 1 active game of type IGame and list of past games played in that room.   
holds the chat for observers.

**IGame:**

Interface for all types of possible games – for now it’s only TexasGame.

**TexasGame:**

The Poker Game which is being played in a room.

**PlayerAction:**

Handler for players actions like register, login , valid email check etc.

**LeagueManager:**

manages all actions involved with Leagues.

**GamePreferences:**

Each IGame has it’s own instance of preferences. Each GamePreferences contains max number of players , buy in ranges , big blind , small blind etc.

**SearchCenter:**

Contains all searches accessible for players.

**Player:**  
contains the basic player info – user name, password , email etc.

**Spectator:**

A Player who is present in a Room but does not engage in an active game.

**GamePlayer:**

A Player that joined a Game – class adds info to basic Player class that characterizes the Player in the game - chair number , amount of money, currentBet etc.

**Message:**

A stream of characters which is being sent from one player to others.

**Whisper:**

A type of message which private and being sent to a specific receipient.

**Use case name :**   
create Texas Holdem games

**Description :**  
each user can create a game with his unique properties that match the league constrains.  
**Actors :**  
primary actor – the player.

**Pre-conditions :**

The player must have non zero wallet.

**Post-conditions :**

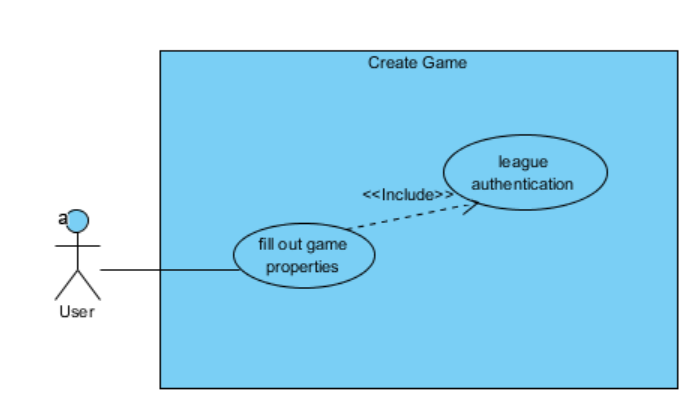
The game created and placed in the right league.

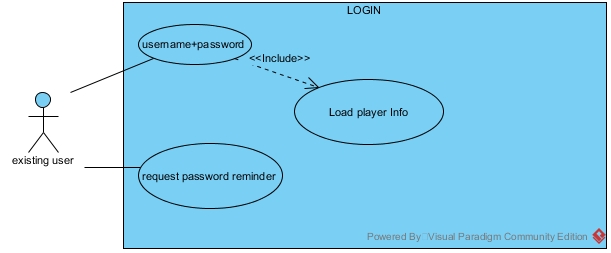
**Main success scenario :**

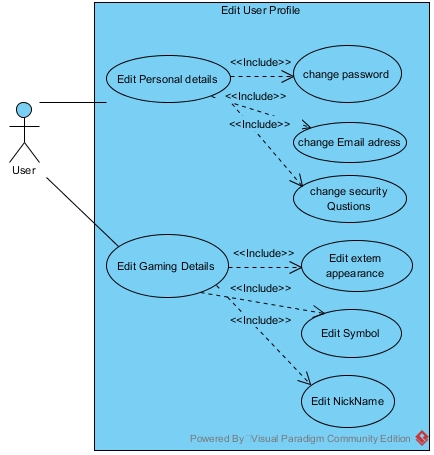
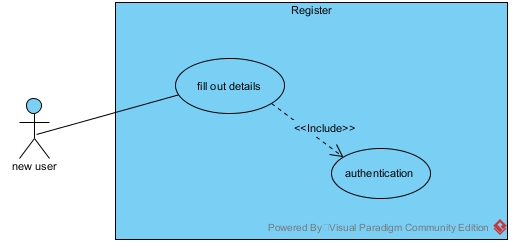
1. The player enters the “Create new Game Menu”  
2. The System offers the leagues that available for the specific user info.  
3. The player defines the game properties (small blind start, max num of players, etc..)  
4. the game is created and added to the league.

**Alternative :**

2. the actor has no money in his wallet.  
 2.1 the system displays an error – the player must wait 24h for his wallet   
to be initialized for a minimum value of 20$ (given to every player that reached 0$)  
**Alternative :**

2. The player created beyond the promitted number of games.  
 2.1 the system will display a warning –   
 “player reached maximum number of games created and still active.  
 new games cannot be created until the old ones are finished or closed”. 





**1. Use case:** Join Existing Game

**1.1** **Brief description:** A user can join an existing game. The user chooses the game he wants to join, the desired seat and amount of money as buy-in and the system will add him to the game.

**2. Actors:** Primary- the user, secondary- the game's system.

**3. Preconditions:** The user is logged in, the game he wants to join isn't full, he has enough money to join.

**4. Postconditions:** The user joins the game and waits for next hand.

**5. Flow of events:**

**5.1 Basic flow:**

**5.1.1.** A user requests to join an existing game.

**5.1.2.** The user chooses a seat in the table.

**5.1.3.** The user enters the desired buy-in.

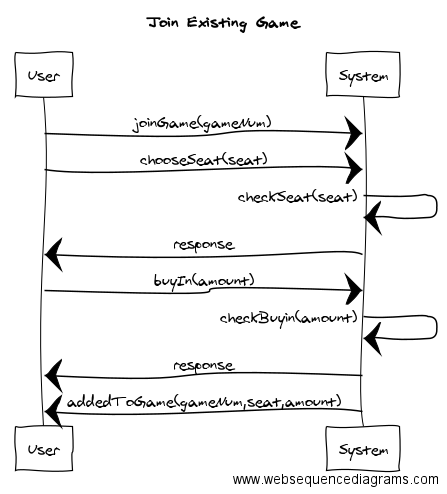
**5.1.4.** The system puts the user in his seat with the amount he entered.

**5.1.5.** The system updates the game log.

**5.2 Alternate flows:**

**5.2.1** Illegal input- If the user does not entered a buy-in that's big enough or enters illegal characters in the buy-in field, the system will print an error message, update the log and tell the user to re-type an amount.

**5.2.2** Full room- If a player chooses to join a game that is full the system should tell him there is no place and would leave him as a spectator.



**1. Use case:** Spectate Active Game

**1.1** **Brief description:** A user can enter to any room as a spectator. The user chooses the game he wants to spectate and the system adds him to the room as a spectator.

