MemeBattle Card Game Server - Complete API Documentation

Version: v1
Base URL: `http://localhost:3000/api/v1`
WebSocket: `ws://localhost:3000`
Date: November 25, 2025

Quick Reference

REST API Endpoints (51 total)

Authentication (9 endpoints)

- POST /auth/register - Register new user
- POST /auth/login - Login
- POST /auth/refresh - Refresh token
- POST /auth/verify-email - Verify email
- GET /auth/verify-email?token=xxx - Verify via link
- POST /auth/resend-verification - Resend email
- POST /auth/logout - Logout
- GET /auth/me - Get profile
- GET /auth/status - Check auth status

Deck Management (7 endpoints)

- POST /decks - Create deck
- GET /decks - Get all decks
- GET /decks/active - Get active deck
- GET /decks/:deckId - Get specific deck
- PUT /decks/:deckId - Update deck
- PATCH /decks/:deckId/activate - Set as active
- DELETE /decks/:deckId - Delete deck

Inventory (4 endpoints)

- GET /inventory - Get inventory
- POST /inventory/cards - Add card
- DELETE /inventory/cards/:cardId - Remove card
- POST /inventory/characters - Add character

Users (1 endpoint)

- GET /users/search?username=query - Search users

Friends (7 endpoints)

- GET /friends - Get friends
- POST /friends/requests - Send request
- GET /friends/requests/sent - Sent requests
- GET /friends/requests/pending - Pending requests
- POST /friends/requests/:requestId/accept - Accept
- POST /friends/requests/:requestId/decline - Decline
- DELETE /friends/:friendId - Remove friend

Lobbies (12 endpoints)

- POST /lobbies - Create lobby
- GET /lobbies/public - Get public lobbies
- GET /lobbies/me/current - Get current lobby

- GET /lobbies/:lobbyId - Get lobby
- POST /lobbies/:lobbyId/join - Join lobby
- POST /lobbies/:lobbyId/leave - Leave lobby
- PUT /lobbies/:lobbyId/deck - Select deck
- PUT /lobbies/:lobbyId/character - Select character
- PUT /lobbies/:lobbyId/settings - Update settings
- POST /lobbies/:lobbyId/kick - Kick player
- POST /lobbies/:lobbyId/start - Start game
- DELETE /lobbies/:lobbyId - Cancel lobby

Games (8 endpoints)

- GET /games/:gameId - Get game
- GET /games/me/history - Game history
- GET /games/me/recent - Recent games
- GET /games/me/stats - Statistics
- GET /games/leaderboard - Leaderboard
- GET /games/head-to-head/:opponentId - H2H record
- GET /games/active/:gameId - Active game state
- POST /games/:gameId/forfeit - Forfeit game

Gacha (3 endpoints)

- POST /gacha/pull/single - Pull 1 card
- POST /gacha/pull/multi - Pull 10 cards
- GET /gacha/info - Gacha info

WebSocket Events

Lobby Events (Client → Server)

- lobby:joined - Join lobby room
- lobby:leave - Leave lobby
- lobby:update:settings - Update settings
- lobby:select:deck - Select deck
- lobby:select:character - Select character
- lobby:kick:player - Kick player
- lobby:start:game - Start game
- lobby:ready:toggle - Toggle ready

Lobby Events (Server → Client)

- lobby:reconnected - Reconnected
- lobby:state:update - State update
- lobby:left - Left confirmation
- lobby:kicked - Kicked notification
- lobby:closed - Lobby closed
- game:started - Game started
- error - Error message

Game Events (Client → Server)

- game:join - Join game
- game:dice_roll:submit - Roll dice
- game:action:hover - Preview card
- game:action:play_card - Play card
- game:action:skip_turn - Skip turn
- game:leave - Leave game

Game Events (Server → Client)

- game:load - Initial state

- game:state:update - State update
- game:dice_roll:start - Start rolling
- game:dice_roll:wait - Wait for opponent
- game:dice_roll:result - Roll results
- game:action:preview - Card preview
- game:end - Game ended
- game:error - Game error

Lobby List Broadcasts

- lobbyList:lobby:created - Lobby created
- lobbyList:lobby:updated - Lobby updated
- lobbyList:lobby:deleted - Lobby deleted
- lobbyList:lobby:started - Lobby started

Socket.IO Architecture

Connection Flow

```
Client → Connect with JWT token
  ↓
Server → Authenticate via middleware
  ↓
Success → Attach user info to socket
  ↓
Register → Lobby & Game event handlers
  ↓
Auto-join → Active lobby (if exists)
```

Room System

- Each lobby = one room (lobbyId)
- Each game = one room (gameId)
- Broadcasts only within rooms

Data Storage

- **MongoDB:** Lobbies, Users, Decks, Completed Games
- **Redis:** Active game states (fast access)

Reconnection

- 2-minute grace period
- Auto-rejoin active lobby
- Game state preserved

Authentication

Token Flow

1. Login → Access token (cookie, 15min) + Refresh token (response body)
2. Access token expires → Use refresh token to get new pair
3. Store refresh token securely on client

Headers

```
Cookie: accessToken=<token>
```

or

```
Authorization: Bearer <token>
```

Complete Documentation

See full API_DOCUMENTATION.md for:

- Detailed request/response examples
- Validation rules
- Error codes
- Data models
- Testing examples