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

**2. Good:**

**2.1** The user chooses to join a game that matches his league and that isn't full and the system seats him down at the table.

**3. Sad:**

**3.1** The user tries to join a game that is full. The system will alert him of the problem and would keep him as a spectator.

**3.2** The user tries to join a game that sets a minimum buy-in that is too high for the player. The system will alert him of the problem and would keep him as a spectator.

**4. Bad:**

**4.1** The user types an illegal character in the buy-in field (anything that's not a number). The system will alert him of the problem and tell him to re-enter a buy-in.

**4.1** The user types a value that doesn’t match the room's settings in the buy-in field (number too high or too low). The system will alert him of the problem and tell him to re-enter a buy-in.

**1. Use case:** Register new user

**2. Good:**

**2.1** A guest user wants to sign up. Enters valid email, valid and unused username and a valid password twice. The system adds his information to the database and performs log in immediately.

**3. Sad:**

**3.1** A guest user enters a non-valid value in one of the fields. The system should alert him of the wrong field and let him type again and perform nothing else.

**3.2** A guest user enters a username that already exists in the database. The system should alert him that the username is taken and let him choose a different one and perform nothing else.

**4. Bad:**

**4.1** The user leaves one or more fields empty. The system should alert him that he must fill those fields and perform nothing else.

**1. Use case:** Spectate active game

**2. Good:**

**2.1** The user chooses to spectate a game out of the list of games he can spectate and starts watching the game. The system adds him to the list of spectator, so he will be able to see the messages on the room's chat.

**3. Sad:**

**3.1** The user tries to spectate a game that is not on the list. The system should not let him enter the room and alert him of the problem (that he can't spectate this game)

**4. Bad:**

**4.1** The user chooses to spectate a game from the list, but the game ended between the moment the user chose the game and the moment the game was loaded. The system should alert him that the game is over and suggest to watch a replay.

**1. Use case:** Logging a registered user

**2. Good:**

**2.1** The user wishes to login to his account. He enters his username and his password, the system validates and then redirects him to the homepage, showing his stats and games he can join.

**3. Sad:**

**3.1** The user accidentally entered an incorrect value in one of the fields (password doesn't much username, username doesn't exist). The system should send a standard message (e.g "invalid username/password"), keep him in the login screen and let him try again.

**4. Bad:**

**4.1** The user leaves one of the fields empty. the system should alert him that all fields are mandatory, keep him in the login screen and let him try again.

**1. Use case:** Edit user profile.

**2. Good:**

**2.1** A logged in user wishes to edit **his own** information. Enters valid email, valid and unused username and a valid password twice. The system updates his information to the database and performs log in immediately.

**3. Sad:**

**3.1** A logged in user enters a non-valid value in one of the fields. The system should alert him of the wrong field and let him type again and perform nothing else.

**3.2** A logged in user enters a username that already exists in the database. The system should alert him that the username is taken and let him choose a different one and perform nothing else.

**4. Bad:**

**4.1** The user leaves one or more fields empty. The system should alert him that he must fill those fields and perform nothing else.

**4.2** very bad scenario- A non admin user tries to edit a different user's information. The system should recognize that and let the user know that he doesn't have permission to perform this action.

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

**2. Good:**

**2.1** The user chooses to watch the replay of a game that has already ended. The system displays the game log of that game on the user's screen.

**3. Sad:**

**3.1** The user tries to watch the replay of a game that is still active. The system should alert him of that and suggest to join the room as a spectator.

**4. Bad:**

**4.1** The user tries to watch the replay of a game that doesn't exist. The system should alert him that there is no such game and leave at the same spot.

**1. Use case:** Display all games user can join/spectate

**2. Good:**

**2.1** The user requests to see a list of all the active games he can spectate or join. The system should search through the active games and display them to the user.

**3. Sad:**

**3.1** The user requests to see a list of all the active games he can spectate or join, but there aren't any. The system will go through the active games, to see there are none he can join, and display a related message.

**4. Bad:**

**4.1** The user requests to see a list of all the active games he can spectate or join. The system searches through the active games but during the search something went wrong with the connection to the server. The system should alert the user that the connection was lost and ask him to retry.

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

**2. Good:**

**2.1** The user is currently playing a game and requests to leave. The system lets him out of the room but make the other players think he's still there. The system will fold him if he haven't yet and will free his chair once the current hand ends.

**3. Sad:**

**3.1** The user somehow sent a leave game request, without being an active player in that game. The system should display an appropriate message and keep him in the same spot.