**2. Actors:** Primary- the user, secondary- the game's system.

**3. Preconditions:** The user is logged in, the room he wants to join exists.

**4. Postconditions:** The user enters the room as a spectator.

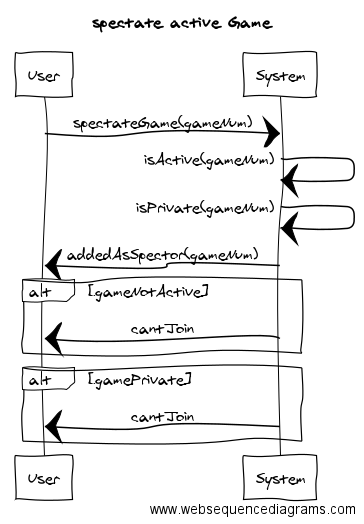
**5. Flow of events:**

**5.1 Basic flow:**

**5.1.1.** A user requests to spectate an existing game.

**5.1.2.** The user enters the room as a spectator.

**5.1.3.** The system updates the game log.



**1. Use case:** Leave a game

**1.1** **Brief description:** A user wants to get out of his chair and leave. All of the user's current bid should go to the pot (the sum of bids in the current hand) and his cards are "burnt".

**2. Actors:** Primary- the user, secondary- the game's system.

**3. Preconditions:** The user is logged in, the user is a player in an active game.

**4. Postconditions:** The user is no longer a part of the game, his bid is added to the pot and the cards are "burnt".

**5. Flow of events:**

**5.1 Basic flow:**

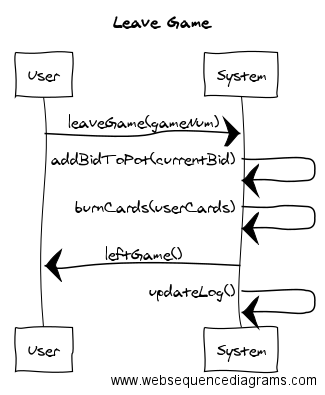
**5.1.1.** A user requests to leave the game.

**5.1.2.** The system adds his bid to the pot.

**5.1.3.** The system burns his cards.

**5.1.4.** The user leaves the room.

**5.1.5.** The system updates the game log.



**1. Use case:** Replay Game

**1.1** **Brief description:** A user can watch a replay of a game that's no longer being played. The user will choose the game he wants to replay, and If it's inactive the system will display it on the user's screen.

**2. Actors:** Primary- the user, secondary- the game's system.

**3. Preconditions:** The user is logged in, the game he wants to watch has ended.

**4. Postconditions:** The replay is displayed on the screen.

**5. Flow of events:**

**5.1 Basic flow:**

**5.1.1.** A user requests to watch a replay of a game.

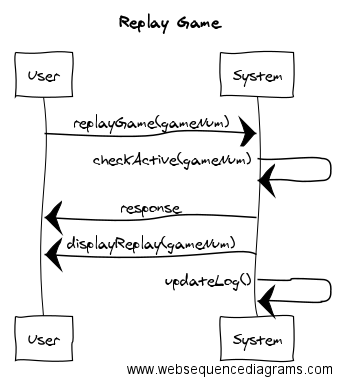
**5.1.2.** The system checks whether the game is still active

**5.1.3.** The system displays the replay on the screen.

**5.1.4.** The system updates the game log.

**5.2 Alternate flows:**

**5.2.1** Active game- If the user requests to watch a game that hasn't yet ended, the system would display a relative message (e.g suggest spectating the game) and would keep the user in the lobby.



**Use Case Name:** Find all active games which the Player\User can join.

**Description:** This scenario describes the process of a Player\User that is currently in the Lobby (the Game Center) searches for rooms in which an active game, that he can join, is taking place.

**Actors:** Player\User, System.

**Pre-Conditions:**

1. The user signed up for the system and is logged in.

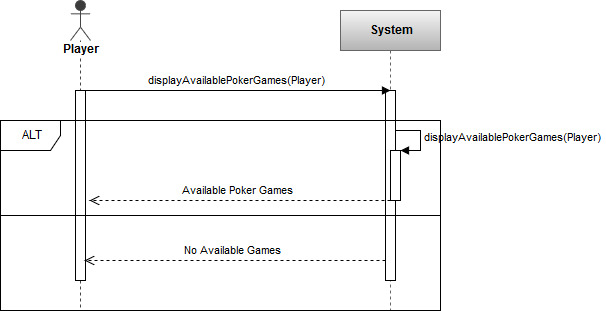
**Post-Conditions:** none.

**Main Success Scenario:**

1. The User\Player searches for active games which he can join.
2. The System analyzes the player’s profile (i.e. level, experience, league and money amount), then it calculates the games he can join accordingly.
3. The System presents the list of available active games to the User.

**Alternatives:**

3a. The System has not find available active games; hence it will present “There are no available games”.



**Use Case Name:** Present all active games available for spectating.

**Description:** This scenario describes in which a logged User wishes to see all the active games that are available for spectating.

**Actors:** User\System.

**Pre-Conditions:**

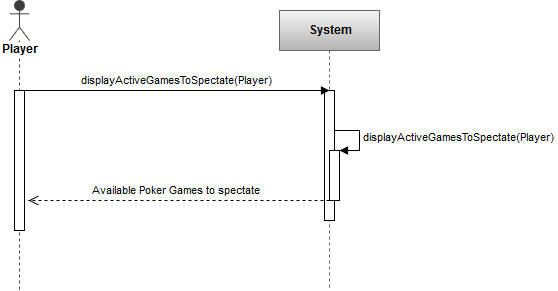
1. The user signed up for the system and is logged in.

**Post-Conditions:** none.

**Main Success Scenario:**

1. The User requests to be presented with all the active games available for spectating.
2. The Systems calculates all the active games available for the User to spectate.
3. The system presents the User with the list of active games available for spectating.

**Alternatives:** none.



**1. Use case:** Player checks

**1.1** **Brief description:** During a hand, if there no need to call, a player can choose to check, meaning he does not want to raise and passes the turn to the next player.

**2. Actors:** Primary- the user, secondary- the game's system.

**3. Preconditions:** The user is in a player in an active game, he hasn't folded yet, no one raised above him.

**4. Postconditions:** The turn passes to the player on the left without any affect to the current pot.

**5. Flow of events:**

**5.1 Basic flow:**

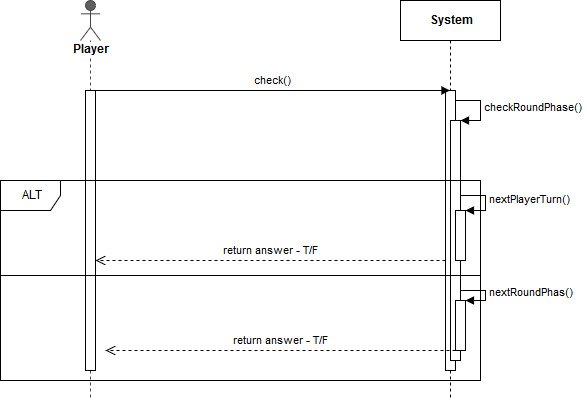
**5.1.1.** A player, only in his turn, chooses to check (all preconditions stand).

**5.1.2.** The system passes the turn to the next player.

**5.1.3.** The system updates the game log.

**5.2 Alternate flow:**

**5.2.1** The system moves to the next round phase.



**1. Use case:** Player calls

**1.1** **Brief description:** During a hand, if there's a bet placed by a previous player that's higher than the current player's bet, he can choose to call, meaning he evens his bet to the current highest bet.

