Milestone 3: Game API Design

Game Overview
Crazy Eights is a turn-based card game. Players try to discard all cards in their hand by matching the rank or suit of the top card. Eights are wild and let the player choose a new suit.

Public and Private State

Public State
Current Turn
Active Players
Top card on discard pile
Current Suit (if an eight was played)
Cards left in deck
Cards left for each player
Game phase

Private State
Player hand
Drawn card
Chosen suit after wild card

Socket.io Events

game:state:update
Scope: all players
Trigger: any change to shared state
Data:
```
{
  game_id,
  turn,
  current_player_id,
  top_card,
  current_suit,
  player_card_counts,
  deck_count
}
```

game:turn:changed
Scope: all players
Trigger: after a turn ends

Data:
{
  game_id,
  next_player_id,
  turn
}

game:hand:update
Scope: one player
Trigger: player draws or discards
Data:
{
  game_id,
  player_id,
  hand
}

game:player:joined
Scope: all players
Trigger: player joins lobby
Data:
{
  game_id,
  player_id,
  player_count
}

API Endpoints
Create Game
Method: POST /api/games
Purpose: start a new game
Authorization: logged in
Validation: user session exists
State Updates: create game in lobby
Success: 202
Events: none

Join Game
Method: POST /api/games/:id/join
Purpose: join a lobby

Authorization: logged in
Validation: game exists, game in lobby, player not already in game
State Updates: add player to game
Success: 202
Events: game:player:joined

Start Game
Method: POST /api/games/:id/start
Purpose: deal cards and begin play
Authorization: host only
Validation: game exists, user is host, enough players joined
State Updates: deal cards, set top card, set turn
Success: 202
Events: game:state:update, game:turn:changed

Get Game State
Method: GET /api/games/:id/state
Purpose: return full state for player
Authorization: must be in game
Validation: game exists, player in game
State Updates: none
Success: 200
Events: none

Game Action Endpoints
Play Card
Method: POST /api/games/:id/play
Purpose: discard a card
Request Body:
```
{
  card,
  chosen_suit
}
```
Authorization: must be current player
Validation:
Game exists
Player in game
Game in playing phase
It is this player's turn
Player owns this card

Card matches rank or suit
If card is an eight, chosen suit is valid
State Updates:
Remove card from hand
Update top card
Update current suit
Check win condition
Advance turn (because card can only be played if valid, thus ending turn)
Success: 202
Events: game:state:update, game:turn:changed, game:hand:update

Draw Card
Method: POST /api/games/:id/draw
Purpose: draw one card
Authorization: must be current player
Validation: game exists, player in game, deck has cards
State Updates: add card to player hand
Success: 202
Events: game:hand:update

End Turn
Method: POST /api/games/:id/end-turn
Purpose: pass the turn
Authorization: must be current player
Validation: game exists, player in game, correct player turn
State Updates: advance to next player
Success: 202
Events: game:turn:changed

Security Notes
Server validates all actions.
Client never decides if a card is legal.
Private hands stay private (info shouldn't be stored globally on any one client).
All actions check ownership and turn order.

Edge Cases
Deck is empty.
Player draws but still cannot play.
Player tries to play illegal card.
Two actions sent at the same time.

Player disconnects during their turn.