MemeBattle Card Game Server - Complete API Documentation

Version: 1.0.0 **Last Updated:** 2025-11-25 **Base URL:** `http://localhost:3000/api/v1`
WebSocket URL: `ws://localhost:3000`

Table of Contents

Overview

The MemeBattle Card Game Server provides a comprehensive REST API and WebSocket connection for managing a multiplayer card game. The system includes:

- **51 REST API Endpoints** across 8 major modules
- **WebSocket Events** for real-time lobby and game interactions
- **JWT-based Authentication** with refresh tokens
- **MongoDB** for data persistence
- **Redis** for active game state management

Server Configuration

```
{
  "baseUrl": "http://localhost:3000/api/v1",
  "websocket": "ws://localhost:3000",
  "port": 3000,
  "environment": "development"
}
```

Authentication

Authentication Flow

The server uses **JWT (JSON Web Tokens)** with a dual-token system:

1. **Access Token**
   - Stored in HTTP-only cookie
   - 15 minutes expiration
   - Used for all authenticated requests
2. **Refresh Token**
   - Returned in response body (client stores)
   - 7 days expiration
   - Used to obtain new access tokens

How to Authenticate

Option 1: Cookie-based (Recommended)

```
Cookie: accessToken=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Option 2: Bearer Token

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

WebSocket Authentication

```
const socket = io('ws://localhost:3000', {
  auth: {
    token: 'your_jwt_access_token'
  }
});
```

REST API Endpoints

1. AUTHENTICATION ENDPOINTS

1.1 Register New User

Creates a new user account and sends verification email.

Endpoint: `POST /api/v1/auth/register` **Access:** Public **Content-Type:** `application/json`

Request Body:

```
{
  "email": "user@example.com",
  "username": "username123",
  "password": "Password123!",
  "displayName": "My Display Name"
}
```

Validation Rules:

- `email` : Valid email format, required
- `username` : 3-20 alphanumeric characters, required
- `password` : 8-128 characters, must contain uppercase, lowercase, number, and special character, required
- `displayName` : 2-30 characters, required

Success Response (201 Created):

```
{
  "success": true,
  "message": "Registration successful! Please check your email to verify your
account before logging in.",
  "data": {
    "user": {
      "uid": "u_1234567890abcdef",
      "username": "username123",
      "email": "user@example.com",
      "displayName": "My Display Name",
      "profilePic": "https://example.com/default-avatar.png",
      "isEmailVerified": false,
      "isOnline": false,
      "stats": {
        "winRate": 0,
        "totalGames": 0,
        "wins": 0,
        "losses": 0
      },
      "createdAt": "2025-11-25T10:00:00.000Z"
    }
  }
}
```


Error Responses:


```
{
  "success": false,
  "message": "Validation error",
  "errors": [
    {
      "field": "email",
      "message": "Email already registered"
    }
  ]
}
```

1.2 Login

Authenticates a user and returns tokens.

Endpoint: `POST /api/v1/auth/login` **Access:** Public **Content-Type:** `application/json`

Request Body (Option 1 - Email):

```
{
  "email": "user@example.com",
  "password": "Password123!"
}
```

Request Body (Option 2 - Username):

```
{
  "username": "username123",
  "password": "Password123!"
}
```

Validation Rules:

- Must provide either `email` OR `username` (not both)
- `password` : Required

Success Response (200 OK):

```
{
  "success": true,
  "message": "Login successful!",
  "data": {
    "refreshToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJ1XzEyMzQ1Njc4OTBhYmNkZWYiLCI
  }
}
```

Side Effect: Sets HTTP-only cookie `accessToken` with 15-minute expiration

Error Responses:

```
{
  "success": false,
```

```
  "message": "Invalid credentials"
}
```

```
{
  "success": false,
  "message": "Please verify your email before logging in"
}
```

1.3 Refresh Access Token

Obtains a new access token using refresh token.

Endpoint: `POST /api/v1/auth/refresh` **Access:** Public **Content-Type:** `application/json`

Request Body:

```
{
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Success Response (200 OK):

```
{
  "success": true,
  "message": "Token refreshed successfully",
  "data": {
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
  }
}
```

Side Effect: Sets new HTTP-only cookie `accessToken` with 15-minute expiration

---

1.4 Verify Email (POST)

Verifies user email with token.

Endpoint: `POST /api/v1/auth/verify-email` **Access:** Public **Content-Type:** `application/json`

Request Body:

```
{
  "token": "verification_token_from_email"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "message": "Email verified successfully! You can now login to access all features.",
  "data": {
    "user": {
      "uid": "u_1234567890abcdef",
      "username": "username123",
      "email": "user@example.com",
      "displayName": "My Display Name",
      "isEmailVerified": true,
      "isOnline": false,
      "stats": {
        "winRate": 0,
        "totalGames": 0,
        "wins": 0,
        "losses": 0
      },
      "createdAt": "2025-11-25T10:00:00.000Z"
    }
  }
}
```

1.5 Verify Email (GET)

Email verification via link (opens in browser).

Endpoint: `GET /api/v1/auth/verify-email?token=xxx` **Access:** Public

Query Parameters:

- `token` : Verification token (required)

Success Response: HTML page confirming verification **Error Response:** HTML page with error message

---

1.6 Resend Verification Email

Resends email verification to user.

Endpoint: `POST /api/v1/auth/resend-verification` **Access:** Public **Content-Type:** `application/json`

Request Body:

```
{
  "email": "user@example.com"
}
```

Success Response (200 OK):

```
{
  "success": true,
  "message": "Verification email sent! Please check your inbox.",
  "data": null
}
```

1.7 Logout

Logs out the current user.

Endpoint: `POST /api/v1/auth/logout` **Access:** Private (requires authentication)

Request Body: None

Success Response (200 OK):

```
{
  "success": true,
  "message": "Logout successful!",
```

```
  "data": null
}
```

Side Effect: Clears `accessToken` cookie

1.8 Get Current User Profile

Retrieves authenticated user's profile.

Endpoint: `GET /api/v1/auth/me` **Access:** Private (requires authentication)

Success Response (200 OK):

```
{
  "success": true,
  "message": "User profile retrieved successfully",
  "data": {
    "user": {
      "uid": "u_1234567890abcdef",
      "username": "username123",
      "email": "user@example.com",
      "displayName": "My Display Name",
      "profilePic": "https://example.com/profile.jpg",
      "isEmailVerified": true,
      "isOnline": true,
      "stats": {
        "winRate": 65.5,
        "totalGames": 20,
        "wins": 13,
        "losses": 7
      },
      "createdAt": "2025-11-25T10:00:00.000Z"
    }
  }
}
```

1.9 Check Authentication Status

Checks if user is authenticated (with optional authentication).