**2. Actors:** Primary- the user, secondary- the game's system.

**3. Preconditions:** The user is in a player in an active game, he hasn't folded yet, a player prior to him placed a higher bet than his current bet.

**4. Postconditions:** The system adds the player's bet to the pot and passes the turn to the next player, the player's amount of money is reduced.

**5. Flow of events:**

**5.1 Basic flow:**

**5.1.1.** A player, only in his turn, chooses to call (all preconditions stand).

**5.1.2.** The system adds his bet to the pot.

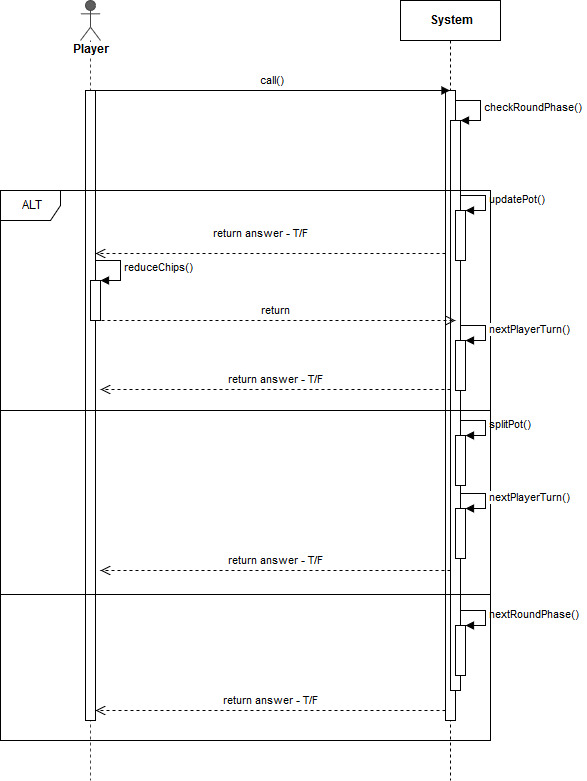
**5.1.3.** The system reduced the amount of money the player has, according to the bet.

**5.1.4.** The system passes the turn to the next player.

**5.1.5.** The system updates the game log.

**5.2 Alternate flows:**

**5.2.1.** Not enough money- if a player wants to call but the current bet his higher than the amount of money he has, he will go all in and the system will then split the pot.



**1. Use case:** Player folds

**1.1** **Brief description:** During a hand, a player can choose to fold, meaning he gave up on his chance to win the hand. All of his current bet goes to the pot and his cards are burnt.

**2. Actors:** Primary- the user, secondary- the game's system.

**3. Preconditions:** The user is in a player in an active game, he hasn't folded yet, it is his turn.

**4. Postconditions:** The system adds the player's bet to the pot, burns his cards, marks him as a folded player in the current hand and passes the turn to the next player.

**5. Flow of events:**

**5.1 Basic flow:**

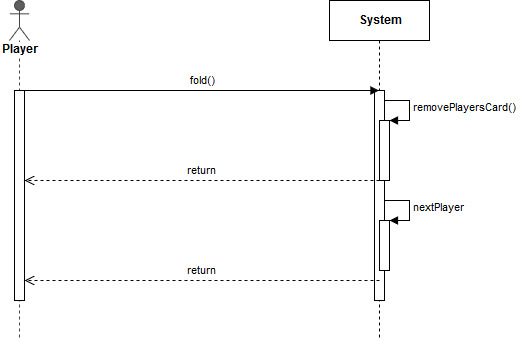
**5.1.1.** A player, only in his turn, chooses to fold (all preconditions stand).

**5.1.2.** The system burns his card.

**5.1.3.** The system marks him as folded in the hand.

**5.1.4.** The system passes the turn to the next player.

**5.1.5.** The system updates the game log.



**1. Use case:** Player raises

**1.1** **Brief description:** During a hand, a player can choose to raise the bet, meaning he is very confident about his chance to win the hand and wants to have a bigger profit. He does so by adding more money to the current highest bet.

**2. Actors:** Primary- the user, secondary- the game's system.

**3. Preconditions:** The user is a player in an active game, he hasn't folded yet, it is his turn.

**4. Postconditions:** The system adds the player's bet to the pot, marks his bet as the current highest bet and passes the turn to the next player.

**5. Flow of events:**

**5.1 Basic flow:**

**5.1.1.** A player, only in his turn, chooses to raise (all preconditions stand).

**5.1.2.** The system adds his bet to the pot.

**5.1.3.** The system reduced the amount of money the player has, according to the difference between the current bet and his previous bet.

**5.1.4.** The system marks his as bet as the current highest bet in the hand.

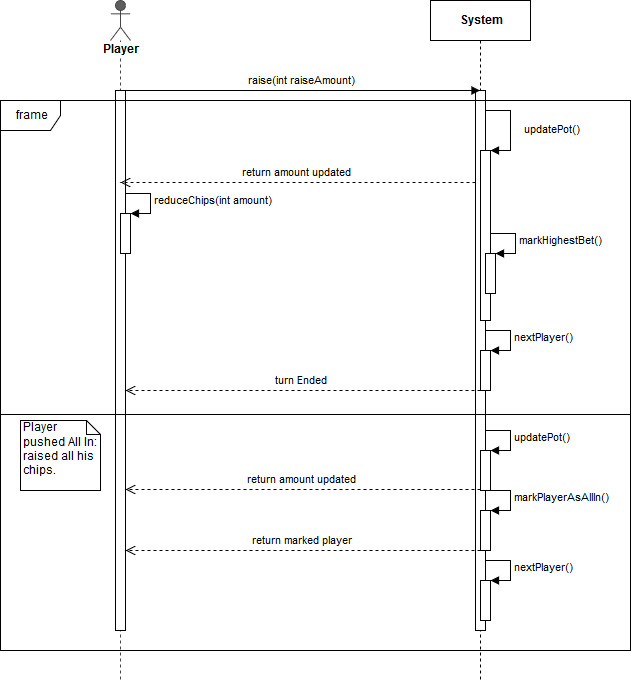
**5.1.5.** The system passes the turn to the next player.

**5.1.6.** The system updates the game log.

**5.2 Alternate flow:**

**5.2.1** All in- if a player is **really** confident with his chances and wants to maximize his profit, he will announce an all in by pushing all his money on the table to the pot in a dramatic motion. The system will mark him as an all in player.

**5.2.2** trying to cheat- if a player tries to cheat the system by betting an amount bigger than the money he owns, the system shall recognize that and tell the player to choose a legal amount, in accordance with the money he has.



Functional Requirements (Ass1)

● Edit user profile

○ Change password / email

● Create game

○ Set game preferences

■ Game type policy: limit, no limit, pot limit

■ Buy-in policy – the cost of joining the game.

■ Chip policy – determine the amount of chips each player is given

(a value of zero means the game is played with real currency).