**4. Bad:**

**4.1** The user a leave game request in the name of a different user. The system should recognize that the user who sent the request isn't the one mentioned inside the request, thus display a message that he doesn't have permission to perform this action and leave him in the same spot.

**1. Use case:** Filter games by player name

**2. Good:**

**2.1** The user enters the name of a user (could be a different user). The system validates that the user exists and returns all of his games.

**2.1** The user leaves the name field empty. The system should display all games of all users.

**3. Sad:**

**3.1** The user enters a name of a user that doesn't have any games. The system will not be able to find games and will display a message (i.e "couldn't find games for user") and keep the current state of the window.

**4.1** The user enters a name of a user that doesn't exist. The system should recognize that and display a message (i.e "user doesn't exist") and keep the current state of the window.

**1. Use case:** Filter games by pot size

**2. Good:**

**2.1** The user enters the name of a user (could be a different user). The system validates that the user exists and returns all of his games.

**2.1** The user leaves the name field empty. The system should display all games of all users.

**3. Sad:**

**3.1** The user enters a name of a user that doesn't have any games. The system will not be able to find games and will display a message (i.e "couldn't find games for user") and keep the current state of the window.

**4.1** The user enters a name of a user that doesn't exist. The system should recognize that and display a message (i.e "user doesn't exist") and keep the current state of the window.

**1. Use case:** Filter games by preference

**2. Good:**

**2.1** The user enters the name of a user (could be a different user). The system validates that the user exists and returns all of his games.

**2.1** The user leaves the name field empty. The system should display all games of all users.

**3. Sad:**

**3.1** The user enters a name of a user that doesn't have any games. The system will not be able to find games and will display a message (i.e "couldn't find games for user") and keep the current state of the window.

**4.1** The user enters a name of a user that doesn't exist. The system should recognize that and display a message (i.e "user doesn't exist") and keep the current state of the window.

**1. Use case:** Call

**2. Good:**

**2.1** A player decides to call the current highest bet. The system raises his bet current bet to even them up passes the turn to the next player.

**3. Sad:**

**3.1** A player tries to call when the current highest bet is 0. The system will alert him there's no bet to call to and that he can check instead.

**4. Bad:**

**4.1** A user tries to call in the name of a different user. The system should recognize that the user who sent the request isn't the one mentioned inside the request, thus display a message that he doesn't have permission to perform this action and leave the table's state as is.

**1. Use case:** Check

**2. Good:**

**2.1** A player decides to check when the current highest bet is 0. The system keeps his current bet and passes the turn to the next player.

**3. Sad:**

**3.1** A player tries to check when the current highest bet is higher than 0. The system will alert him there's a bet to call to and that he can call or raise to higher bet instead.

**4. Bad:**

**4.1** A user tries to check in the name of a different user. The system should recognize that the user who sent the request isn't the one mentioned inside the request, thus display a message that he doesn't have permission to perform this action and leave the table's state as is.

**1. Use case:** Raise

**2. Good:**

**2.1** A player decides to raise an amount that is lower or equals to the money he has. The system raises his current bet to the amount he entered and passes the turn to the next player.

**3. Sad:**

**3.1** A player tries to raise when he doesn't have enough money (i.e tries to raise 200 when he has 100). The system will alert him that he doesn’t have enough money and suggest to choose a different amount.

**4. Bad:**

**4.1** A user tries to raise in the name of a different user. The system should recognize that the user who sent the request isn't the one mentioned inside the request, thus display a message that he doesn't have permission to perform this action and leave the table's state as is.

**4.2** A user tries to raise a negative amount, and in that way steal money from the system. The system should recognize that the amount mentioned inside the request is negative, display a message that he doesn't have permission to perform this action (raise must be positive) and leave the table's state as is.

**1. Use case:** Fold

**2. Good:**

**2.1** A player decides to fold. The system keeps his current bet, takes his card back to the dealer and passes the turn to the next player. The system will pass on this player's turn for the rest of the ongoing hand.

**3. Sad:**

**3.1** A player tries to fold not during a hand (somehow sent a fold request in between hands). The system should recognize that and offer him to leave instead (if he wanted to fold maybe he meant that he wants to leave).

**4. Bad:**

**4.1** A user tries to Fold in the name of a different user. The system should recognize that the user who sent the request isn't the one mentioned inside the request, thus display a message that he doesn't have permission to perform this action and leave the table's state as is.