Endpoint: `GET /api/v1/auth/status` **Access:** Public (optional authentication)

Success Response (200 OK) - Authenticated:

```
{
  "success": true,
  "message": "User is authenticated and verified",
  "data": {
    "isAuthenticated": true,
    "isVerified": true,
    "user": {
      "uid": "u_1234567890abcdef",
      "username": "username123",
      "displayName": "My Display Name",
      "profilePic": "https://example.com/profile.jpg",
      "isEmailVerified": true,
      "isOnline": true,
      "stats": {
        "winRate": 65.5,
        "totalGames": 20,
        "wins": 13,
        "losses": 7
      }
    }
  }
}
```

Success Response (200 OK) - Not Authenticated:

```
{
  "success": true,
  "message": "User is not authenticated",
  "data": {
    "isAuthenticated": false,
    "isVerified": false,
    "user": null
  }
}
```

---

2. DECK MANAGEMENT ENDPOINTS

All deck endpoints require authentication.

2.1 Create Deck

Creates a new deck for the user.

Endpoint: `POST /api/v1/decks` **Access:** Private **Content-Type:** `application/json`

Request Body:

```
{
  "deckTitle": "My Awesome Deck",
  "cards": [
    {
      "cardId": "507f1f77bcf86cd799439011",
      "position": 0
    },
    {
      "cardId": "507f1f77bcf86cd799439012",
      "position": 1
    }
    // ... minimum 15 cards, maximum 30 cards
  ],
  "isActive": false
}
```

Validation Rules:

- `deckTitle` : 1-50 characters, required
- `cards` : Array of 15-30 cards, required, duplicates allowed
- `cards[].cardId` : Valid MongoDB ObjectId, required
- `cards[].position` : 0-29, optional (auto-assigned if not provided)
- `isActive` : Boolean, optional, default: false

Success Response (201 Created):

```
{
  "success": true,
  "message": "Deck created successfully",
  "data": {
    "deck": {
      "deckId": "507f1f77bcf86cd799439020",
      "deckTitle": "My Awesome Deck",
      "userId": "u_1234567890abcdef",
      "isActive": false,
```

```
      "cardCount": 20,
      "cards": [
        {
          "cardId": "507f1f77bcf86cd799439011",
          "position": 0,
          "name": "Pepe Strike",
          "power": 5,
          "rarity": "epic",
          "cardType": "attack",
          "cardImage": "https://example.com/cards/pepe-strike.png",
          "pawnRequirement": 2
        },
        {
          "cardId": "507f1f77bcf86cd799439012",
          "position": 1,
          "name": "Doge Shield",
          "power": 4,
          "rarity": "rare",
          "cardType": "defense",
          "cardImage": "https://example.com/cards/doge-shield.png",
          "pawnRequirement": 1
        }
      ],
      "createdAt": "2025-11-25T10:00:00.000Z",
      "updatedAt": "2025-11-25T10:00:00.000Z"
    }
  }
}
```

2.2 Get All User Decks

Retrieves all decks belonging to the authenticated user.

Endpoint: `GET /api/v1/decks` **Access:** Private

Success Response (200 OK):


```
{
  "success": true,
  "message": "Decks retrieved successfully",
  "data": {
    "decks": [
      {
        "deckId": "507f1f77bcf86cd799439020",
        "deckTitle": "My Awesome Deck",
```

```
        "isActive": true,
        "cardCount": 20,
        "createdAt": "2025-11-25T10:00:00.000Z"
      },
      {
        "deckId": "507f1f77bcf86cd799439021",
        "deckTitle": "Another Deck",
        "isActive": false,
        "cardCount": 25,
        "createdAt": "2025-11-24T15:30:00.000Z"
      }
    ],
    "count": 2
  }
}
```

2.3 Get Active Deck

Retrieves the user's currently active deck.

Endpoint: `GET /api/v1/decks/active` **Access:** Private

Note: This route must be defined before `/:deckId` to avoid route conflicts.

Success Response (200 OK):


```
{
  "success": true,
  "message": "Active deck retrieved successfully",
  "data": {
    "deck": {
      "deckId": "507f1f77bcf86cd799439020",
      "deckTitle": "My Active Deck",
      "userId": "u_1234567890abcdef",
      "isActive": true,
      "cardCount": 20,
      "cards": [
        {
          "cardId": "507f1f77bcf86cd799439011",
          "position": 0,
          "name": "Pepe Strike",
          "power": 5,
          "rarity": "epic",
          "cardType": "attack",
```

```
          "cardImage": "https://example.com/cards/pepe-strike.png",
          "pawnRequirement": 2
        }
      ],
      "createdAt": "2025-11-25T10:00:00.000Z",
      "updatedAt": "2025-11-25T10:00:00.000Z"
    }
  }
}
```


Error Response (404 Not Found):


```
{
  "success": false,
  "message": "No active deck found"
}
```

2.4 Get Specific Deck by ID

Retrieves a specific deck by its ID.

Endpoint: `GET /api/v1/decks/:deckId` **Access:** Private (must own the deck)

URL Parameters:

- `deckId` : MongoDB ObjectId (required)

Success Response (200 OK):


```
{
  "success": true,
  "message": "Deck retrieved successfully",
  "data": {
    "deck": {
      "deckId": "507f1f77bcf86cd799439020",
      "deckTitle": "My Awesome Deck",
      "userId": "u_1234567890abcdef",
      "isActive": false,
      "cardCount": 20,
      "cards": [
        {
          "cardId": "507f1f77bcf86cd799439011",
```

```
          "position": 0,
          "name": "Pepe Strike",
          "power": 5,
          "rarity": "epic",
          "cardType": "attack",
          "cardImage": "https://example.com/cards/pepe-strike.png",
          "pawnRequirement": 2,
          "cardInfo": "A powerful strike attack card",
          "pawnLocations": [
            {"x": 0, "y": 0},
            {"x": 1, "y": 0}
          ],
          "ability": {
            "type": "damage",
            "value": 2,
            "description": "Deals 2 extra damage"
          }
        }
      ],
      "createdAt": "2025-11-25T10:00:00.000Z",
      "updatedAt": "2025-11-25T10:00:00.000Z"
    }
  }
}
```

2.5 Update Deck

Updates an existing deck.

Endpoint: `PUT /api/v1/decks/:deckId` **Access:** Private (must own the deck) **Content-Type:** `application/json`

URL Parameters:

- `deckId` : MongoDB ObjectId (required)

Request Body:


```
{
  "deckTitle": "Updated Deck Name",
  "cards": [
    {
      "cardId": "507f1f77bcf86cd799439011",
      "position": 0
```

```
    }
    // ... 15-30 cards
  ],
  "isActive": true
}
```

Note: All fields are optional, but at least one must be provided.

Success Response (200 OK):

```
{
  "success": true,
  "message": "Deck updated successfully",
  "data": {
    "deck": {
      "deckId": "507f1f77bcf86cd799439020",
      "deckTitle": "Updated Deck Name",
      "userId": "u_1234567890abcdef",
      "isActive": true,
      "cardCount": 20,
      "cards": [...],
      "createdAt": "2025-11-25T10:00:00.000Z",
      "updatedAt": "2025-11-25T10:05:00.000Z"
    }
  }
}
```

2.6 Set Deck as Active

Sets a deck as the user's active deck.

Endpoint: `PATCH /api/v1/decks/:deckId/activate` **Access:** Private (must own the deck)

URL Parameters:

- `deckId` : MongoDB ObjectId (required)

Request Body: None

Note: Automatically deactivates all other user decks.

