Online Multiplayer Board Game Platform

Use Case Descriptions - Iteration 1

Date: 2025-11-01

Use Case 1: Register Account

Goal: Create a new user account to access OMG.
Primary Actor(s): Guest
Preconditions: User is not authenticated.
Postconditions (Success guarantees): New account created; user can log in.

Main Success Scenario:

1. Guest opens the app and chooses “Sign Up”.
2. System collects email, password, and display name.
3. System validates input (format, password strength).
4. System calls AuthService stub to create an account (DB mocked). 5 System confirms account creation.

Alternate/Exceptions:

(1-3 a): Invalid input: show inline errors; user retries.
(4a): AuthService stub returns failure: show generic error; suggest retry.

Notes: DB calls are stubbed; persistence simulated.

Use Case 2: Login

Goal: Authenticate an existing user.
Primary Actor(s): Registered User
Preconditions: User has a registered account (real DB replaced by stub).
Postconditions (Success guarantees): User session is established; home/lobby displayed.

Main Success Scenario:

1. User opens “Login”.
2. System collects credentials.
3. System calls AuthService stub to verify.
4. System starts a local session and navigates to Lobby.

Alternate/Exceptions:

(3a): Wrong credentials: show error; allow retry.

(3b): Service unavailable: offline mode; limited access.

Use Case 3: View/Update Profile

Goal: View and edit profile fields (avatar, bio).
Primary Actor(s): Registered User
Preconditions: User is authenticated.
Postconditions (success guarantees): Profile updated locally; queued to ProfileService stub.

Main Success Scenario:

1. User opens Profile page.
2. System displays current profile & stats (from local model).
3. User edits fields and saves.
4. System validates and updates the local model and calls the ProfileService stub.

Alternate/Exceptions:

(4a): Stub failure: keep local change; mark as ‘Pending Sync’.

Use Case 4: Queue for Match (Matchmaking)

Goal: Enter the matchmaking queue and be paired with an opponent of a similar rating.
Primary Actor(s): Player
Preconditions: User is authenticated and not in a match.
Postconditions (Success guarantees): Match is created; players moved into a GameSession.

Main Success Scenario:

1. Player selects a game (e.g., Chess).
2. Player clicks “Play / Find Match”.
3. System adds player to Queue and queries RatingService (ELO).
4. MatchmakingService pairs players based on rating, wait time, and region (simulated).
5. System creates GameSession and notifies both players.

Alternate/Exceptions:

(4a): No suitable opponent: keep in queue; show ETA.

(4b): Player cancels: remove from queue, return to Lobby.

Use Case 5: Create Lobby

Goal: Create a lobby with options to invite a friend
Primary Actor(s): Player
Preconditions: Game is selected
Postconditions (Success guarantees): Lobby created successfully; Lobby ID displayed; ready to invite friend.

Main Success Scenario:

1. Player selects “Create Lobby”.
2. System collects options and calls LobbyService (stub) to create a lobby and returns Lobby ID.
3. System displays Lobby ID and “Invite Friend”.

Alternate/Exceptions:

(2a): Lobby not found/full: show error.

(3a): Stub failure; allow retry

Use Case 6: Join Lobby

Goal: Join a friend’s lobby or accept an invite.
Primary Actor(s): Player
Preconditions: User is authenticated; lobby exists.
Postconditions (Success guarantees): Player joins the lobby and is ready to start

Main Success Scenario:

1. Player opens Friends or Lobby ID dialogue.
2. System locates lobby (LobbyService stub).
3. System adds player; lobby view updates.

Alternate/Exceptions:

(2a): Lobby not found/full: show error.

(3a): Stub failure; allow retry

Use Case 7: Player’s Turn

Goal: Make a legal move on the board during an active session.
Primary Actor(s): Player
Preconditions:

- Active gameSession exists; it's the players turn.

Postconditions (Success guarantees):

- Move is validated and applied to the current gameState
- Board/UI is re-rendered to show the new position/state.
- Turn passes to the opponent.
- Move is logged (local MatchLog).
- Opponent is notified (via Network/MatchService stub).

Main Success Scenario:

1. Player selects/makes a move on the board (e.g., drag piece, click tile, select card). System sends the move to the GameRules engine for that game type (chess, checkers, etc.).
2. System verifies that:
    - Its the player's turn
    - The move is legal under the current board state
    - The piece/card/tile belongs to this player (if applicable)
3. If valid, System updates the GameState (board, active player, timers).
4. System re-renders the board to reflect the new state.
5. System appends the move to the local MatchLog (for replay/history).
6. System notifies opponent via Network stub (simulated send).
7. System checks for end-of-game conditions (win, draw, timeout) and acts accordingly.

Alternate/Exceptions:

(2a): Illegal move: highlight rule violation.

- The system highlights the error
- System does not update GameSate
- Player may try again

(4a): Network stub failure: retry/queue; show reconnecting.

