1. Game Description

Our UNO Web Game is a real-time multiplayer card game where authenticated users can create or join game rooms and play a fully server-controlled UNO match. The server acts as the single source of truth, enforcing all game rules such as turn order, card legality, and UNO calls. All players receive updates via Socket.io events, ensuring synchronized gameplay while protecting hidden (private) player information.

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 UNO game lobby and assigns the requesting user as the host.
4. Authorization	Must be logged in.
5. Request Body	`{ "max_players": 4 }`
6. Validation Checks	- User authenticated - max_players between 2–8
7. State Updates	Insert into `games` and `game_participants`, generate `room_code`
8. Success Response	`201 Created` + `{ "game_id": 123, "room_code": "ABCD12" }`
9. Error Cases	401, 400
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 user to existing lobby.
4. Authorization	Logged in.
5. Request Body	`{ "room_code": "ABCD12" }`
6. Validation Checks	- Game exists - Status is waiting - Seats available
7. State Updates	Insert into `game_participants`
8. Success Response	`202 Accepted`
9. Error Cases	404, 403
10. Socket.io Events	`game:player:joined`, `game:joined:ack`

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 waiting → playing, deals cards.
4. Authorization	Must be host.
5. Request Body	none

6. Validation Checks	- Host match - Enough players
7. State Updates	- Deal cards - Shuffle deck - Set current_player
8. Success Response	`202 Accepted`
9. Error Cases	403, 409
10. Socket.io Events	state update, hand update, 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 public state + private hand.
4. Authorization	Must be player in game.
5. Request Body	none
6. Validation Checks	Game exists, user in game
7. State Updates	none
8. Success Response	`200 OK`
9. Error Cases	403, 404
10. Socket.io Events	none

Endpoint – List Joinable Games

Field	Description
1. Endpoint Name	List Games

2. HTTP Method & Route	`GET /api/games`
3. Purpose	Lists games in lobby.
4. Authorization	optional
5. Request Body	none
6. Validation Checks	none
7. State Updates	none
8. Success Response	`200 OK`
9. Error Cases	none
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	Player draws card(s).
4. Authorization	Logged in + must be turn (unless penalty).
5. Request Body	`{ "reason": "normal" }`
6. Validation Checks	Game active, user in game, deck available
7. State Updates	Add card to hand
8. Success Response	`202 Accepted`
9. Error Cases	403, 409
10. Socket.io Events	`game:hand:update`, `game:state:update`

Endpoint – Play Card

Field	Description
1. Endpoint Name	Play Card
2. HTTP Method & Route	`POST /api/games/:game_id/play-card`
3. Purpose	Plays card + applies UNO effects.
4. Authorization	Must be active player.
5. Request Body	`{ "card_id": 201, "chosen_color": "red" }`
6. Validation Checks	Correct turn, owns card, legal move
7. State Updates	Remove card, update discard, effects
8. Success Response	`202 Accepted`
9. Error Cases	400, 403
10. Socket.io Events	state update, turn changed, hand update, game ended

Endpoint – Call UNO

Field	Description
1. Endpoint Name	Call UNO
2. HTTP Method & Route	`POST /api/games/:game_id/call-uno`
3. Purpose	Marks UNO call to avoid penalty.
4. Authorization	Logged in + player in game.
5. Request Body	none
6. Validation Checks	Player has exactly 1 card

7. State Updates	Mark UNO call
8. Success Response	`202 Accepted`
9. Error Cases	400, 403
10. Socket.io Events	state update

3. Socket.io Events

Event – game:player:joined

Event Name: `game:player:joined`
Scope: All players
Trigger: A player joins
Data:

```
{
  "game_id": 123,
  "player": { "user_id": 5, "seat_number": 2 }
}
```

Event – game:state:update

Event Name: `game:state:update`
Scope: All players
Trigger: Public state changes
Data:

```
{
  "game_id": 123,
  "discard_top": { "color": "green", "value": "5" }
}
```

Event – game:turn:changed

Event Name: `game:turn:changed`
Scope: All players
Trigger: Turn order changes
Data:

```
{
  "current_player_id": 7
}
```

Event – game:hand:update

Event Name: `game:hand:update`
Scope: One player only
Trigger: Hand changes
Data:

```
{
  "hand": [ ... ]
}
```

Event – game:ended

Event Name: `game:ended`
Scope: All players
Trigger: Game is won
Data:

```
{
  "winner_id": 5
}
```