**Use case name**: Registration to the system.

**Description**: This scenario describes process of a new user registering to the system.

**Actors**: User, System.

**Pre-conditions**: None.

**Post-conditions**:

1. The new user data is saved in the system.
2. The activity log of the system is updated with the relevant information.

**Main success scenario**:

1. The user asked to register to the system.
2. The user enters his personal data:

User name, password, email, date of birth, picture (optional).

1. The system saves the user personal data.
2. The system displays a message for the user, that the registration was successful.

**Alternatives**:

1. a) User name already exists – the system displays appropriate message and the user should enter a new user name in order to proceed.

b) Email or date of birth are not in the correct format – the system displays appropriate message about the incorrect field, and the user should correct relevant mistakes to proceed.

registration sequence

**Use case name**: Login to the system.  
**Description**: This use case describes the process of a registered user logging-in into the system.  
**Actors**: User, System.  
**Pre-conditions**:  
 1. User has a registered account.   
**Post-conditions**:  
 1. The user is logged-in.  
 2. The activity log of the system is updated with the relevant information.  
**Main success scenario**:   
 1. The user enters the username and the password into the system.  
 2. The user is logging-in and the system presents relevant message.  
**Alternatives**:   
 2.a. The username does not exist in the system. The system displays a relevant  
 message. Move to step 1. at main success scenario.  
 2.b. The password is incorrect. The system displays a relevant message. Move to  
 step 1. at main success scenario.

sd-Login (1) (1)

**Use case name**: Logout from the system.  
**Description**: This use case describes the process of a logged in user logging out from the system.   
**Actors**: User, System.  
**Pre-conditions**:  
 1. User is logged-in.   
**Post-conditions**:  
 1. The user is logged-out.  
 2. The activity log of the system is updated with the relevant information.  
**Main success scenario**:  
 1. The user is logging-out the system and the system presents relevant massage.

C:\Users\אחיעד\Downloads\sd-Logout (1).png

**Use case name**: Edit user profile.  
**Description**: This use case describes the process editing an existing user profile.  
**Actors**: User, System.  
**Pre-conditions**:  
 1. User is logged-in the system.   
**Post-conditions**:  
 1. The new user profile details are saved in system's permanent memory.  
**Main success scenario**:   
 1. The system presents the current user profile details: Username, Password, Email  
 address, Date of birth and a personal avatar image.  
 2. The user changes the data he is willing to, except from username which cannot  
 be changed.  
 3. The user saves the changes.   
 4. The new data is saved in the system.  
 5. The system presents relevant massage.  
**Alternatives**:   
 3. The user inserts an invalid email address/date of birth: the system displays a  
 relevant message. Go back to step 2 at main success scenario.

sd-Edit user profile (1)

**Use case name**: Creating new Texas Hold'em game.

**Description**: This scenario describes the process of creating a new Texas Hold'em game, which can be done by a user.

**Actors**: User, System.

**Pre-conditions**:

1. The user is logged in.

**Post-conditions**:

1. A new game is created and the user is added to the game.
2. The activity log of the system is updated with the relevant information.
3. The relevant game instance log is updated with the relevant information.

**Main success scenario**:

1. The user asked to create a new game.
2. The user enters the new game parameters:

Room name, game type policy, minimum buy-in, Chip policy, minimum bet, number of players, whether the game available for spectating or not.

1. The system creates a new game according to the user's selections.
2. The user is automatically added to this game.
3. The system displays a message that the game was created successfully.

**Alternatives**:

1. a) The user entered number of players which is greater than 9 or smaller than   
    2 – The system displays a message that the number of players is not valid   
    and returns to step 2.
2. The user entered minimum buy-in which is greater than the amount he has - The system displays a message that the minimum buy-in should be smaller than the user amount and returns to step 2.
3. The user entered minimum buy-in which is smaller than the big blind - The system displays a message that the minimum buy-in should be greater than the big blind and returns to step 2.
4. The user entered Room name that already exists in the system, returns step 2.



