Hidden Joker Rummy – API Design Document

Team z-g

1. Game Description

Our Hidden Joker Rummy is a turn-based multiplayer card game where players draw, discard, and form melds (sets and runs) to empty their hand. At the beginning of each game, the server randomly selects a hidden joker rank, which acts as a wildcard and can complete sets or runs. Players take turns drawing from the stock or discard pile, discarding, and optionally declaring “Rummy” by submitting valid melds that use all their cards.

2. API Endpoints

Endpoint – Create Game

Field	Description
1. Endpoint Name	Create Game
2. HTTP Method & Route	**POST /api/games**
3. Purpose	Creates a new Hidden Joker Rummy game lobby and assigns the requesting user as the host.
4. Authorization	Must be logged in.
5. Request Body	`{ "max_players": 4, "visibility": "public" }`
6. Validation Checks	- User authenticated - `max_players` between 2–4 - `visibility` is `"public"` or `"private"`

7. State Updates	- Insert row into **games** (host_user_id, max_players, visibility, state = `"lobby"`) - Insert row into **game_players** for host as seat #1 - Optionally generate a **room_code** for invite joins
8. Success Response	**201 Created** + `{ "game_id": 123, "room_code": "ABCD12" }`
9. Error Cases	**401** (not logged in), **400** (invalid max_players or visibility)
10. Socket.io Events	none

Endpoint – Join Game

Field	Description
1. Endpoint Name	Join Game
2. HTTP Method & Route	**POST /api/games/:game_id/join**
3. Purpose	Adds the logged-in user to an existing Hidden Joker Rummy lobby.
4. Authorization	Must be logged in.
5. Request Body	`{ "room_code": "ABCD12" }` *(if your game uses join codes; otherwise this can be empty)*
6. Validation Checks	- Game exists - Game state is `"lobby"` (waiting) - If provided, `room_code` matches game's room_code - Seats available (`current_players < max_players`) - User is not already in this game
7. State Updates	Insert new row into **game_players** for this user and game (assign next available seat).
8. Success Response	**202 Accepted**
9. Error Cases	**404** (game not found or wrong room_code), **403** (game full or already started, or user already joined)
10. Socket.io Events	`game:player:joined`, `game:state:update`

Endpoint – Start Game

Field	Description
1. Endpoint Name	Start Game
2. HTTP Method & Route	**POST /api/games/:game_id/start**
3. Purpose	Moves game from lobby → playing, shuffles the deck, selects the hidden joker rank, and deals initial hands to all players.
4. Authorization	Must be the host of the game.
5. Request Body	none
6. Validation Checks	- Game exists - Requesting user is `host_user_id` - Game state is `"lobby"` - Player count ≥ 2
7. State Updates	- Create full deck for this game and shuffle it - Randomly choose `hidden_joker_rank` and save to **games** - Deal starting hands (e.g., 13 cards) to each player (update card ownership) - Move one card to discard pile, rest remain in stock - Set `current_player_id` and `turn_number = 1` - Set game state to `"playing"`
8. Success Response	**202 Accepted**
9. Error Cases	**403** (not host), **404** (game not found), **409** (game not in lobby or already started)
10. Socket.io Events	`game:state:update`, `game:hand:update` (to each player), `game:turn:changed`

Endpoint – Get Game State

Field	Description

1. Endpoint Name	Get Game State
2. HTTP Method & Route	**GET /api/games/:game_id**
3. Purpose	Returns the current public game state plus the requesting player's private hand.
4. Authorization	Must be logged in and must be a player in the game.
5. Request Body	none
6. Validation Checks	- Game exists - User exists in **game_players** for this game
7. State Updates	none (read-only)
8. Success Response	**200 OK** + JSON containing: - Game info (state, current_player_id, hidden_joker_rank, turn_number) - Public state (discard_top, stock_count, each player's hand_size, scores) - Private state (requesting player's full hand)
9. Error Cases	**403** (user not in game), **404** (game not found)
10. Socket.io Events	none

Endpoint – Draw Card

Field	Description
1. Endpoint Name	Draw Card
2. HTTP Method & Route	**POST /api/games/:game_id/draw-card**
3. Purpose	Allows the current player to draw a card from either the stock pile or the top of the discard pile.
4. Authorization	Must be logged in and must be the current player (unless special penalty rules are added later).
5. Request Body	`{ "source": "stock" }` or `{ "source": "discard" }`

6. Validation Checks	- Game exists - Game state is `"playing"` - User is in **game_players** - User is `current_player_id` - `source` is `"stock"` or `"discard"` - If `source === "stock"`: stock_count > 0 - If `source === "discard"`: discard pile not empty - Player has not already drawn this turn (per Rummy rules)
7. State Updates	- Remove top card from chosen pile (stock or discard) - Assign that card to player's hand (update card owner) - Mark that player as `has_drawn = true` for current turn
8. Success Response	**202 Accepted**
9. Error Cases	**403** (not your turn, not in game, or already drew this turn), **404** (game not found), **409** (no cards available / invalid game state)
10. Socket.io Events	`game:hand:update` (to drawing player), `game:state:update`

Endpoint – Discard Card

Field	Description
1. Endpoint Name	Discard Card
2. HTTP Method & Route	**POST /api/games/:game_id/discard-card**
3. Purpose	Current player discards one card from their hand to the discard pile, usually ending their turn.
4. Authorization	Must be logged in and must be the current player in the game.
5. Request Body	`{ "card_id": "c_42" }`