Success Response (200 OK):

```
{
  "success": true,
  "message": "Deck activated successfully",
  "data": {
    "deck": {
      "deckId": "507f1f77bcf86cd799439020",
      "deckTitle": "My Active Deck",
      "userId": "u_1234567890abcdef",
      "isActive": true,
      "cardCount": 20,
      "cards": [...],
      "createdAt": "2025-11-25T10:00:00.000Z",
      "updatedAt": "2025-11-25T10:10:00.000Z"
    }
  }
}
```

2.7 Delete Deck

Deletes a deck permanently.

Endpoint: `DELETE /api/v1/decks/:deckId` **Access:** Private (must own the deck)

URL Parameters:

- `deckId` : MongoDB ObjectId (required)

Success Response (200 OK):


```
{
  "success": true,
  "message": "Deck deleted successfully",
  "data": {
    "deletedDeckId": "507f1f77bcf86cd799439020",
    "deletedAt": "2025-11-25T10:15:00.000Z"
  }
}
```


---

3. INVENTORY ENDPOINTS

All inventory endpoints require authentication.

3.1 Get User Inventory

Retrieves the authenticated user's inventory.

Endpoint: `GET /api/v1/inventory` **Access:** Private

Success Response (200 OK):


```
{
  "success": true,
  "message": "Success",
  "data": {
    "userId": "u_1234567890abcdef",
    "cards": [
      {
        "cardId": "507f1f77bcf86cd799439011",
        "quantity": 3
      },
      {
        "cardId": "507f1f77bcf86cd799439012",
        "quantity": 1
      },
      {
        "cardId": "507f1f77bcf86cd799439013",
        "quantity": 5
      }
    ],
    "characters": [
      "507f1f77bcf86cd799439020",
      "507f1f77bcf86cd799439021"
    ]
  }
}
```

3.2 Add Card to Inventory

Adds a card (or multiple copies) to user's inventory.

Endpoint: `POST /api/v1/inventory/cards` **Access:** Private **Content-Type:** `application/json`

Request Body:

```
{
  "cardId": "507f1f77bcf86cd799439011",
  "quantity": 2
}
```

Validation Rules:

- `cardId` : Valid MongoDB ObjectId, required
- `quantity` : Integer >= 1, default: 1

Success Response (201 Created):

```
{
  "success": true,
  "message": "Card added successfully",
  "data": {
    "userId": "u_1234567890abcdef",
    "cards": [
      {
        "cardId": "507f1f77bcf86cd799439011",
        "quantity": 5
      }
    ],
    "characters": [...]
  }
}
```

3.3 Remove Card from Inventory

Removes a card (or quantity) from user's inventory.

Endpoint: `DELETE /api/v1/inventory/cards/:cardId` **Access:** Private **Content-Type:** `application/json`

URL Parameters:

- `cardId` : MongoDB ObjectId (required)

Request Body:

```
{
  "quantity": 1
}
```

Validation Rules:

- `quantity` : Integer >= 1, default: 1

Success Response (200 OK):

```
{
  "success": true,
  "message": "Card removed successfully",
  "data": {
    "userId": "u_1234567890abcdef",
    "cards": [
      {
        "cardId": "507f1f77bcf86cd799439011",
        "quantity": 4
      }
    ],
    "characters": [...]
  }
}
```

Note: If quantity reaches 0, the card is removed from inventory entirely.

3.4 Add Character to Inventory

Adds a character to user's inventory.

Endpoint: `POST /api/v1/inventory/characters` **Access:** Private **Content-Type:** `application/json`

Request Body:

```
{
  "characterId": "507f1f77bcf86cd799439020"
}
```

Validation Rules:

- `characterId` : Valid MongoDB ObjectId, required

Success Response (201 Created):

```
{
  "success": true,
  "message": "Character added successfully",
  "data": {
    "userId": "u_1234567890abcdef",
    "cards": [...],
    "characters": [
      "507f1f77bcf86cd799439020",
      "507f1f77bcf86cd799439021",
      "507f1f77bcf86cd799439022"
    ]
  }
}
```

---

4. USER ENDPOINTS

4.1 Search Users

Searches for users by username.

Endpoint: `GET /api/v1/users/search` **Access:** Private

Query Parameters:

- `username` : String, min 2 characters (required)

Example: `GET /api/v1/users/search?username=john`

Success Response (200 OK):

```
{
  "success": true,
  "message": "Users found successfully",
  "data": {
    "users": [
      {
```

```
        "uid": "u_1234567890abcdef",
        "username": "john_player",
        "displayName": "John Player",
        "profilePic": "https://example.com/profiles/john.jpg",
        "isOnline": true
      },
      {
        "uid": "u_0987654321fedcba",
        "username": "johnny_memer",
        "displayName": "Johnny Memer",
        "profilePic": "https://example.com/profiles/johnny.jpg",
        "isOnline": false
      }
    ],
    "count": 2
  }
}
```


---

5. FRIEND SYSTEM ENDPOINTS

All friend endpoints require authentication.

5.1 Get Friend List

Retrieves the authenticated user's friend list.

Endpoint: `GET /api/v1/friends` **Access:** Private

Success Response (200 OK):


```
{
  "success": true,
  "message": "Friends retrieved successfully",
  "data": {
    "friends": [
      {
        "uid": "u_friend1234567890",
        "username": "friend1",
        "displayName": "Friend One",
        "profilePic": "https://example.com/profiles/friend1.jpg",
        "isOnline": true,
        "friendsSince": "2025-11-20T10:00:00.000Z"
      },
      {
```

```
        "uid": "u_friend0987654321",
        "username": "friend2",
        "displayName": "Friend Two",
        "profilePic": "https://example.com/profiles/friend2.jpg",
        "isOnline": false,
        "friendsSince": "2025-11-15T10:00:00.000Z"
      }
    ],
    "count": 2
  }
}
```

5.2 Send Friend Request

Sends a friend request to another user.

Endpoint: `POST /api/v1/friends/requests` **Access:** Private **Content-Type:** `application/json`

Request Body:


```
{
  "toUserId": "u_target1234567890"
}
```


Validation Rules:

- `toUserId` : Valid MongoDB ObjectId, required

Success Response (201 Created):


```
{
  "success": true,
  "message": "Friend request sent successfully",
  "data": {
    "requestId": "507f1f77bcf86cd799439030",
    "fromUserId": "u_1234567890abcdef",
    "toUserId": "u_target1234567890",
    "status": "pending",
    "createdAt": "2025-11-25T10:00:00.000Z"
```

```
  }
}
```

5.3 Get Sent Friend Requests

Retrieves friend requests sent by the authenticated user.

Endpoint: `GET /api/v1/friends/requests/sent` **Access:** Private

Success Response (200 OK):

```
{
  "success": true,
  "message": "Sent requests retrieved successfully",
  "data": {
    "requests": [
      {
        "requestId": "507f1f77bcf86cd799439030",
        "toUser": {
          "uid": "u_target1234567890",
          "username": "recipient",
          "displayName": "Recipient Name",
          "profilePic": "https://example.com/profiles/recipient.jpg"
        },
        "status": "pending",
        "createdAt": "2025-11-25T10:00:00.000Z"
      }
    ],
    "count": 1
  }
}
```

5.4 Get Pending Friend Requests

Retrieves friend requests received by the authenticated user.