■ Choose minimum bet (equals to the big blind).

■ Define minimal and maximal amount of players in the table.

■ Choose whether spectating a game (viewing the game without

playing) is allowed or not.

● Search / filter active games

○ By player name

○ By pot size

○ By game preference

**Use Case name:** Edit user profile – change password\email.

**Description:** describes the scenario of a user; that is logged into the system and enters the edit user profile option, choses to change his password\email.

**Actors:** User, System.

**Pre-Conditions:**

1. The user is logged into the system.
2. The user is in the “edit user profile” option.
3. User input is valid.

**Post-Conditions:**

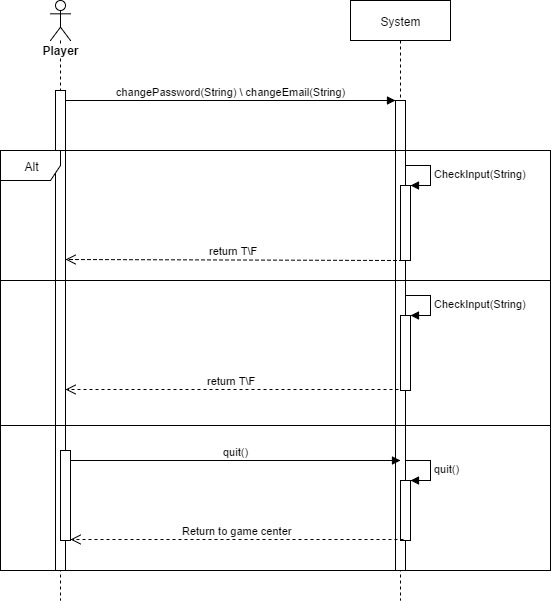
1. User has the new password\email.

**Main success scenario:**

1. A user choses to change his password\email.
2. The system presents the user with an input option for password\email.
3. The user enters a valid password\email and confirms.
4. the system presents the user with a “password\email changed successfully” message.

**Alternatives:**

3. a.1 The user enters an invalid input.  
3. a.2 The system presents an error message.  
3. b.1 The user regrets and cancels the operation.  
3. b.2 The user returns to the Game-Center.

**Use case name:** game type policy.

**Description:** describes the scenario of a user; that is logged into the system and chose to create a new game enters the preferences option, sets his game type policy: limit, no limit, pot limit.

**Actors:** User, System.

**Pre-Conditions:**

1. The user is logged into the system.
2. The user entered “create new game” option.
3. The user entered the preferences option.

**Post-Conditions:**

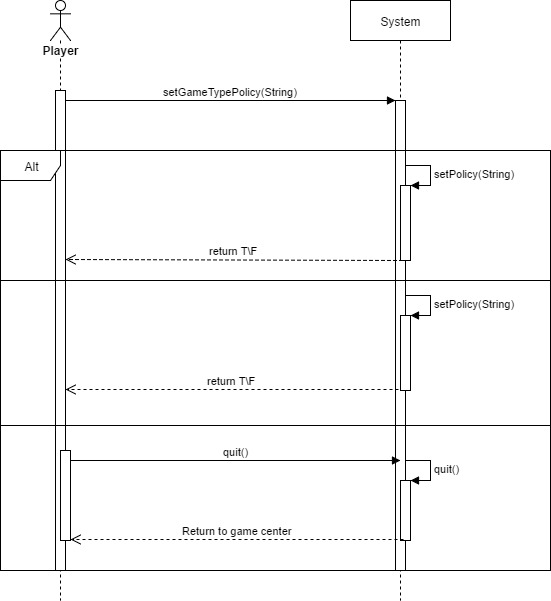
None.

**Main success scenario:**

1. A user choses set game type policy option.
2. The system presents the user with the options: limit, no limit, pot limit.
3. The user choses the desired policy.
4. the system indicates the user that his choice has been received.

**Alternatives:**

3. a.1 The user enters an invalid input.  
3. a.2 The system presents an error message.  
3. b.1 The user regrets and cancels the operation.  
3. b.2 The user returns to the Game-Center.



**Use case name:** Buy-In policy.

**Description:** describes the scenario of a user; that is logged into the system and chose to create a new game enters the preferences option, sets his Buy-In policy – the cost of joining the game.

**Actors:** User, System.

**Pre-Conditions:**

1. The user is logged into the system.
2. The user entered “create new game” option.
3. The user entered the preferences option.

**Post-Conditions:**

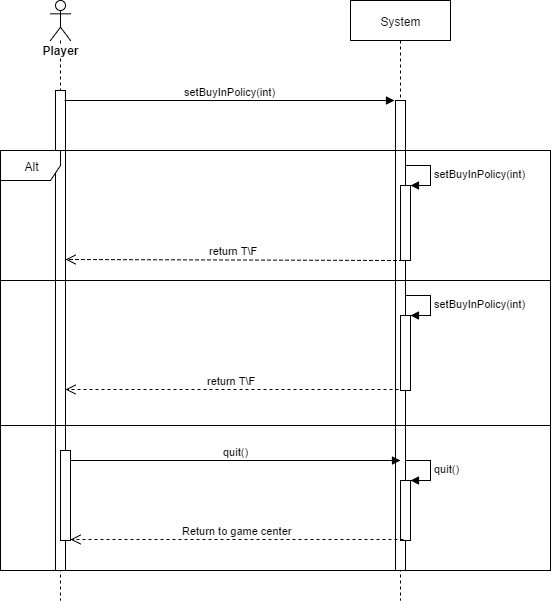
None.

**Main success scenario:**

1. A user choses set Buy-In policy option.
2. The system presents the user with the Buy-In policy.
3. The user enters the amount of money.
4. the system indicates the user that his choice has been received.

**Alternatives:**

3. a.1 The user enters an invalid input.  
3. a.2 The system presents an error message.  
3. b.1 The user regrets and cancels the operation.  
3. b.2 The user returns to the Game-Center.

****

**Use case name:** Chip policy.

**Description:** describes the scenario of a user; that is logged into the system and chose to create a new game enters the preferences option, sets his Chip policy – the amount of chips each player is given (a zero value means real currency).

**Actors:** User, System.

**Pre-Conditions:**

1. The user is logged into the system.
2. The user entered “create new game” option.
3. The user entered the preferences option.

**Post-Conditions:**

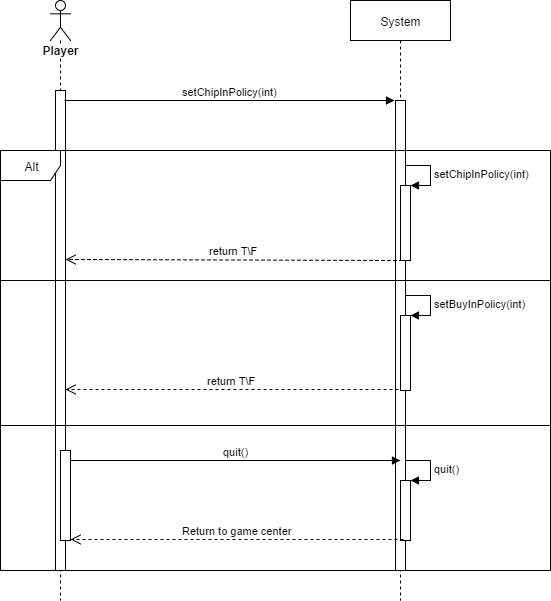
None.