- System queues the move locally
- Show status (ie. “reconnecting”)
- Keep local state consistent

Use Case 8: View Leaderboard

Goal: Allow a user to see top players and, if logged in, their own rank
Primary Actor(s): Player / Guest
Preconditions:

- App is running
- Mock data is available, or a cached sample is available

Postconditions (Success guarantees):

- Leaderboard list is displayed, showing top N players
- If the user is logged in, their position is shown/highlighted
- Filters/paging are applied locally or via stub

Main Success Scenario:

1. User opens the Leaderboard tab from the main UI
2. System fetches ranks from LeaderboardService stub.
3. System shows the top N players (rank, name, points, wins)
    a. If logged in, the user's rankings
4. USer can scroll or switch filters (i.e., chess, go, tictac, all games)

Alternate/Exceptions:

(2a): Stub down/no response

- show cached sample
- display banner 'Offline data'.

(2b): User not logged in

- System still shows top N
- Instead of "your rank", display will say "log in to see your rank"

Use Case 9: In-Game Chat

Goal: Send/receive short text messages during a match.
Primary Actor(s): Player
Preconditions: Active GameSession exists.
Postconditions (Success guarantees): Message appended to ChatLog and displayed to both players.

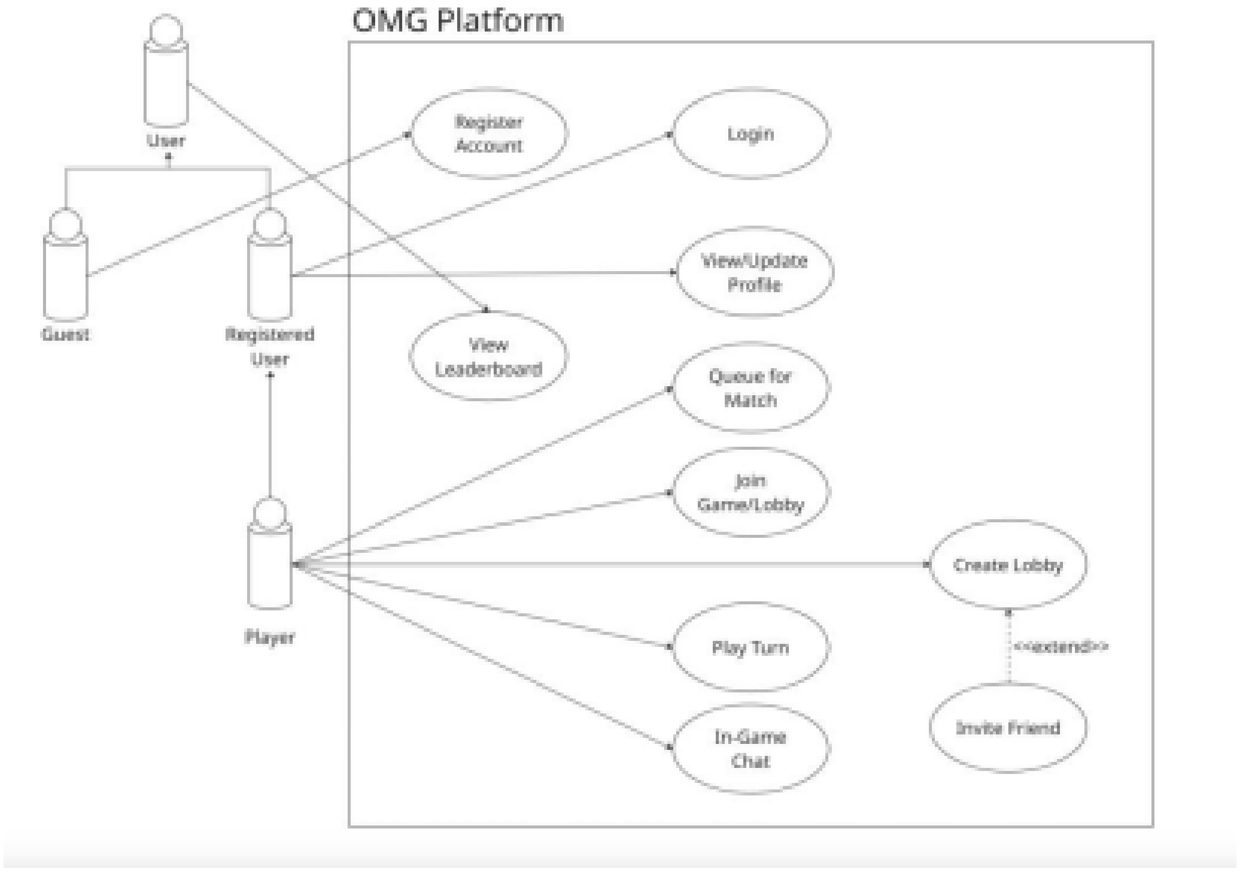
Main Success Scenario:

1. Player types a message and hits Send.
2. System sanitizes and timestamps the message.
3. System sends via ChatService stub to the opponent.
4. Messages appear in the chat pane.

Alternate/Exceptions:

(3a): Stub failure: show 'not delivered'; allow resend.

USE-CASE DIAGRAM


OMG Platform
User
Guest
Registered User
Player
Register Account
Login
View/Update Profile
View Leaderboard
Queue for Match
Join Game/Lobby
Create Lobby
Play Turn
In-Game Chat
Invite Friend
<<extend>>