Endpoint: `GET /api/v1/friends/requests/pending` **Access:** Private

Success Response (200 OK):

```
{
  "success": true,
  "message": "Pending requests retrieved successfully",
  "data": {
    "requests": [
      {
        "requestId": "507f1f77bcf86cd799439031",
        "fromUser": {
          "uid": "u_sender1234567890",
          "username": "sender",
          "displayName": "Sender Name",
          "profilePic": "https://example.com/profiles/sender.jpg"
        },
        "status": "pending",
        "createdAt": "2025-11-24T15:00:00.000Z"
      }
    ],
    "count": 1
  }
}
```

5.5 Accept Friend Request

Accepts a pending friend request.

Endpoint: `POST /api/v1/friends/requests/:requestId/accept` **Access:** Private

URL Parameters:

- `requestId` : MongoDB ObjectId (required)

Request Body: None

Success Response (200 OK):


```
{
  "success": true,
  "message": "Friend request accepted successfully",
  "data": {
    "friendship": {
      "user1": "u_sender1234567890",
      "user2": "u_1234567890abcdef",
      "createdAt": "2025-11-25T10:30:00.000Z"
```

```
    }
  }
}
```

5.6 Decline Friend Request

Declines a pending friend request.

Endpoint: `POST /api/v1/friends/requests/:requestId/decline` **Access:** Private

URL Parameters:

- `requestId` : MongoDB ObjectId (required)

Request Body: None

Success Response (200 OK):

```
{
  "success": true,
  "message": "Friend request declined successfully",
  "data": {
    "requestId": "507f1f77bcf86cd799439031"
  }
}
```

5.7 Remove Friend

Removes a friend from the user's friend list.

Endpoint: `DELETE /api/v1/friends/:friendId` **Access:** Private

URL Parameters:

- `friendId` : User ID (required)

Success Response (200 OK):

```
{
  "success": true,
  "message": "Friend removed successfully",
  "data": {
```

```
    "removedFriendId": "u_friend1234567890"
  }
}
```

---

6. LOBBY ENDPOINTS

All lobby endpoints require authentication.

6.1 Create Lobby

Creates a new game lobby.

Endpoint: `POST /api/v1/lobbies` **Access:** Private **Content-Type:** `application/json`

Request Body:

```
{
  "lobbyName": "Epic Battle Room",
  "mapId": "507f1f77bcf86cd799439040",
  "isPrivate": false,
  "password": "",
  "gameSettings": {
    "turnTimeLimit": 60,
    "allowSpectators": false
  }
}
```

Validation Rules:

- `lobbyName` : 3-50 characters, required
- `mapId` : Valid MongoDB ObjectId, required
- `isPrivate` : Boolean, default: false
- `password` : 4-20 characters, required if `isPrivate` is true
- `gameSettings.turnTimeLimit` : 30-300 seconds, default: 60
- `gameSettings.allowSpectators` : Boolean, default: false

Success Response (201 Created):

```
{
  "success": true,
```

```
  "message": "Lobby created successfully",
  "data": {
    "_id": "507f1f77bcf86cd799439050",
    "lobbyName": "Epic Battle Room",
    "status": "waiting",
    "isPrivate": false,
    "maxPlayers": 2,
    "playerCount": 1,
    "isFull": false,
    "host": {
      "_id": "u_1234567890abcdef",
      "username": "player1",
      "displayName": "Player One",
      "profilePic": "https://example.com/profiles/player1.jpg"
    },
    "players": [
      {
        "_id": "u_1234567890abcdef",
        "username": "player1",
        "displayName": "Player One",
        "profilePic": "https://example.com/profiles/player1.jpg",
        "deckId": null,
        "characterId": null,
        "isReady": false,
        "joinedAt": "2025-11-25T10:00:00.000Z"
      }
    ],
    "map": {
      "_id": "507f1f77bcf86cd799439040",
      "name": "Battlefield Alpha",
      "image": "https://example.com/maps/battlefield-alpha.jpg",
      "gridSize": {
        "width": 10,
        "height": 3
      },
      "difficulty": "medium"
    },
    "gameSettings": {
      "turnTimeLimit": 60,
      "allowSpectators": false
    },
    "createdAt": "2025-11-25T10:00:00.000Z",
    "startedAt": null,
    "expiresAt": "2025-11-25T10:30:00.000Z"
  }
}
```

6.2 Get Public Lobbies

Retrieves a paginated list of public lobbies.

Endpoint: `GET /api/v1/lobbies/public` **Access:** Private

Query Parameters:

- `page` : Integer >= 1, default: 1
- `limit` : Integer 1-50, default: 20
- `status` : 'waiting' | 'ready', optional
- `showFull` : Boolean, default: false

Example: `GET /api/v1/lobbies/public?page=1&limit=10&status=waiting&showFull=false`

Success Response (200 OK):


```
{
  "success": true,
  "message": "Public lobbies retrieved successfully",
  "data": {
    "lobbies": [
      {
        "_id": "507f1f77bcf86cd799439050",
        "lobbyName": "Epic Battle Room",
        "hostUsername": "player1",
        "playerCount": 1,
        "maxPlayers": 2,
        "isFull": false,
        "isPrivate": false,
        "status": "waiting",
        "mapName": "Battlefield Alpha",
        "createdAt": "2025-11-25T10:00:00.000Z"
      },
      {
        "_id": "507f1f77bcf86cd799439051",
        "lobbyName": "Duel Arena",
        "hostUsername": "player2",
        "playerCount": 2,
        "maxPlayers": 2,
        "isFull": true,
        "isPrivate": false,
        "status": "ready",
        "mapName": "Desert Wasteland",
```

```
        "createdAt": "2025-11-25T09:45:00.000Z"
      }
    ],
    "pagination": {
      "page": 1,
      "limit": 10,
      "totalPages": 1,
      "totalItems": 2
    }
  }
}
```

6.3 Get Current User's Active Lobby

Retrieves the lobby the authenticated user is currently in.

Endpoint: `GET /api/v1/lobbies/me/current` **Access:** Private

Success Response (200 OK):


```
{
  "success": true,
  "message": "Current lobby retrieved successfully",
  "data": {
    "_id": "507f1f77bcf86cd799439050",
    "lobbyName": "Epic Battle Room",
    "status": "waiting",
    "isPrivate": false,
    "maxPlayers": 2,
    "playerCount": 1,
    "isFull": false,
    "host": {...},
    "players": [...],
    "map": {...},
    "gameSettings": {...},
    "createdAt": "2025-11-25T10:00:00.000Z"
  }
}
```


Error Response (404 Not Found):

```
{
  "success": false,
  "message": "Not currently in any lobby"
}
```

6.4 Get Lobby by ID

Retrieves a specific lobby by its ID.

