Milestone 3:
Game API Design

UNO!

Team z-h: Carlos Rodriguez-Malak, Santiago Aguilar, Wilmaire Mejia, Fady Youssef

High Level Endpoints

Game

POST /api/games → *creates new UNO game*
GET /api/games/:gameId → *gets current game state*
POST /api/games/:gameId/start → *starts game once all players have joined*
POST /api/games/:gamesId/end → *ends game and record result*
POST /api/games/:gameId/play-card → *play a card during the player's turn*
POST /api/games/:gameId/draw-card → *draw a card from the deck*
GET /api/games/:gameId/turn → *checks whose turn it is and turn over*

Global Chat

GET /api/chat/global → *fetches all messages in global chat room*
POST /api/chat/global → *sends messages to global chat*

Game Chat

GET /api/games/:gameId/chat → *retrieves all chat messages in game*
POST /api/games/:gamesId/chat → *sends message to game chat*

User & Profile

POST /api/users/register → *create new user*
POST /api/users/login → *authorized user*
GET /api/users/:id → *receives user information*
GET /api/users/:id/profile → *grabs user's profile info*
PUT /api/users/:id/profile → *updates profile*

High Level Endpoints

Game Results & History

GET /api/games/:gameId/results → *retrieves final results of game*
POST /api/games/:gameId/results → *saves game results (for outcomes/ranks)*
GET /api/games/:gameId/history → *retrieves full game history (start/end times, player performance, and rank)*

Game Cards

GET /api/games/:gameId/cards → *retrieves all cards in the game and their location*
PUT /api/games/:gameId/cards/:cardId → *moves card (deck to hand, hand to discard)*

Participants

GET /api/games/:gameId/participants - *lists all players in game, their turn order, cards in hand, and UNO status*
POST /api/games/:gameId/ participants - *adds players to game*
PUT /api/games/:gameId/participants/:participantsId - *updates participant info (cards, UNO call, etc.)*

Endpoint Play-Card

Validation Checks

- Must ensure request has required fields: playerId, gameId, card
- Verify that the provided game and player exist before using.
- Check that it is the correct players turn.
- Confirm the player owns the card they are placing.
- Validate the card attempting to be played (UNO rules).
- If a special card is placed ensure the rules are maintained.

Edge Cases

- The next player disconnects
- A player never plays a card

State Updates

- Remove cards from hand
- Update top card on pile
- Add card to player hand
- Turn skips
- No cards left ends the game

Event Triggers

- game:card:played - show all players
- player:hand:remove - remove the played card from hand
- game:turn:skip - Skip a players turn
- game:end - A player no longer has any cards ends the game

Endpoint Draw Card

Validation Checks

- Ensure the system has playerId and gameId
- Confirm its the players turn as you cannot draw cards out of turn
- Make sure the deck resets if its empty

Edge Cases

- Deck runs out of cards
- If a player draws a card that is playable
- Drawing cards after the game has ended

State Updates

-
- Draw top card from the deck
- Update the deck count
- Drawing a card ends players turn next player turn automatic

Event Triggers

- game:card:drew - new card added to players hand
- game:player:drew - new card for player to see(priv)
- game:deck:update - new deck count
- game:turn:update - new card drawn skips players turn

Socket.io Event Summary

Game Events

Focus on in-game mechanics these are responsible for the flow of the game, game rules to be upheld, and prevent client side interference.

Previous Examples:

- game:card:played - A card was played
- game:card:drew - Player draws a card

Player Events

Main focus is the player and handling their cards, ensuring they are following the rules of the game, and who's turn it is currently.

Previous Example:

- player:hand:init - Players initial hand start of game.

Lobby Events

These events will be to used for creating games, joining games, and handling lobby chat.

Previous Example:

- lobby:game:create - Create a new game