Milestone 3 group: z-i

Game description

Playing a game of UNO based on modified Mattel rules. Each player is given seven cards to start, and can discard cards in their hands by playing a wild card or a card that matches the color, number, or action of the top card of the discard pile. If a player draws a card, they forfeit their turn for that round. The object of the game is to be the first player to get rid of all the cards.

Socket.io Events Planning

Event: game: state: update

Scope: All players in the game room, users who played the card
Trigger: When the active player successfully plays a card, the room should be notified of the card played, and the card should disappear from the current player's hand
Data:
{
game_id: number,
player_id: number,
card_id: number,
next_player_id: number,
top_discard: card
}

Event: game: player joined

Scope: All players in the game room
Trigger: When a new player joins the room
Data:
{
game_id: number,
player_id: number,
seat: number
}

game:state: update - game state changed (updates number of cards in everyone's hands, as well as who the next player is)
game: player: joined
game: hand: draw - player has to draw
game: hand: update - player's hand changed (private)

game : ended - game ended
game:chat: updated - new chat message in game room
lobby :chat :updated - new chat message in lobby
lobby: game: created - new game created
lobby :game :updated - list of games updated (if a game is full)

API endpoints

Endpoint Name: Play Card

HTTP Method and route: POST /api/games/:game_id/cards/:card_id/play
Purpose: Player plays a card from their hand
Authorization:

- User must be authenticated
- The player must be in this game
- Must be the user's turn
- The game must be active

Request body:

{"top_discard": {"color": green, "value": 4} }

Validation checks:

- Game with game_id exists
- User is a player in the game
- The current player is the client who is trying to play the card
- Game state is active/playing
- The card with card_id is in the player's hand
- Card can be played (matching color/number/action/wild)

State updates:

- UPDATE game_player hand (remove card that was played)
- UPDATE game.top_discard with played game
- UPDATE game SET current seat to the next player based on the direction
- INSERT move_log with the information of the move (unless it is a wild card)

Success response:

202 Accepted

Error Cases:

- 404 Unauthorized - Not logged in
- 403 Forbidden - Not the current player, or not a player in the game
- 404 Not Found - Game or card doesn't exist
- 400 Bad Request - Invalid move, illegal play
- 409 Conflict - Game state changed since request started

Socket.io events:

- game: state: update → All players in game
- game: hand: update → Player who played the card

- game: hand: draw → Player who has to draw card has their hand updated

Endpoint Name: Select wild color

HTTP Method and route: POST /api/games/:game_id/wild_color/:color
Purpose: A player just played a wild card, so now they need to select the color
Authorization:

- User must be authenticated
- The player must be in this game
- Must be the user's turn
- The card just played must be a wild card
- The game must be active

Request body:

{"top_discard": {"color": green, "value": "wild"} }

Validation checks:

- Game with game_id exists
- User is a player in the game
- The current player is the client who just played a wild card
- Game state is active/playing
- The color chosen is one of the 4

State updates:

- UPDATE game_player hand (remove card that was played)
- UPDATE game.top_discard with played game
- UPDATE game SET current seat to the next player based on the direction
- INSERT move_log with the information of the color selection

Success response:

202 Accepted

Error Cases:

- 404 Unauthorized - Not logged in
- 403 Forbidden - Not the current player, or not a player in the game
- 404 Not Found - Game or card doesn't exist
- 400 Bad Request - Invalid move, illegal play
- 409 Conflict - Game state changed since request started

Socket.io events:

- game: state: update → All players in game
  {
  game_id: 123,
  current_seat: 2,
  turn: 3,
  top_discard_card: card_id
  }

Endpoint Name: Create Game

HTTP Method and route: POST /api/game/:game_id/player/:user_id/create
Purpose: A player creates a game
Authorization:

- User is a valid user
- Values selected for the game are allowed

Request body:
{ “game_id”: {:id} }
Validation checks:

- Game with game_id doesn’t exist yet
- The user who created the game is listed as a player
- room.status == “waiting”

State updates:

- INSERT game into game table
- INSERT user_id into game_player table with the new game_id
- INSERT room into room_id into room table

Success response:
202 accepted
Error Cases:

- 401 Unauthorized - not logged in
- 400 Bad Request - Game variables out of bounds
- 409 Conflict - Information has changed since the request started

Socket.io events:

- lobby: game: update → All players in lobby
  {
  game_id: 123,
  max_players: 4
  number_of_players:1
  }