6. Validation Checks	- Game exists - Game state is `"playing"` - User is in **game_players** - User is `current_player_id` - `card_id` is provided and belongs to this player's hand - Player has drawn this turn, if rules require draw before discard
7. State Updates	- Remove card from player's hand - Set that card as new `discard_top` - Increment `turn_number` - Compute next `current_player_id` and set it - Reset per-turn flags (like `has_drawn`) for new current player
8. Success Response	**202 Accepted**
9. Error Cases	**400** (missing/invalid card_id), **403** (card not owned or not your turn), **404** (game or card not found), **409** (invalid game state)
10. Socket.io Events	`game:hand:update`, `game:state:update`, `game:turn:changed`

Endpoint – Declare Rummy (Submit Melds)

Field	Description
1. Endpoint Name	Declare Rummy (Submit Melds)
2. HTTP Method & Route	**POST /api/games/:game_id/submit-melds**
3. Purpose	Current player submits their melds (sets and runs) and attempts to declare Rummy using all their cards (except optional final discard). If valid, the game ends and scores are calculated.
4. Authorization	Must be logged in and must be the current player in the game.
5. Request Body	`json<br>{<br> "melds": [<br> { "type": "run", "card_ids": ["c_10","c_11","c_12"] },<br> { "type": "set", "card_ids": ["c_25","c_38","c_51"] }<br> ],<br> "discard_card_id": "c_99"<br>}<br>`

6. Validation Checks	- Game exists - Game state is `"playing"` - User is in **game_players** and is `current_player_id` - All `card_ids` in melds and `discard_card_id` belong to this player's hand - Each meld is structurally valid: • **run**: same suit, consecutive ranks (jokers may substitute ranks) • **set**: same rank, different suits - Hidden joker rule applied correctly (cards of `hidden_joker_rank` treated as wildcards) - All or all-but-one cards in player's hand are accounted for across melds + discard (depending on chosen Rummy variant)
7. State Updates	- Move meld cards (and optional discard card) from player's hand into **game_melds** / table area - Compute final scores for all players based on remaining cards in their hands - Set `winner_player_id` to this player - Set game state to `"ended"`
8. Success Response	**202 Accepted**
9. Error Cases	**400** (invalid meld structure, missing cards, or joker misuse), **403** (not your turn or not in game), **404** (game not found), **409** (game not in playing state / race condition)
10. Socket.io Events	- `game:melds:submitted` → all players - `game:state:update` → all players (state `"ended"`, public melds) - `game:ended` → all players (winner + final scores) - Optionally `game:hand:update` to winner if their hand representation changes

3. Socket.io Events

Event – game:player:joined

Event Name: `game:player:joined`
Scope: All players in the game room
Trigger: A playe successfully joins a game
Data:

```
{
  "game_id": 123,
  "players": [
    { "player_id": 1, "display_name": "Alice" },
    { "player_id": 2, "display_name": "Bob" }
  ],
  "max_players": 4
}
```

Event – game:state:update

Event Name: `game:state:update`
Scope: All players in the room
Trigger: Public state changes (start,draw,discard, meld submission, game end)
Data:

```
{
  "game_id": 123,
  "phase": "playing",
  "current_player_id": 2,
  "turn_number": 5,
  "discard_top": { "id": "c_42", "rank": "7", "suit": "hearts" },
  "stock_count": 28,
  "players": [
    { "player_id": 1, "hand_size": 10 },
    { "player_id": 2, "hand_size": 11 }
  ],
  "hidden_joker_rank": "Q"
}
```

Event – game:turn:changed

Event Name: `game:melds:submitted`
Scope: All players
Trigger: After a valid discard, pass, or rummy declaration that advances the turn

Data:

```
{
  "game_id": 123,
  "previous_player_id": 2,
  "next_player_id": 3,
  "turn_number": 6
}
```

Event – game:melds:submitted

Event Name: `game:melds:submitted`
Scope: All players
Trigger: Player submits valid melds to declare Rummy

Data:

```
{
  "game_id": 123,
  "player_id": 2,
  "melds": [
    { "type": "run", "cards": ["c_20","c_21","c_22"] },
    { "type": "set", "cards": ["c_35","c_48","c_61"] }
  ]
}
```

Event – game:ended

Event Name: `game:ended`
Scope: All players
Trigger: The game transitions to the `"ended"` state (valid rummy declaration or stock exhaustion)
Data:

```
{
  "game_id": 123,
  "winner_player_id": 2,
  "final_scores": [
    { "player_id": 1, "score": 80 },
    { "player_id": 2, "score": 30 },
    { "player_id": 3, "score": 95 }
  ],
  "reason": "rummy_declared"
}
```

Event – game:hand:update

Event Name: `game:hand:update`
Scope: Single player only (the player whose hand changed)

Trigger: That player's hand changes (draw, discard, meld validation, or initial deal)
Data:

```
{
  "game_id": 123,
  "player_id": 2,
  "hand": [
    { "id": "c_12", "rank": "3", "suit": "clubs" },
    { "id": "c_87", "rank": "Q", "suit": "spades", "is_joker": true }
  ]
}
```