C:\Users\אחיעד\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ad-Creation of a new game.png

**Use case name**: Join existing game.

**Description**: This scenario describes the process of user joining an existing Texas Hold'em game.

**Actors**: User, System, Users in existing game.

**Pre-conditions**:

1. The user is logged in.

2. There are active rooms.

**Post-conditions**:

1. The user is added to the existing game.

2. Amount of players in the room is increased by one.

3. The game log is updated with the info that a new user has joined the game.

**Main success scenario**:

1. The user chooses a game he wants to join.
2. The user enters the buy-in amount.
3. The system adds the user to the room.
4. The system updates the amount of players in the room by one more.
5. The system displays a message that a new user had join the room.

**Alternatives**:

3. a) The room was already full – the system displays a message that the room is   
 already full and returns to step 1.

b) The user does not have enough money to pay the minimum buy-in – The   
 system displays appropriate message and returns to step 1.

c) The user is not in the correct league – the system displays a message that the   
 room league is not fit for the user and return to step 1.

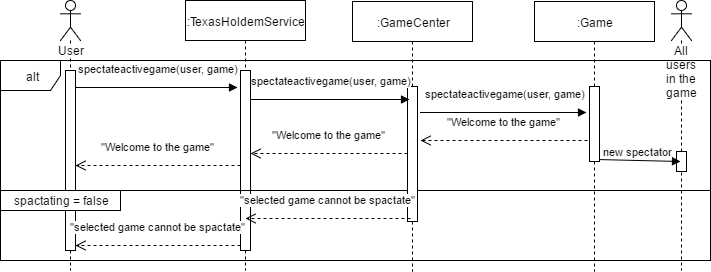
join existing game sequence

C:\Users\אחיעד\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ad-Join existing room.png

**Use case name**: Spectate active game.  
**Description**: This use case describes the process of a user spectates an active game in the game center.   
**Actors**: User, System, Users in room.  
**Pre-conditions**:  
 1. The User is logged-In.   
 2. There are available rooms for spectating.  
**Post-conditions**:  
 1. The User is inside a room, spectating an active game.  
 2. The relevant game instance log is updated with the relevant information.  
**Main success scenario**:  
 1. The User choose one active game, which he wants to spectate.   
 2. The system displays a message that welcomes the user to the game he   
 chose.  
 3. The system displays a massage in the chat of the game, that new user is   
 spectating the game.  
 4. The User is spectating an active game.

**Alternatives**:

2.a. The user choose active game, which users can not spectate. The system displays  
 a message that the user can not spectate this game.



**Use case name**: Leave a game.  
**Description**: This use case describes the process of a user in an active game, which wants to leave the game.   
**Actors:** User, System.  
**Pre-conditions**:  
 1. The User is logged-In.  
 2. The User is playing in an active game.   
**Post-conditions**:  
 1. The game no longer includes the user.  
 2. The relevant game instance log is updated with the relevant information.  
**Main success scenario**:

* + - 1. The User chooses to leave the current game he plays.
      2. The system displays a message that asks the User if he is sure about leaving the game.
      3. The User chooses "Yes".
      4. The system updates the number of players that are in the room.
      5. The system updates the user's balance.
      6. The system displays a massage in the chat of the game, that the user left the game.
      7. The system presents relevant massage to the user.   
         **Alternatives**:   
          3.a. The User chooses "No".



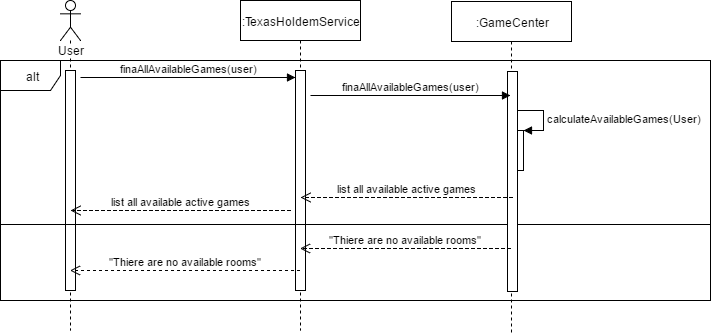
**Use case name**: Replay games that are no longer active.  
**Description**: This use case describes the process of replaying a game that is no longer active.  
**Actors**: User, System.  
**Pre-conditions**:  
 1. User is logged-in the system.   
 2. There are games which can be replayed.  
**Post-conditions**: None.  
**Main success scenario**:   
 1. The user chooses a game from a list of all games which he wishes to see a replay.  
 2. The system displays a replay of the chosen game.

sd-Replay a game

**Use case name**: Find all active games which the user can join.  
**Description**: This use case describes the process of a user in the game center, which wants to find all available rooms that he can join.  
**Actors**: User, System.  
**Pre-conditions**:  
 1. The User is logged-in.  
**Post-conditions**: None.  
**Main success scenario**:

1. The User asks to find all active games he can join.
2. The system calculates the available rooms according to the user level, amount of money and according to the rooms that are not full.
3. The system displays all the available rooms.