Endpoint Name: Join game

HTTP Method and route: POST /api/games/:game_id/player/:user_id/add
Purpose: A player joins a game (from the lobby)
Authorization:

- User must be authenticated
- User must not already be a player in the game
- game.state = “waiting”

Request body:
{“add_player” : {user_id} }
Validation checks:

- Game with game_id exists
- User with user_id is a valid user

- The number of players is less than or equal to the maximum number of players
- User with user_id is not already in the player_list
- game.state == “waiting”

State updates:

- INSERT into game_player user_id…

Success response:

202 Accepted

Error Cases:

- 401 Unauthorized - Not logged in
- 404 Not Found - Game doesn’t exist
- 409 Conflict - Game state changed since the request started

Socket.io events:

- lobby: game: update → All players in lobby
  {
  game_id: 123,
  max_players: 4
  Number_of_players: 3
  }
- game: player: update → All players in the game
  {
  game_id: 123
  user_id: 1234
  }

Endpoint Name: Start game

HTTP Method and route: POST /api/gmes/:game_id/status/start

Purpose: There are enough players in the room to start the game

Authorization:

- User must be authenticated
- Game must be in “waiting” state

Request body:

{“game_status”: {playing} }

Validation checks:

- Game with game_id exists
- Number of players in games is less than or equal to the max_number of players
- Game_player are valid users

State updates:

- UPDATE game_status to active/playing
- INSERT deck of cards for the new game
- INSERT

Success response:

202 Accepted

Error Cases:

- 401 Unauthorized- Not logged in
- 403 Forbidden - Not a player in the game
- 409 Conflict - Game state changed since request started

Socket.io events:

- game:state: update
  {
  game_id: 123
  game_status: “playing”
  }

Endpoint Name: Endgame

HTTP Method and route: POST /api/gmes/:game_id/status/end
Purpose: A player has played all their cards
Authorization:

- User must be authenticated
- Game must be in “playing” state

Request body:

{}

Validation checks:

- Game with game_id exists
- Game status is playing

State updates:

- UPDATE game_status to active/finished
- UPDATE game stats with relevant information

Success response:

202 Accepted

Error Cases:

- 401 Unauthorized- Not logged in
- 403 Forbidden - Not a player in the game
- 404 Not Found - Game doesn’t exist
- 409 Conflict - Game state changed since request started

Socket.io events:

- game:state: update

Endpoint Name: Draw card

HTTP Method and route: POST /api/game/:game_id/draw_card
Purpose: At a player’s turn, they decide to draw a card instead of playing a card
Authorization:

- User must be authenticated
- User must be a player in the game
- Must be the user’s turn
- Game must be in “playing” state

Request body:

{}

Validation checks:

- Game with game_id exists
- User is in game_players table for this game
- User is the current player
- Game_stae == playing/active

State updates:

- UPDATE player_hands add card
- UPDATE games SET current player = next player
- INSERT move_log with information
- UPDATE deck SET card removed to null, or delete

Success response:

202 Accepted

Error Cases:

- 401 Unauthorized - Not logged in
- 403 Forbidden - Not your turn in game
- 404 - Not Found - Game doesn't exist
- 409 Conflict - Game state changed since request started

Socket.io events:

- game: state: update → All players in game
- game: hand: draw → player

Endpoint Name: Send message (lobby)

HTTP Method and route: POST /api/lobby/chat

Purpose: A user submitted a comment to the lobby

Authorization:

- User is authenticated

Request body:

{"user": {user_id},
"message": {message}}

Validation checks:

- User is logged in

State updates:

- INSERT into messages: entry with message, room, and user id

Success response:

202 Accepted

Error Cases:

- 401 Unauthorized - Not logged in
- 409 Conflict - Lobby state changed since the request started

Socket.io events:

- lobby: chat: updated → everyone in the lobby

Endpoint Name: Send message (game)

HTTP Method and route: POST /api/game/:game_id/chat
Purpose: A user posted a comment in a game room
Authorization:

- User is authenticated
- User is in the game

Request body:

{“user”: {user_id),
“message”:{message}}

Validation checks:

- User is logged in
- User is in the game
- Game exists

State updates:

- INSERT into message: entry with message, room, and user id

Success response:

202 Accepted

Error Cases:

- 401 Unauthorized - Not logged in
- 404 Not Found - Game doesn’t exist
- 409 Conflict - Chat state changed sincethe request started

Socket.io events:

- game: chat: updated → everyone in the game