**Main success scenario:**

1. A user choses set Chip policy option.
2. The system presents the user with the Chip policy input option.
3. The user enters the amount chips.
4. the system indicates the user that his choice has been received.

**Alternatives:**

3. a.1 The user enters an invalid input.  
3. a.2 The system presents an error message.  
3. b.1 The user regrets and cancels the operation.  
3. b.2 The user returns to the Game-Center.

****

**Use case name:** Minimum Bet.

**Description:** describes the scenario of a user; that is logged into the system and chose to create a new game enters the preferences option, Chooses minimum bet (equals to the big blind).

**Actors:** User, System.

**Pre-Conditions:**

1. The user is logged into the system.
2. The user entered “create new game” option.
3. The user entered the preferences option.

**Post-Conditions:**

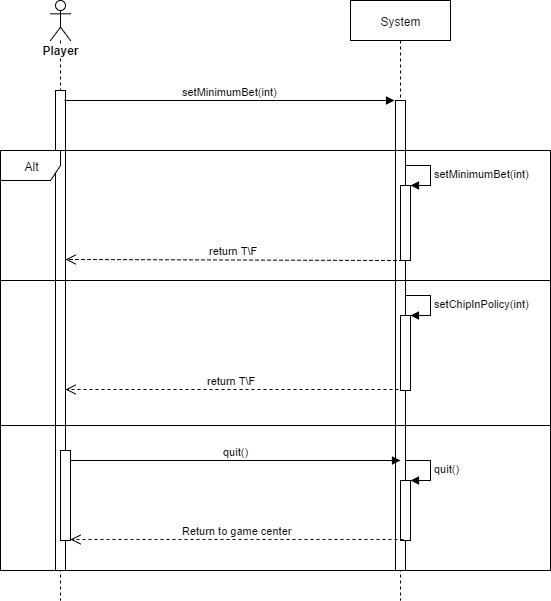
1. Big blind equals to the minimum bet.

**Main success scenario:**

1. A user choses minimum bet option.
2. The system presents the user with input option.
3. The user enters his desired minimum bet.
4. the system indicates the user that his choice has been received.

**Alternatives:**

3. a.1 The user enters an invalid input.  
3. a.2 The system presents an error message.  
3. b.1 The user regrets and cancels the operation.  
3. b.2 The user returns to the Game-Center.

****

**Use case name:** Define minimal and maximal number of players in the table.

**Description:** describes the scenario of a user; that is logged into the system and chose to create a new game enters the preferences option, Defines minimal and maximal number of players in the table.

**Actors:** User, System.

**Pre-Conditions:**

1. The user is logged into the system.
2. The user entered “create new game” option.
3. The user entered the preferences option.

**Post-Conditions:**

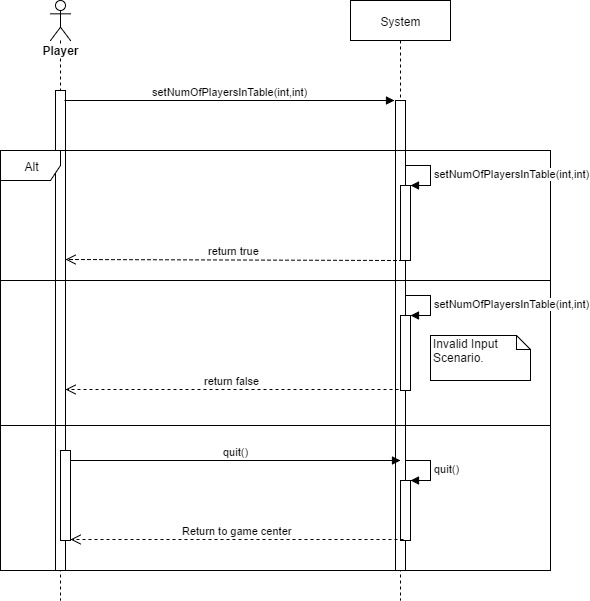
1. None.

**Main success scenario:**

1. A user choses set number of players in the table.
2. The system presents the user with maximal and minimal players input option.
3. The user enters his desired numbers of players.
4. the system indicates the user that his choice has been received.

**Alternatives:**

3. a.1 The user enters an invalid input.  
3. a.2 The system presents an error message.  
3. b.1 The user regrets and cancels the operation.  
3. b.2 The user returns to the Game-Center.

****

**Use case name:** choose whether spectating a game is allowed or not.

**Description:** describes the scenario of a user; that is logged into the system and chose to create a new game enters the preferences option, Decides whether his game is open for spectators or not.

**Actors:** User, System.

**Pre-Conditions:**

1. The user is logged into the system.
2. The user entered “create new game” option.
3. The user entered the preferences option.

**Post-Conditions:**

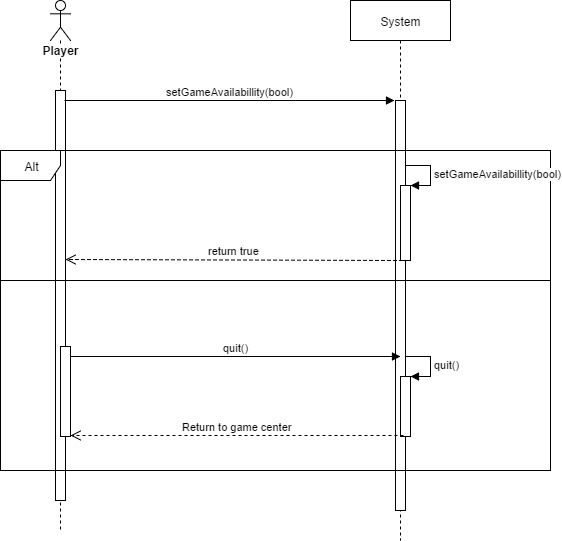
1. None.

**Main success scenario:**

1. A user choses set game availability for spectators.
2. The system presents the user with input option.
3. The user enters his decision (probably yes/no).
4. the system indicates the user that his choice has been received.

**Alternatives:**

3. a.1 The user regrets and cancels the operation.  
3. a.2 The user returns to the Game-Center.



**Use Case Name:** Search games by Player Name.

**Description:** This scenario describes the process of a Player\User; that is currently Logged into the System, looking for a game per a specific Player name.

