Game Description

We are recreating UNO which is a turn-based multiplayer card game where players take turns matching a card in their hand to the top of the discard pile by color (red, blue, green, yellow) or value (0-9). UNO also contains special action cards (Reverse, skip, draw two, wild, wild draw four), the players are supposed to use these to make strategic choices. The goal is to be the first to play all of your cards while managing turns, colors, changes and penalties.

List of All API Endpoints

- **Endpoint 1**
  - **Name:** Create a Game
  - **HTTP Method & Route:** POST /api/games
  - **Purpose:** Creates a new UNO game lobby and assigns the requesting user as the host.
  - **Authorization:** All users must be signed in
  - **Body:** { "players": [{ "id": 0 }] }
  - **Validation Checks:**
    - Users are all authorized to play
    - Number of players is between 2-10 (2 minimum, 4 maximum)
  - **State Updates:**
    - Set Game to Lobby State (Default):
      - Insert host player into gameParticipants SQL Table
      - Insert created game into Games SQL Table
  - **Success Response:** 201 Created
  - **Error Cases:**
    - 400 - Invalid Settings
    - 401 - Not Logged in
  - **Socket IO Events:**
    - Game:player:joined → Send to Host
- **Endpoint 2**
  - **Name:** Join Game
  - **HTTP Method & Route:** POST /api/games/:gameId/join
  - **Purpose:** Adds a player to an existing lobby
  - **Authorization:** Authenticated user
  - **Body:** {“gameId”: 0, “player”: {“id”: 1, “authToken”: “xyz”}}
  - **Validation Checks:**
    - Game exists
    - Game is in “lobby”
    - Player is not already in the game
  - **State Updates:** insert into gamePlayers

    - **Success Response:** 202 Accepted
    - **Error Cases:**
        - 404 - Game not found
        - 403 - game already started/full
        - 409 - Already joined
    - **Socket IO Events:** game:player:joined (public)
- **Endpoint 3**
    - **Name:** Start Game
    - **HTTP Method & Route:** POST /api/games/:gameId/start
    - **Purpose:** Host starts game, the deck is shuffled, hands are dealt and turns set.
    - **Authorization:** Host only
    - **Body:** {“gameId”: 0, “player”: {“id”: 0, “authToken”: xyz}}
    - **Validation Checks:**
        - Game exists
        - Called by host
        - At least 2 players
        - Game is in “lobby” state
    - **State Updates:**
        - Shuffle deck
        - Deal 7 cards to each player
        - Set first card on discard pile
        - Game.state = “playing”
        - Game.current_player = player 1
    - **Success Response:** 202 Accepted
    - **Error Cases:**
        - 403 Forbidden → player other than the host tries to start the game
        - 404 Game Not Found → game id invalid
        - 409 Game Already Started
    - **Socket IO Events:**
        - Game:state:update (public)
        - Game:hand:update (private per player)
- **Endpoint 4**
    - **Name:** Get Game State
    - **HTTP Method & Route:** GET /api/games/:gameId
    - **Purpose:** Returns public game state + private state for this player only.
    - **Authorization:**
        - Make sure person requesting is a player in game
    - **Body:**
    - **Validation Checks:**
        - Make sure game exists
        - Make sure player is signed in + part of this game
    - **State Updates:**
    - **Success Response:** 200 OK
    - **Error Cases:**