Endpoint: `GET /api/v1/lobbies/:lobbyId` **Access:** Private

URL Parameters:

- `lobbyId` : MongoDB ObjectId (required)

Success Response (200 OK):

```
{
  "success": true,
  "message": "Lobby retrieved successfully",
  "data": {
    "_id": "507f1f77bcf86cd799439050",
    "lobbyName": "Epic Battle Room",
    "status": "waiting",
    "isPrivate": false,
    "maxPlayers": 2,
    "playerCount": 2,
    "isFull": true,
    "host": {...},
    "players": [...],
    "map": {...},
    "gameSettings": {...}
  }
}
```

6.5 Join Lobby

Joins an existing lobby.

Endpoint: `POST /api/v1/lobbies/:lobbyId/join` **Access:** Private **Content-Type:** `application/json`

URL Parameters:

- `lobbyId` : MongoDB ObjectId (required)

Request Body:

```
{
  "password": "lobby_password"
}
```

Note: `password` is optional, only required for private lobbies.

Success Response (200 OK):


```
{
  "success": true,
  "message": "Successfully joined lobby",
  "data": {
    "_id": "507f1f77bcf86cd799439050",
    "lobbyName": "Epic Battle Room",
    "status": "waiting",
    "isPrivate": false,
    "maxPlayers": 2,
    "playerCount": 2,
    "isFull": true,
    "host": {...},
    "players": [
      {
        "_id": "u_1234567890abcdef",
        "username": "player1",
        "displayName": "Player One",
        "profilePic": "https://example.com/profiles/player1.jpg",
        "deckId": null,
        "characterId": null,
        "isReady": false,
        "joinedAt": "2025-11-25T10:00:00.000Z"
      },
      {
        "_id": "u_0987654321fedcba",
        "username": "player2",
        "displayName": "Player Two",
        "profilePic": "https://example.com/profiles/player2.jpg",
        "deckId": null,
        "characterId": null,
```

```
        "isReady": false,
        "joinedAt": "2025-11-25T10:05:00.000Z"
      }
    ],
    "map": {...},
    "gameSettings": {...}
  }
}
```

6.6 Leave Lobby

Leaves the current lobby.

Endpoint: `POST /api/v1/lobbies/:lobbyId/leave` **Access:** Private

URL Parameters:

- `lobbyId` : MongoDB ObjectId (required)

Request Body: None

Success Response (200 OK):


```
{
  "success": true,
  "message": "Successfully left lobby",
  "data": null
}
```

6.7 Select Deck in Lobby

Selects a deck for the current lobby.

Endpoint: `PUT /api/v1/lobbies/:lobbyId/deck` **Access:** Private **Content-Type:** `application/json`

URL Parameters:

- `lobbyId` : MongoDB ObjectId (required)

Request Body:

```
{
  "deckId": "507f1f77bcf86cd799439020"
}
```

Validation Rules:

- `deckId` : Valid MongoDB ObjectId, required

Success Response (200 OK):

```
{
  "success": true,
  "message": "Deck selected successfully",
  "data": {
    "_id": "507f1f77bcf86cd799439050",
    "lobbyName": "Epic Battle Room",
    "players": [
      {
        "_id": "u_1234567890abcdef",
        "username": "player1",
        "deckId": "507f1f77bcf86cd799439020",
        "characterId": null,
        "isReady": false
      }
    ]
  }
}
```

6.8 Select Character in Lobby

Selects a character for the current lobby.

Endpoint: `PUT /api/v1/lobbies/:lobbyId/character` **Access:** Private **Content-Type:** `application/json`

URL Parameters:

- `lobbyId` : MongoDB ObjectId (required)

Request Body:

```
{
  "characterId": "507f1f77bcf86cd799439025"
}
```

Validation Rules:

- `characterId` : Valid MongoDB ObjectId, required

Success Response (200 OK):

```
{
  "success": true,
  "message": "Character selected successfully",
  "data": {
    "_id": "507f1f77bcf86cd799439050",
    "lobbyName": "Epic Battle Room",
    "players": [
      {
        "_id": "u_1234567890abcdef",
        "username": "player1",
        "deckId": "507f1f77bcf86cd799439020",
        "characterId": "507f1f77bcf86cd799439025",
        "isReady": false
      }
    ]
  }
}
```

6.9 Update Lobby Settings

Updates lobby settings (host only).

Endpoint: `PUT /api/v1/lobbies/:lobbyId/settings` **Access:** Private (host only) **Content-Type:** `application/json`

URL Parameters:

- `lobbyId` : MongoDB ObjectId (required)

Request Body:

```
{
  "lobbyName": "Updated Room Name",
  "mapId": "507f1f77bcf86cd799439041",
  "gameSettings": {
    "turnTimeLimit": 90,
    "allowSpectators": true
  }
}
```

Note: All fields are optional, but at least one must be provided.

Success Response (200 OK):

```
{
  "success": true,
  "message": "Lobby settings updated successfully",
  "data": {
    "_id": "507f1f77bcf86cd799439050",
    "lobbyName": "Updated Room Name",
    "gameSettings": {
      "turnTimeLimit": 90,
      "allowSpectators": true
    }
  }
}
```

6.10 Kick Player from Lobby

Kicks a player from the lobby (host only).

Endpoint: `POST /api/v1/lobbies/:lobbyId/kick` **Access:** Private (host only) **Content-Type:** `application/json`

URL Parameters:

- `lobbyId` : MongoDB ObjectId (required)

Request Body:

```
{
  "playerId": "u_0987654321fedcba"
}
```

Validation Rules:

- `playerId` : Valid MongoDB ObjectId, required

Success Response (200 OK):

```
{
  "success": true,
  "message": "Player kicked successfully",
  "data": {
    "_id": "507f1f77bcf86cd799439050",
    "playerCount": 1,
    "players": [...]
  }
}
```

6.11 Start Game

Starts the game (host only).

Endpoint: `POST /api/v1/lobbies/:lobbyId/start` **Access:** Private (host only)

URL Parameters:

- `lobbyId` : MongoDB ObjectId (required)

Request Body: None

Requirements:

- Lobby must be full (2 players)
- All players must have selected deck and character

Success Response (200 OK):

```
{
  "success": true,
```

```
  "message": "Game started successfully",
  "data": {
    "gameId": "507f1f77bcf86cd799439060",
    "lobbyId": "507f1f77bcf86cd799439050",
    "status": "started"
  }
}
```

Note: This triggers WebSocket event `game:started` to all players in lobby.

6.12 Cancel/Delete Lobby

Cancels and deletes the lobby (host only).

Endpoint: `DELETE /api/v1/lobbies/:lobbyId` **Access:** Private (host only)

URL Parameters:

- `lobbyId` : MongoDB ObjectId (required)

Success Response (200 OK):

```
{
  "success": true,
  "message": "Lobby cancelled successfully",
  "data": null
}
```

---

7. GAME ENDPOINTS

All game endpoints require authentication.

7.1 Get Completed Game by ID

Retrieves a completed game's details.