**Actors:** Player\User, System.

**Pre-Conditions:**

1. The user signed up for the system and is logged in.

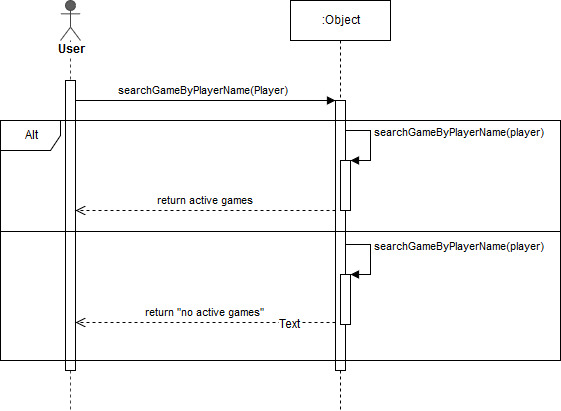
**Post Conditions:** none.

**Main Success Scenario:**

1. The player searches for a game with a Player Name.
2. The System calculates the active games in which the desired Player (Player Name) engages in.
3. The System presents the User with a list of games that satisfies the given criteria.

**Alternatives:**

3a. The System did not find active games with the desired player name; hence it will present “no games found with player: <Player Name>”.



**Use Case Name:** Search games by Pot Size.

**Description:** This scenario describes the process of a Player\User; that is currently Logged into the System, looking for a game per Pot Size.

**Actors:** Player\User, System.

**Pre-Conditions:**

1. The user signed up for the system and is logged in.
2. Pot Size entered is a legal range.

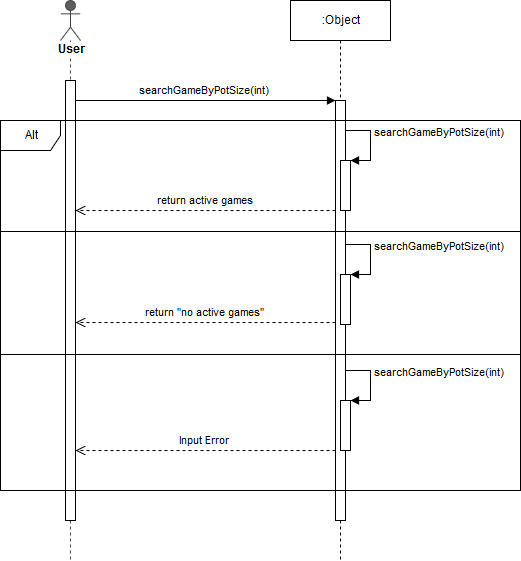
**Post Conditions:** none.

**Main Success Scenario:**

1. The player searches for a game with a Pot Size Range.
2. The System calculates the active games in which the desired Pot Size satisfies the given range.
3. The System presents the User with a list of games that satisfies the given criteria.

**Alternatives:**

3a. The System did not find active games with the desired Pot Size; hence it will present “no games found with Pot Size: <Pot Size Range>”.3b. 1. The user enters an illegal Pot Size Range parameters.  
3b. 2. The System shows the User an Error message.



**Use Case Name:** Search games by Preferences.

**Description:** This scenario describes the process of a Player\User; that is currently Logged into the System, looking for a game matching given preferences.

**Actors:** Player\User, System.

**Pre-Conditions:**

1. The user signed up for the system and is logged in.
2. Preferences are legal values.

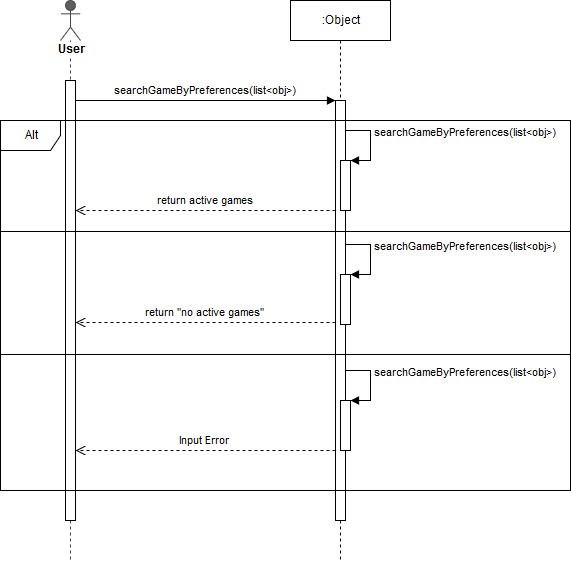
**Post Conditions:** none.

**Main Success Scenario:**

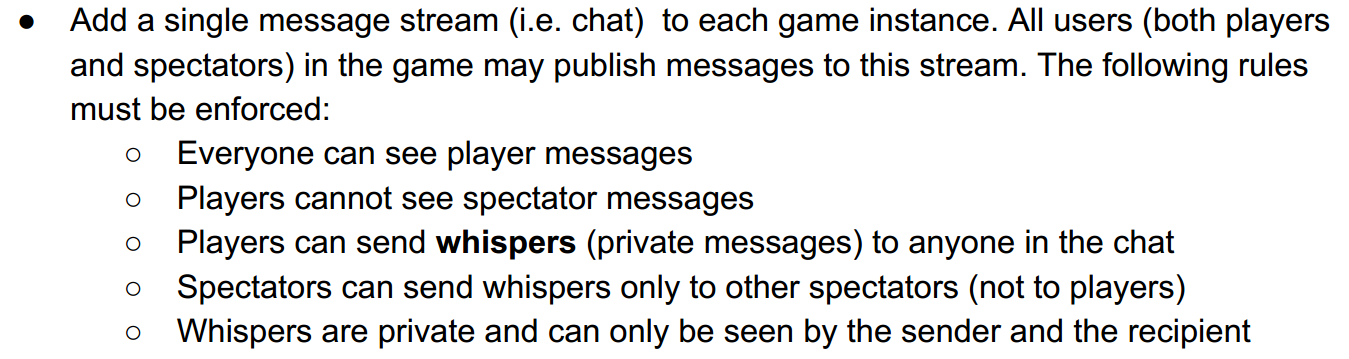
1. The player searches for a game with certain Preferences.
2. The System calculates the active games in which satisfies the given preferences.
3. The System presents the User with a list of games that satisfies the given criteria.

**Alternatives:**

3a. The System did not find active games with the desired preferences; hence it will present “no games found with preferences: <preferences >”.3b. 1. The user enters illegal values as his preferences.  
3b. 2. The System shows the User an Error message.



**Ass2 – New Use Case:**



**Use Case name:** Send GamePlayer Message.

**Description:** A player ,who is present in a game room and sits in an active game, sends a message through the room chat to the other players (both spectators and active players).

**Actors:** Players, Spectators, System.

**Pre Conditions:**

1. All Players are logged into the system.
2. All Players and Spectators are at the same room.
3. The sending player sits in an active game.