        - 401 Unauthorized
        - 403 Forbidden - Not a Player in this Game
        - 404 Not Found
    - **Socket IO Events:** None
- **Endpoint 5**
    - **Name:** Play card
    - **HTTP Method & Route:** POST /api/games/:gameId/play-card
    - **Purpose:** Player plays a card from their hand onto the discard pile
    - **Authorization:** Must be a current player
    - **Body:** {“card_id”: 52, “chosenColor”:”blue”}
    - **Validation Checks:**
        - Game exists
        - Game is in “playing” state
        - User is current player
        - User owns the card (exists in playerHand)
        - Card is legal according to UNO rules:
            - Color matches OR value matches OR card is Wild
        - If card is Wild/Wild+4 = chosenColor present
        - If Wild+4 = player has no matching color cards
        - Pending penalties handled (eg. stack effects) (optional)
    - **State Updates:**
        - Remove card from player’s hand
        - Add card to discard pile
        - Apply effects (skip, reverse, draw 2, draw 4)
        - Advance to next player depending on game rules
        - If player’s hand empty - game.state = “ended”
    - **Success Response:** 202 Accepted
    - **Error Cases:**
        - 400 - illegal card play
        - 403 - not your turn
        - 404 - card not found
        - 409 - game state changed
    - **Socket IO Events:**
        - Game:state:update (public - updated turn + discard)
        - Game:hand:update (private - new hand for player)
        - Game:ended (if hand empty)
- **Endpoint 6**
    - **Name:** Draw Card
    - **HTTP Method & Route:** POST /api/games/:gameId/draw
    - **Purpose:** Player draws a card from the deck
    - **Authorization:** Must be current player
    - **Body:** None
    - **Validation Checks:**
        - Game exists

        - Player's turn
        - Game in "playing"
        - Deck has cards
        - Player is not under a mandatory penalty draw (skip if needed)
    - **State Updates:**
        - Give 1 card to player
        - If no playable cards - auto end turn
    - **Success Response:** 202 Accepted
    - **Error Cases:**
        - 403 - Not your turn
        - 409 - Deck empty (must reshuffle)
    - **Socket IO Events:**
        - Game:hand:update (private)
        - Game:state:update (public)
- **Endpoint 7**
    - **Name:** End Turn
    - **HTTP Method & Route:** POST /api/games/:gameId/end-turn
    - **Purpose:** Player manually ends their turn (only allowed after drawing a card)
    - **Authorization:** must be the current player's turn
    - **Body:**
    - **Validation Checks:**
        - Player has drawn at least once this turn
        - Player has no playable cards
        - Game is active
    - **State Updates:** Move turn to next player
    - **Success Response:** 202 Accepted
    - **Error Cases:**
        - 401 invalid token (player not valid to keep playing)
        - 403 Not a Player in this game || Not the current player
    - **Socket IO Events:** game:turn:changed

List of All Socket IO Events

Public:

1. **Game:start**
    a. Trigger:
        i. Host starts the game
    b. Data:
        i. Turn order decided
        ii. Top Discard card decided
2. **Game:player:joined**

    a. Trigger:
        i. A player enters a lobby
    b. Data:
        i. updated list of participants
        ii. Player ID of new player
3. **Game:player:left**
    a. Trigger:
        i. A player leaves/disconnects from lobby
    b. Data:
        i. Player ID of leaving player
        ii. updated list of participants
4. **Game:state:update**
    a. Trigger:
        i. Any public game updates
    b. Data:
        i. Current player
        ii. top card from discarded
        iii. card count of each player
        iv. Game status ?
5. **Game:card:played**
    a. Trigger: A player successfully plays a card
    b. Data:

```
{
  "game_id": 0,
  "player_id": 0,
 "card": { "color": "", "value": "" },
  "discard_pile_top": { "color": "", "value": "" },
  "next_player_id": 0
}
```

6. **Game:turn:changed**
    a. Trigger:
        i. Current player becomes the next player
    b. Data:
        i. Current player ID
        ii. Next Player ID
7. **Game:uno:called**
    a. Trigger:
        i. A player calls UNO with one card remaining
    b. Data:
        i. player ID
8. **Game:ended**
    a. Trigger:
        i. someone wins
    b. Data:

i. Player ID of winner

Private:

1. **Game:hand:update**
    a. Trigger:
        i. drawing cards
        ii. playing cards
        iii. Penalty actions-skip,plus 4, ect
    b. Data:
        i. Players list/array/deck of cards

2. **Game:draw:result**
    a. Trigger:
        i. draw request processed
    b. Data:
        i. New Cards that player will receive
        ii. Players updated list/array/deck of cards

3. **Game:error**
    a. Trigger:
        i. invalid move
    b. Data:
        i. Error message
        ii. Try again message