Endpoint: `GET /api/v1/games/:gameId` **Access:** Private

URL Parameters:

- `gameId` : MongoDB ObjectId (required)

Success Response (200 OK):


```
{
  "success": true,
  "message": "Game retrieved successfully",
  "data": {
    "gameId": "507f1f77bcf86cd799439060",
    "lobbyId": "507f1f77bcf86cd799439050",
    "status": "completed",
    "winner": {
      "userId": "u_1234567890abcdef",
      "username": "player1",
      "displayName": "Player One"
    },
    "players": [
      {
        "userId": "u_1234567890abcdef",
        "username": "player1",
        "finalScore": 85,
        "deck": {...},
        "character": {...}
      },
      {
        "userId": "u_0987654321fedcba",
        "username": "player2",
        "finalScore": 72,
        "deck": {...},
        "character": {...}
      }
    ],
    "totalTurns": 15,
    "duration": 1200,
    "startedAt": "2025-11-25T10:00:00.000Z",
    "endedAt": "2025-11-25T10:20:00.000Z"
  }
}
```

7.2 Get User's Game History

Retrieves the authenticated user's game history with pagination.

Endpoint: `GET /api/v1/games/me/history` **Access:** Private

Query Parameters:

- `page` : Integer, default: 1
- `limit` : Integer, default: 20

Example: `GET /api/v1/games/me/history?page=1&limit=20`

Success Response (200 OK):


```
{
  "success": true,
  "message": "Game history retrieved successfully",
  "data": {
    "games": [
      {
        "gameId": "507f1f77bcf86cd799439060",
        "opponent": {
          "userId": "u_0987654321fedcba",
          "username": "player2",
          "displayName": "Player Two"
        },
        "result": "win",
        "myScore": 85,
        "opponentScore": 72,
        "endedAt": "2025-11-25T10:20:00.000Z"
      },
      {
        "gameId": "507f1f77bcf86cd799439061",
        "opponent": {
          "userId": "u_anotherplayer123",
          "username": "player3",
          "displayName": "Player Three"
        },
        "result": "loss",
        "myScore": 68,
        "opponentScore": 75,
        "endedAt": "2025-11-24T18:15:00.000Z"
      }
    ],
    "pagination": {
      "page": 1,
      "limit": 20,
      "totalPages": 5,
      "totalGames": 95
    }
```

```
  }
}
```

7.3 Get User's Recent Games

Retrieves the user's 10 most recent games.

Endpoint: `GET /api/v1/games/me/recent` **Access:** Private

Success Response (200 OK):


```
{
  "success": true,
  "message": "Recent games retrieved successfully",
  "data": {
    "games": [
      {
        "gameId": "507f1f77bcf86cd799439060",
        "opponent": {
          "username": "player2",
          "displayName": "Player Two"
        },
        "result": "win",
        "myScore": 85,
        "opponentScore": 72,
        "endedAt": "2025-11-25T10:20:00.000Z"
      }
    ],
    "count": 10
  }
}
```

7.4 Get User's Game Statistics

Retrieves comprehensive statistics for the authenticated user.

Endpoint: `GET /api/v1/games/me/stats` **Access:** Private

Success Response (200 OK):

```
{
  "success": true,
  "message": "Statistics retrieved successfully",
  "data": {
    "totalGames": 95,
    "wins": 62,
    "losses": 33,
    "winRate": 65.26,
    "averageScore": 78.5,
    "highestScore": 95,
    "totalPlaytime": 114000,
    "favoriteCharacter": {
      "characterId": "507f1f77bcf86cd799439040",
      "name": "Pepe Warrior",
      "gamesPlayed": 45
    },
    "recentForm": {
      "last10Games": "WWLWWLWWWL",
      "last10WinRate": 70.0
    }
  }
}
```

7.5 Get Global Leaderboard

Retrieves the global leaderboard.

Endpoint: `GET /api/v1/games/leaderboard` **Access:** Private

Query Parameters:

- `limit` : Integer, default: 100
- `timeframe` : 'all' | 'week' | 'month', default: 'all'

Example: `GET /api/v1/games/leaderboard?limit=50&timeframe=week`

Success Response (200 OK):


```
{
  "success": true,
  "message": "Leaderboard retrieved successfully",
  "data": {
    "leaderboard": [
```

```
      {
        "rank": 1,
        "userId": "u_topplayer12345",
        "username": "topplayer",
        "displayName": "Top Player",
        "profilePic": "https://example.com/profiles/top.jpg",
        "totalGames": 250,
        "wins": 195,
        "losses": 55,
        "winRate": 78.0,
        "points": 1950
      },
      {
        "rank": 2,
        "userId": "u_secondplace987",
        "username": "secondplace",
        "displayName": "Second Place",
        "profilePic": "https://example.com/profiles/second.jpg",
        "totalGames": 180,
        "wins": 135,
        "losses": 45,
        "winRate": 75.0,
        "points": 1800
      }
    ],
    "myRank": {
      "rank": 45,
      "points": 950
    }
  }
}
```

7.6 Get Head-to-Head Record

Retrieves head-to-head record against a specific opponent.

Endpoint: `GET /api/v1/games/head-to-head/:opponentId` **Access:** Private

URL Parameters:

- `opponentId` : User ID (required)

Success Response (200 OK):

```
{
  "success": true,
  "message": "Head-to-head record retrieved successfully",
  "data": {
    "opponent": {
      "userId": "u_0987654321fedcba",
      "username": "player2",
      "displayName": "Player Two",
      "profilePic": "https://example.com/profiles/player2.jpg"
    },
    "totalGames": 12,
    "myWins": 8,
    "opponentWins": 4,
    "draws": 0,
    "winRate": 66.67,
    "recentGames": [
      {
        "gameId": "507f1f77bcf86cd799439060",
        "result": "win",
        "myScore": 85,
        "opponentScore": 72,
        "playedAt": "2025-11-25T10:20:00.000Z"
      },
      {
        "gameId": "507f1f77bcf86cd799439055",
        "result": "loss",
        "myScore": 68,
        "opponentScore": 74,
        "playedAt": "2025-11-23T14:30:00.000Z"
      }
    ]
  }
}
```

7.7 Get Active Game State

Retrieves active game state from Redis (for ongoing games).

Endpoint: `GET /api/v1/games/active/:gameId` **Access:** Private

URL Parameters:

- `gameId` : Game ID (required)

Success Response (200 OK):


```
{
  "success": true,
  "message": "Active game state retrieved successfully",
  "data": {
    "gameId": "507f1f77bcf86cd799439060",
    "status": "active",
    "phase": "play",
    "currentTurn": "u_1234567890abcdef",
    "turnNumber": 5,
    "me": {
      "userId": "u_1234567890abcdef",
      "username": "player1",
      "position": "left",
      "hand": [
        {
          "cardId": "507f1f77bcf86cd799439011",
          "name": "Pepe Strike",
          "power": 5,
          "rarity": "epic",
          "cardType": "attack",
          "cardImage": "https://example.com/cards/pepe-strike.png",
          "pawnRequirement": 2
        }
      ],
      "deckCount": 15,
      "totalScore": 45,
      "rowScores": [15, 18, 12]
    },
    "opponent": {
      "userId": "u_0987654321fedcba",
      "username": "player2",
      "position": "right",
      "handCount": 5,
      "deckCount": 14,
      "totalScore": 42,
      "rowScores": [14, 16, 12]
    },
    "board": [
      [
        {
          "x": 0,
          "y": 0,
          "card": null,
          "owner": null,
```

```
          "pawns": {},
          "special": null
        },
        {
          "x": 1,
          "y": 0,
          "card": {
            "cardId": "507f1f77bcf86cd799439011",
            "name": "Pepe Strike",
            "power": 5
          },
          "owner": "u_1234567890abcdef",
          "pawns": {
            "u_1234567890abcdef": 2
          },
          "special": null
        }
      ]
    ]
  }
}
```

7.8 Forfeit Active Game

Forfeits the active game (instant loss).