**Post Condition:** none.

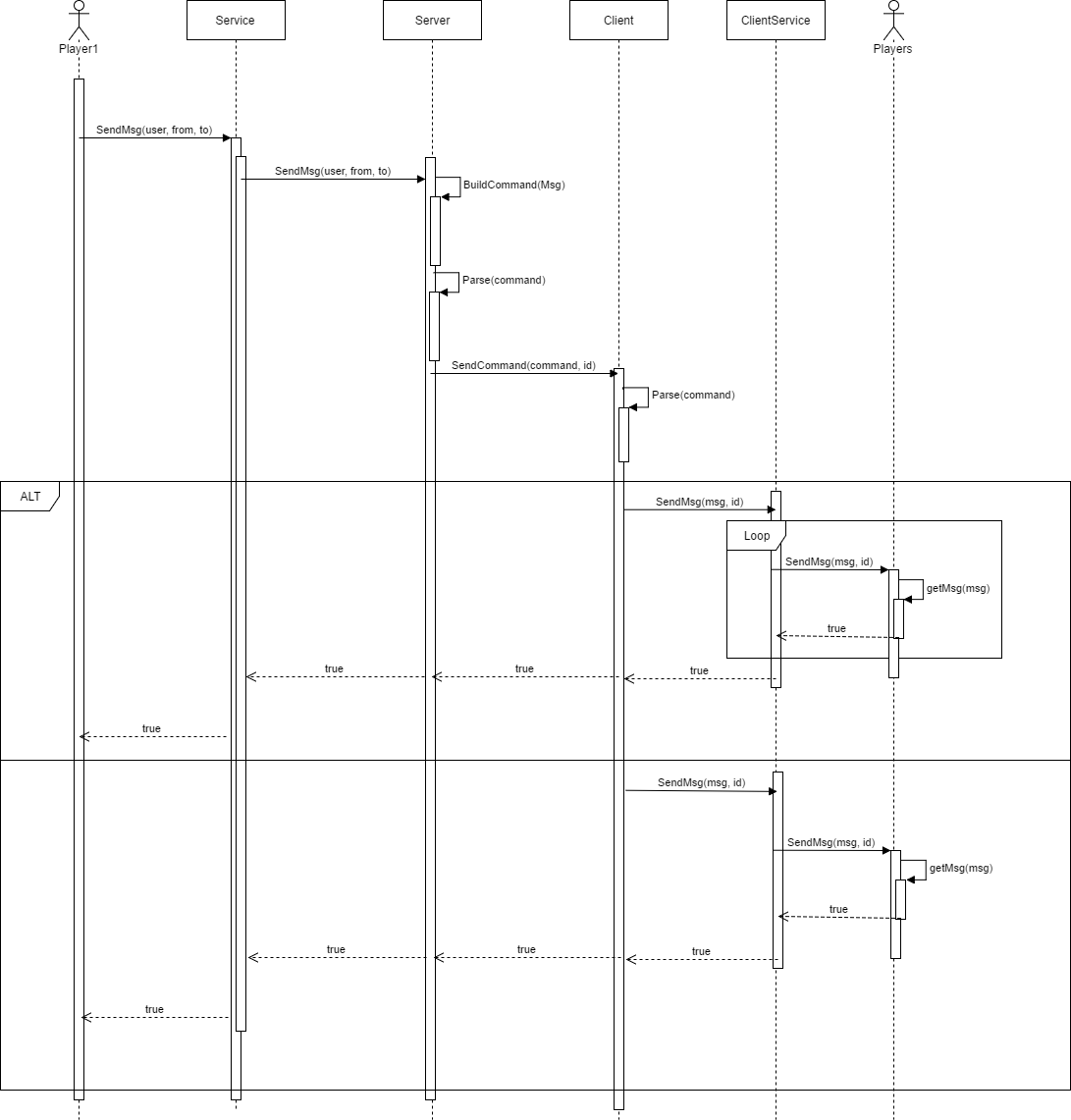
**Main Success Scenario:**

1. The player sends a message to the other players (including spectators) present in the room.
2. The system receives the player’s message and sends it to the server via the Service.
3. The server processes the message and parses it.
4. The server Sends the processed message to the Client.
5. The Client processes the received command from the server.
6. The Client distributes the message to the relevant player via the Client-Service.
7. The Players receives the message.

**Alternative Scenarios:**

1.a. The Player sends a Whisper to a specific receipient (the process continues as before).

6.a. The reeipient cannot be found, the system will send back a suitable message.



**Use Case name:** Send SpectatorPlayer Message.

**Description:** A player ,who is present in a game room and is a spectator, sends a message through the room chat to the other spectators.

**Actors:** Spectators, System.

**Pre Conditions:**

1. All Spectators are logged into the system.
2. All Spectators are at the same room.

**Post Condition:** none.

**Main Success Scenario:**

1. The spectator sends a message to the other spectators present in the room.
2. The system receives the spectator’s message and sends it to the server via the Service.
3. The server processes the message and parses it.
4. The server Sends the processed message to the Client.
5. The Client processes the received command from the server.
6. The Client distributes the message to the relevant player via the Client-Service.
7. The spectators receives the message.