**Alternatives**:   
 3.a. The system displays to the user that there are no available rooms for him.



**Use case name**: List all active games that are available for spectating.

**Description**: This scenario describes the process of user asking to see all games that are available for spectating, and those game presented to him.

**Actors**: User, System.

**Pre-conditions**:

1. The user is logged in.

**Post-conditions**:

None.

**Main success scenario**:

1. The user asks to see all games that are available for spectating.
2. The system calculate all relevant games.
3. The system displays all relevant games for spectating to the user.



**Use case name**: Playing Texas Hold'em game.

**Description**: This scenario describes the flow of playing one round in the game.

**Actors**: System, Users in existing game.

**Pre-conditions**:

1. There are enough players to start the game.

**Post-conditions**:

1. Users balance has been updated accordingly.

2. The game log is updated with the info that relevant to this round.

**Main success scenario**:

1. Each user gets two cards from the deck.
2. Players at positions small and big blind put their chips automatically.
3. Each player plays his turn. (use case "play turn")
4. System displays 3 cards from the deck.
5. Each player plays his turn. (use case "play turn")
6. System displays the "turn" card.
7. Each player plays his turn. (use case "play turn")
8. System displays the "river" card.
9. Each player plays his turn. (use case "play turn")
10. System Calculates winner.
11. System Updates users' balance according to their winnings.
12. The system presents relevant massage to the users.
13. The system updates the game log.
14. New round is starting, back to step 1.

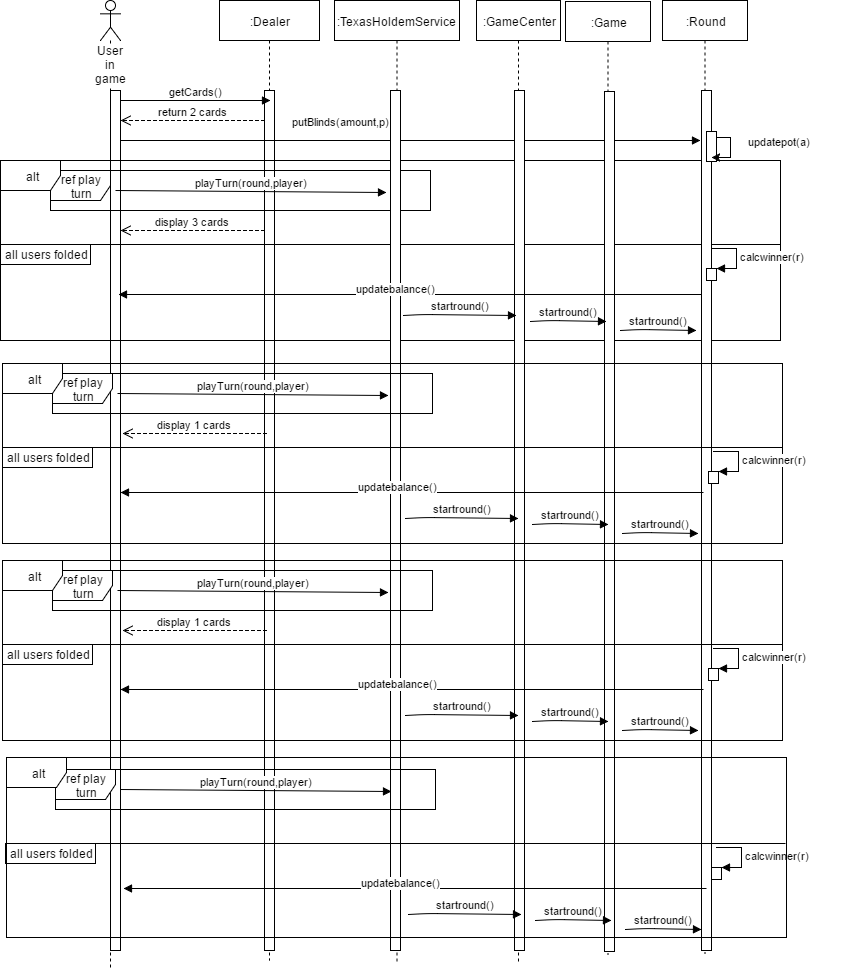
**Alternatives**:   
 4.a. all players except one have folded. Move to step 10.