Endpoint: `POST /api/v1/games/:gameId/forfeit` **Access:** Private

URL Parameters:

- `gameId` : Game ID (required)

Request Body: None

Success Response (200 OK):


```
{
  "success": true,
  "message": "Game forfeited successfully",
  "data": {
    "gameId": "507f1f77bcf86cd799439060",
    "status": "completed",
    "result": "forfeit",
    "winner": {
      "userId": "u_0987654321fedcba",
```

```
      "username": "player2"
    }
  }
}
```

---

8. GACHA ENDPOINTS

All gacha endpoints require authentication.

8.1 Pull Single Card

Pulls a single card from the gacha system.

Endpoint: `POST /api/v1/gacha/pull/single` **Access:** Private

Cost: 1 coin

Request Body: None

Success Response (200 OK):

```
{
  "success": true,
  "message": "Card pulled successfully",
  "data": {
    "card": {
      "cardId": "507f1f77bcf86cd799439070",
      "name": "Epic Doge Shield",
      "power": 7,
      "rarity": "epic",
      "cardType": "defense",
      "cardImage": "https://example.com/cards/doge-shield.png",
      "isNew": true
    },
    "coinsRemaining": 49,
    "pityCounter": {
      "epicPity": 5,
      "legendaryPity": 15
    }
  }
}
```

8.2 Pull 10 Cards

Pulls 10 cards from the gacha system.

Endpoint: `POST /api/v1/gacha/pull/multi` **Access:** Private

Cost: 10 coins

Guarantee: At least 1 epic card

Request Body: None

Success Response (200 OK):


```
{
  "success": true,
  "message": "10 cards pulled successfully",
  "data": {
    "cards": [
      {
        "cardId": "507f1f77bcf86cd799439070",
        "name": "Epic Doge Shield",
        "power": 7,
        "rarity": "epic",
        "cardType": "defense",
        "cardImage": "https://example.com/cards/doge-shield.png",
        "isNew": true
      },
      {
        "cardId": "507f1f77bcf86cd799439071",
        "name": "Common Pepe",
        "power": 3,
        "rarity": "common",
        "cardType": "attack",
        "cardImage": "https://example.com/cards/common-pepe.png",
        "isNew": false
      },
      {
        "cardId": "507f1f77bcf86cd799439072",
        "name": "Rare Wojak",
        "power": 5,
        "rarity": "rare",
        "cardType": "special",
        "cardImage": "https://example.com/cards/rare-wojak.png",
        "isNew": true
      }
      // ... 7 more cards
```

```
    ],
    "summary": {
      "legendary": 0,
      "epic": 2,
      "rare": 3,
      "common": 5
    },
    "coinsRemaining": 40,
    "pityCounter": {
      "epicPity": 0,
      "legendaryPity": 25
    }
  }
}
```

8.3 Get Gacha Info

Retrieves gacha system information and user's gacha stats.

Endpoint: `GET /api/v1/gacha/info` **Access:** Private

Success Response (200 OK):

```
{
  "success": true,
  "message": "Gacha info retrieved successfully",
  "data": {
    "coins": 50,
    "pityCounter": {
      "epicPity": 5,
      "legendaryPity": 15
    },
    "rates": {
      "legendary": {
        "baseRate": 1.0,
        "pityAt": 50
      },
      "epic": {
        "baseRate": 5.0,
        "pityAt": 10
      },
      "rare": {
        "baseRate": 20.0
      },
      "common": {
```

```
        "baseRate": 74.0
      }
    },
    "prices": {
      "single": 1,
      "multi": 10
    }
  }
}
```

WebSocket Events

Connection Setup

Connect to WebSocket server and authenticate:

```
import { io } from 'socket.io-client';

const socket = io('ws://localhost:3000', {
  auth: {
    token: 'your_jwt_access_token'
  }
});

// Connection events
socket.on('connect', () => {
  console.log('Connected to WebSocket server');
});

socket.on('disconnect', () => {
  console.log('Disconnected from WebSocket server');
});

socket.on('error', (error) => {
  console.error('Socket error:', error);
});
```

LOBBY EVENTS

Client → Server Events

1. lobby:joined

Join a lobby room after REST API join.

```
socket.emit('lobby:joined', {
  lobbyId: '507f1f77bcf86cd799439050'
});
```

2. lobby:leave

Leave the current lobby.

```
socket.emit('lobby:leave', {
  lobbyId: '507f1f77bcf86cd799439050'
});
```

3. lobby:update:settings

Update lobby settings (host only).

```
socket.emit('lobby:update:settings', {
  lobbyName: 'New Lobby Name',
  mapId: '507f1f77bcf86cd799439041',
  gameSettings: {
    turnTimeLimit: 90,
    allowSpectators: true
  }
});
```

4. lobby:select:deck

Select a deck.

```
socket.emit('lobby:select:deck', {
  deckId: '507f1f77bcf86cd799439020'
});
```

5. lobby:select:character

Select a character.

```
socket.emit('lobby:select:character', {
  characterId: '507f1f77bcf86cd799439025'
});
```

6. lobby:kick:player

Kick a player from the lobby (host only).

```
socket.emit('lobby:kick:player', {
  playerId: 'u_0987654321fedcba'
});
```

7. lobby:start:game

Start the game (host only).

```
socket.emit('lobby:start:game', {
  lobbyId: '507f1f77bcf86cd799439050'
});
```

8. lobby:ready:toggle

Toggle ready status.

```
socket.emit('lobby:ready:toggle', {
  isReady: true
});
```

Server → Client Events

1. lobby:reconnected

Player reconnected to lobby.

```
socket.on('lobby:reconnected', (data) => {
  console.log(data);
```

```
  // { lobbyId: '507f...', message: 'Reconnected to lobby' }
});
```

2. lobby:state:update

Lobby state updated (sent to all players in lobby).

```
socket.on('lobby:state:update', (lobbyDto) => {
  console.log('Lobby updated:', lobbyDto);
  // Full lobby state with all players, settings, etc.
});
```

3. lobby:left

Successfully left the lobby.

```
socket.on('lobby:left', (data) => {
  console.log(data.message);
  // { message: 'Successfully left lobby' }
});
```

4. lobby:kicked

Player was kicked from the lobby.

```
socket.on('lobby:kicked', (data) => {
  console.log(data.message);
  // { message: 'You have been kicked from the lobby' }
});
```

5. lobby:closed

Lobby was closed/deleted by host.

```
socket.on('lobby:closed', (data) => {
  console.log(data.message);
  // { message: 'Lobby has been closed' }
});
```

6. game:started

Game has started.

```
socket.on('game:started', (data) => {
  console.log('Game started:', data);
  // { lobbyId: '507f...', gameId: '507f...', message: 'Game is starting!' }
});
```

LOBBY LIST BROADCAST EVENTS

These events are broadcast to all connected clients watching the lobby list.