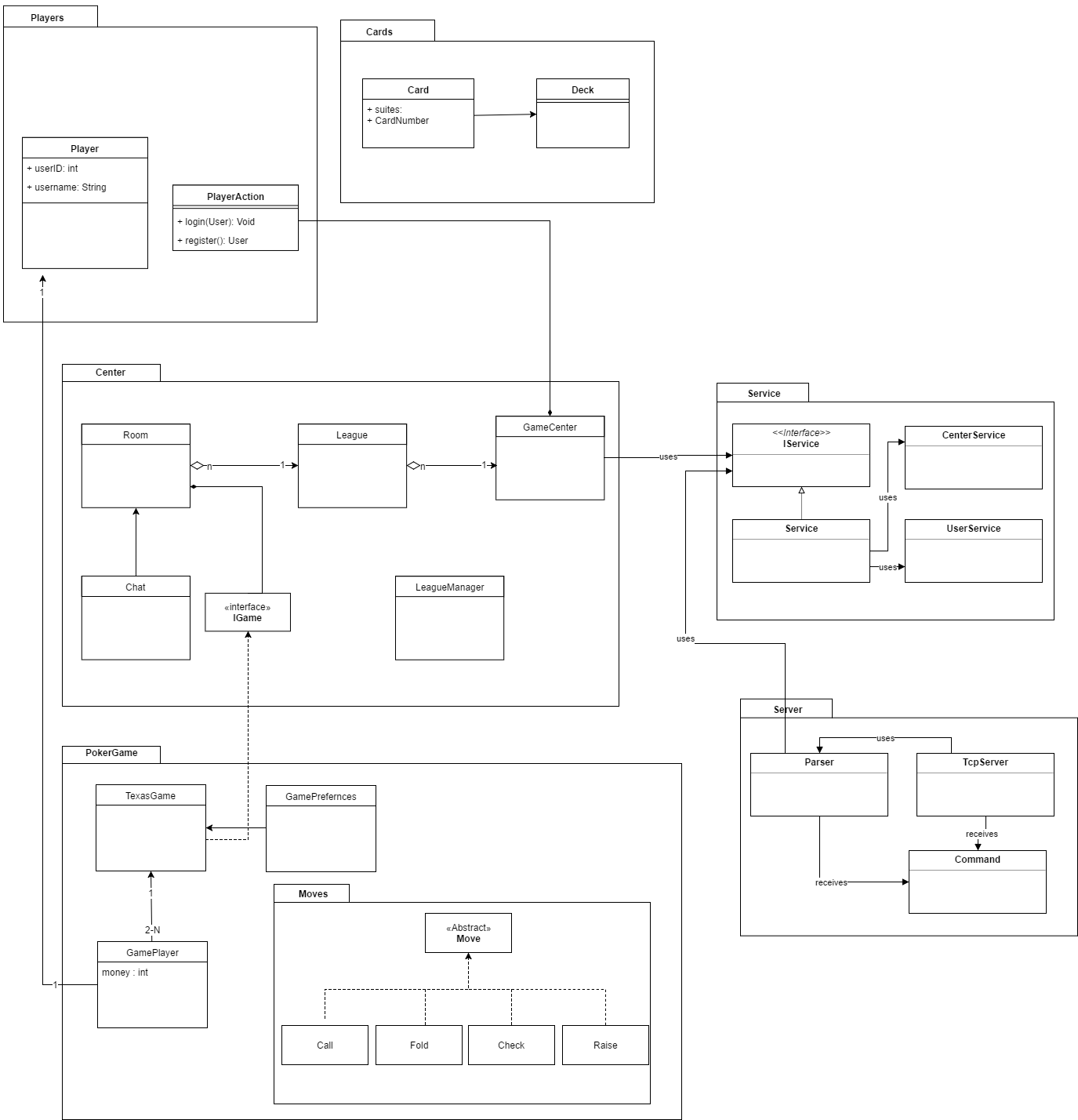
**Alternative Scenarios:**

1.a. The spectator sends a Whisper to a specific sprctator receipient (the process continues as before).

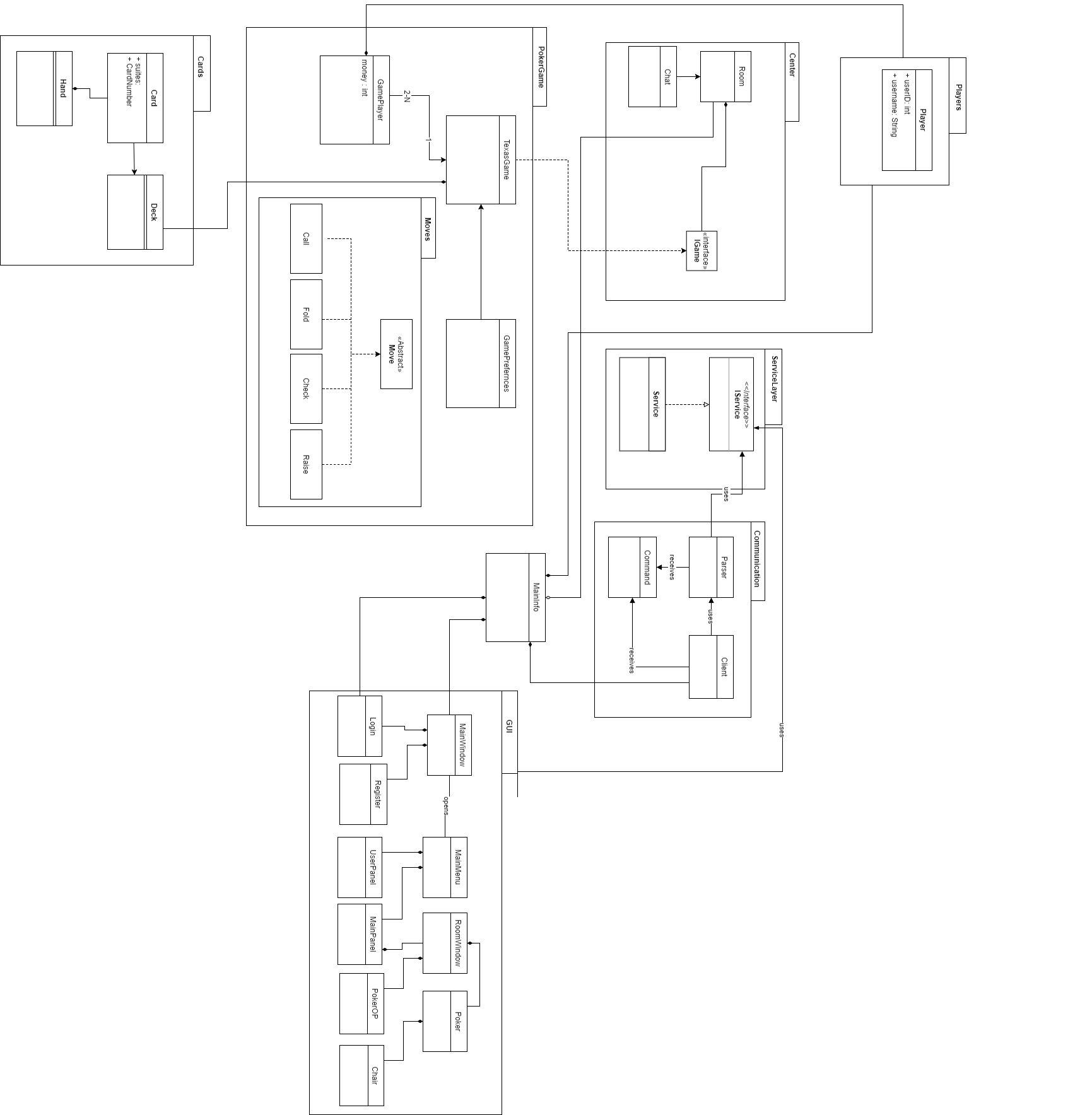
6.a. The reeipient cannot be found, the system will send back a suitable message.

**Class Diagrams:**

Poker:

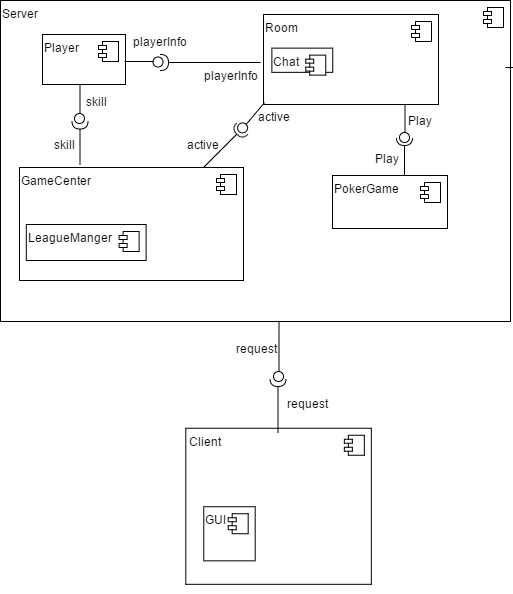


PokerCLient:



The Poker and PokerClient communicate through TCP sockets, using String commands in the corresponding Server and Communication components.

**Component Diagram:**



**Hierarchal GUI Statecharts**