6.a. all players except one have folded. Move to step 10.

8.a. all players except one have folded. Move to step 10.

Alternative for all steps: player exits the game. (Look at use case "player exits

game").



**Use case name**: Play turn.

**Description**: This scenario describes the flow of a player playing a game.

**Actors**: System, User in existing game.

**Pre-conditions**:

* + - 1. Player is in a running game.
      2. This is the player turn.

**Post-conditions**:

1. The game log is updated with the info that relevant to this turn.

**Main success scenario**:

1. The system displays the player the relevant actions he can choose.
2. The user chooses from the following actions:
   1. Check. (use case "player checks")
   2. Fold. (use case "player folds")
   3. Raise. (use case "player raise")
   4. Call. (use case "player call")
3. The system displays message about the player action.
4. The system updates the game log.



**Use case name**: player checks.

**Description**: This scenario describes a player choosing the check action.

**Actors**: System, User in existing game.

**Pre-conditions**:

1. Player is in a running game.
2. This is the player turn.
3. All players who played before did not raise.

**Post-conditions**:

1. The pot balance remains the same.

**Main success scenario**:

1. The player chooses the "check" action.
2. The system updates the other user that this player has checked.

C:\Users\אחיעד\Downloads\playerChecks.png

**Use case name**: player raise.

**Description**: This scenario describes a player choosing the raise action.

**Actors**: System, User in existing game.

**Pre-conditions**:

1. Player is in a running game.
2. This is the player turn.

**Post-conditions**:

1. The pot balance increased by the amount the player chooses.

2. The player balance is decreased by the amount the player chose.

**Main success scenario**:

1. The player chooses the "raise" action.
2. The system displays the user the minimum amount to raise.
3. The player chooses the amount to raise.
4. The system updates the pot balance accordingly.
5. The system updates the player balance accordingly.
6. The system updates the other users that this player has raised.



**Use case name**: player folds.

**Description**: This scenario describes a player choosing the fold action.

**Actors**: System, User in existing game.

**Pre-conditions**:

1. Player is in a running game.
2. This is the player turn.

**Post-conditions**:

1. The number of active players in the round is decreased by 1.

**Main success scenario**:

1. The player chooses the "fold" action.
2. The system decreases the number of active players in the round by one.
3. The system updates the other users that this player fold.



**Use case name**: player calls.

**Description**: This scenario describes a player choosing the call action.

**Actors**: System, User in existing game.

**Pre-conditions**:

1. Player is in a running game.
2. This is the player turn.
3. At least one player before raised.

**Post-conditions**:

1. The pot balance increased by the amount the player calls.

2. The player balance is decreased by the amount the player chose.

**Main success scenario**:

1. The player chooses the "call" action.
2. The system updates the pot balance accordingly.
3. The system updates the player balance accordingly.
4. The system updates the other users that this player has called.



**Use case name**: Search/filter active games

**Description**: This scenario describes the flow of searching active games in the system.

**Actors**: System, User.

**Pre-conditions**:

1. The User is logged-In.

**Post-conditions**:

1. The user receives all the active games he looked for according to the searching

Criteria.

**Main success scenario**:

1. The user choose criteria for searching an active games from those:

Search by player name/ Search by pot size/ Search by game type policy/ Search by buy in/ Search by chip policy/ Search by minimum bet/ Search by minimal amount of players/ Search by maximal amount of players/Search by spectating criteria.

1. The user search active games by this criteria.
2. The system presents relevant massage to the users, include list of the games that founded in the search.

**Alternatives**:   
 3.a. list of games is empty. Move to step 1.