1. lobbyList:lobby:created

New lobby was created.

```
socket.on('lobbyList:lobby:created', (lobbyDto) => {
  console.log('New lobby created:', lobbyDto);
});
```

2. lobbyList:lobby:updated

Lobby was updated.

```
socket.on('lobbyList:lobby:updated', (lobbyDto) => {
  console.log('Lobby updated:', lobbyDto);
});
```

3. lobbyList:lobby:deleted

Lobby was deleted.

```
socket.on('lobbyList:lobby:deleted', (data) => {
  console.log('Lobby deleted:', data.lobbyId);
  // { lobbyId: '507f...' }
});
```

4. lobbyList:lobby:started

Lobby started a game.

```
socket.on('lobbyList:lobby:started', (data) => {
  console.log('Lobby started:', data.lobbyId);
  // { lobbyId: '507f...' }
});
```

GAME EVENTS

Client → Server Events

1. game:join

Join a game.

```
socket.emit('game:join', {
  gameId: '507f1f77bcf86cd799439060'
});
```

2. game:dice_roll:submit

Submit dice roll (for determining first player).

```
socket.emit('game:dice_roll:submit', {});
```

3. game:action:hover

Preview card placement before playing.

```
socket.emit('game:action:hover', {
  cardId: '507f1f77bcf86cd799439011',
  x: 3,
  y: 1
});
```

4. game:action:play_card

Play a card from hand.

```
socket.emit('game:action:play_card', {
  cardId: '507f1f77bcf86cd799439011',
  handCardIndex: 2,
  x: 3,
  y: 1
});
```

5. game:action:skip_turn

Skip the current turn.

```
socket.emit('game:action:skip_turn', {});
```

6. game:leave

Leave the game.

```
socket.emit('game:leave', {
  gameId: '507f1f77bcf86cd799439060'
});
```

Server → Client Events

1. game:load

Initial game state (sent when player joins).

```
socket.on('game:load', (gameState) => {
  console.log('Game loaded:', gameState);
  // Full game state with player-specific view
});
```

2. game:state:update

Game state updated.

```
socket.on('game:state:update', (gameState) => {
  console.log('Game state updated:', gameState);
```

```
  // Updated game state after actions
});
```

3. game:dice_roll:start

Start rolling dice for first turn.

```
socket.on('game:dice_roll:start', (data) => {
  console.log(data.message);
  // { message: 'Roll for first turn!' }
});
```

4. game:dice_roll:wait

Waiting for opponent to roll dice.

```
socket.on('game:dice_roll:wait', (data) => {
  console.log('My roll:', data.myRoll);
  // { myRoll: 5, message: 'Waiting for opponent...' }
});
```

5. game:dice_roll:result

Dice roll results.

```
socket.on('game:dice_roll:result', (data) => {
  console.log('Dice results:', data);
  // {
  //   type: 'result' | 'tie',
  //   player1Roll: 5,
  //   player2Roll: 3,
  //   firstPlayer: 'u_1234567890abcdef'
  // }
});
```

6. game:action:preview

Card placement preview.

```
socket.on('game:action:preview', (preview) => {
  console.log('Card preview:', preview);
  // {
  //   isValid: true,
  //   pawnLocations: [{x: 3, y: 1}, {x: 4, y: 1}],
  //   effectLocations: [{x: 3, y: 0}],
  //   message: 'Card can be played here'
  // }
});
```

7. game:end

Game ended.

```
socket.on('game:end', (data) => {
  console.log('Game ended:', data);
  // {
  //   status: 'completed',
  //   winnerId: 'u_1234567890abcdef',
  //   finalScore: { player1: 85, player2: 72 },
  //   coinsWon: 10
  // }
});
```

8. game:error

Game error occurred.

```
socket.on('game:error', (data) => {
  console.error('Game error:', data.message);
  // { message: 'Error description' }
});
```

Error Handling

Standard Error Response Format

All API errors follow this consistent format:

```
{
  "success": false,
  "message": "Human-readable error message",
  "errors": [
    {
      "field": "fieldName",
      "message": "Specific error for this field"
    }
  ]
}
```

HTTP Status Codes

Code	Meaning	Usage
200	OK	Request successful
201	Created	Resource created successfully
400	Bad Request	Validation error or malformed request
401	Unauthorized	Authentication required or failed
403	Forbidden	Authenticated but not authorized
404	Not Found	Resource not found
409	Conflict	Resource conflict (e.g., duplicate)
422	Unprocessable Entity	Semantic validation error
500	Internal Server Error	Server error

Common Error Examples

Validation Error (400)

```
{
  "success": false,
  "message": "Validation error",
  "errors": [
```

```
    {
      "field": "email",
      "message": "Invalid email format"
    },
    {
      "field": "password",
      "message": "Password must be at least 8 characters"
    }
  ]
}
```

Authentication Error (401)

```
{
  "success": false,
  "message": "Authentication required"
}
```

```
{
  "success": false,
  "message": "Invalid or expired token"
}
```

Authorization Error (403)

```
{
  "success": false,
  "message": "You do not have permission to access this resource"
}
```

Not Found Error (404)

```
{
  "success": false,
```

```
  "message": "Resource not found"
}
```

Conflict Error (409)

```
{
  "success": false,
  "message": "Email already registered"
}
```

Appendix

File Structure Reference

```
BE/card-game-server/
├── src/
│   ├── routes/
│   │   ├── auth.routes.js
│   │   ├── deck.routes.js
│   │   ├── inventory.routes.js
│   │   ├── user.routes.js
│   │   ├── friend.routes.js
│   │   ├── lobby.routes.js
│   │   ├── game.routes.js
│   │   └── gacha.routes.js
│   ├── controllers/
│   │   ├── auth.controller.js
│   │   ├── Deck.controller.js
│   │   ├── inventory.controller.js
│   │   ├── user.controller.js
│   │   ├── friend.controller.js
│   │   ├── Lobby.controller.js
│   │   ├── Game.controller.js
│   │   └── Gacha.controller.js
│   ├── sockets/
│   │   ├── index.js
│   │   ├── Lobby.socket.js
│   │   ├── Game.socket.js
│   │   └── Lobbylist.broadcast.js
```

```
│   ├── middlewares/
│   │   ├── auth.middleware.js
│   │   ├── validation.middleware.js
│   │   └── errorHandler.middleware.js
│   ├── dto/
│   │   └── validation/
│   ├── app.js
│   └── server.js
└── .env
```

Document End

For questions or issues, please contact the development team.