

# Feladat: Online Sakk (valós idejű, 2 játékos)

## 1) Technológiai keretek

- **Backend:** Node.js + Express
  - **View:** EJS
  - **Realtime:** Socket.IO
  - **Állapot tárolás:** memóriában (nincs DB)
  - **Kliens:** HTML + CSS + Vanilla JS
  - **Sakk logika:** saját implementáció (vagy opcionálisan chess.js – extra)
- 

## 2) Funkcionális követelmények

### 2.1 Belépés / lobby

- A főoldalon a játékos megadja:
    - `playerName` (kötelező, min. 3 karakter)
    - `roomCode` (kötelező; ha nem létezik, létrejön)
  - „Csatlakozás” gomb után:
    - a szerver a socketet belépteti a szobába
    - visszajelzi, hányadik játékos a szobában
  - A szoba kódja megjelenik a UI-on (megosztható)
- 

### 2.2 Szobaszabályok

- Egy szobában **max. 2 játékos** lehet.
  - Ha 3. játékos próbál belépni:
    - hibaüzenet: „A szoba megtelt.”
  - Ha 1 játékos van:
    - UI: „Várakozás ellenfélre...”
  - Ha 2 játékos van:
    - a játék automatikusan elindul
- 

### 2.3 Játék indítása

- A szerver kiosztja a színeket:
  - első játékos: **Fehér**
  - második játékos: **Fekete**
- Kezdőállás:
  - standard sakk kezdőpozíció (8×8 tábla)

- **Fehér kezd**
- 

## 2.4 Lépések kezelése

- A kliens:
    - egy figurát kiválaszt
    - majd egy célmezőt
  - Lépés csak akkor érvényes, ha:
    - a játék fut
    - a lépő játékos köre van
    - a figura a játékosé
    - a lépés sakk szabályai szerint legális
    - nem hagyja sakkban a saját királyt
  - Érvényes lépés után a szerver:
    - frissíti a táblát
    - váltja a kört
    - broadcastolja az új állapotot a szobába
- 

## 2.5 Speciális sakk szabályok

Kezelendő szabályok:

- ✓ ütés
  - ✓ gyalog első lépése (2 mező)
  - ✓ átalakulás (promotion – alapértelmezett: vezér)
  - ✓ sáncolás
  - ✓ en passant (opcionális – extra pont)
  - ✓ sakk / matt
  - ✓ patt (döntetlen)
- 

## 2.6 Játék vége

A szerver minden lépés után ellenőrzi:

- sakk-matt
- patt
- döntetlen:
  - nincs legális lépés
  - opcionálisan: 50 lépés szabály (extra)

Játék vége esetén:

- státusz üzenet:

- „Fehér nyert (matt)”
    - „Fekete nyert (matt)”
    - „Döntetlen (patt)”
  - a tábla lezár
- 

## 2.7 Új játék (Rematch)

- „Új játék” gomb:
    - csak akkor aktív, ha 2 játékos van bent
  - Bármelyik játékos kérheti
  - Új játék **csak akkor indul**, ha mindkét fél rákattint
    - simple **ready / rematch** mechanizmus
  - Új játék esetén:
    - tábla reset
    - színek felcserélődnek (extra UX)
- 

## 2.8 Kilépés / disconnect

- Ha egy játékos lecsatlakozik:
    - a másik játékos üzenetet kap:
      - „Az ellenfél kilépett.”
    - a játék megszakad
    - a szoba állapota resetelődik
  - Új belépővel a játék újraindítható
- 

## 3) Nem-funkcionális követelmények

- Áttekinthető UI:
  - 8×8 sakk tábla
  - világos / sötét mezők
  - kijelölt figura és lehetséges lépések
- Státusz sáv:
  - aktuális kör
  - sakk / matt jelzés
- Hibakezelés:
  - invalid lépés → kliens oldali tiltás + szerver oldali védelem
- Kódstruktúra:
- `server.js`
- `/views`
- `index.ejs`
- `room.ejs`
- `/public`
- `style.css`
- `chess.js`

- `socket.js`
- 

## 4) Socket.IO események (konkrét specifikáció)

### Kliens → Szerver

- `joinRoom`
  - `{ "playerName": string, "roomCode": string }`
  - `makeMove`
  - `{ "roomCode": string, "from": "e2", "to": "e4" }`
  - `requestRematch`
  - `{ "roomCode": string }`
  - `leaveRoom`
  - `{ "roomCode": string }`
- 

### Szerver → Kliens

- `roomJoined`
  - `{`
  - `"roomCode": string,`
  - `"color": "white" | "black",`
  - `"players": [{ "name": string, "color": string }],`
  - `"status": "waiting" | "playing"`
  - `}`
  - `stateUpdate`
  - `{`
  - `"board": string[][],`
  - `"turn": "white" | "black",`
  - `"status": "playing" | "finished",`
  - `"check": boolean,`
  - `"winner": "white" | "black" | null`
  - `}`
  - `errorMessage`
  - `{ "message": string }`
  - `opponentLeft`
  - `{ "message": string }`
  - `rematchStatus`
  - `{ "readyCount": 0 | 1 | 2 }`
- 

## 5) Állapotmodell (memóriában)

Szobánként tárolandó:

- `players:`
  - `socketId`
  - `name`
  - `color`

- board:
    - 8×8 mátrix (pl. "wp", "bk", "")
  - turn: "white" | "black"
  - status: "waiting" | "playing" | "finished"
  - check: boolean
  - winner: "white" | "black" | null
  - castlingRights
  - enPasantTarget
  - rematchReady: Set<socketId>
- 

## 6) Kész definíció (Done kritérium)

A feladat kész, ha:

- két külön böngészőből játszható ugyanazzal a roomCode-dal
  - a sakk tábla valós időben frissül
  - szabálytalan lépés nem hajtható végre
  - sakk, matt és patt korrektül felismerésre kerül
  - rematch csak kétoldali jóváhagyással indul
  - kilépéskor a másik fél értesül
- 

## 7) Pontozás (100 pont)

### A) Alap működés (25p)

- (10p) Room join + színek kiosztása
- (10p) Valós idejű frissítés
- (5p) Max 2 játékos / szoba

### B) Sakk logika (45p)

- (20p) Alap lépések helyesek
- (10p) Sakk / matt felismerés
- (10p) Patt / döntetlen
- (5p) Speciális szabályok (sánc, promotion)

### C) Rematch (10p)

- (10p) Ready mechanizmus

### D) Disconnect kezelés (10p)

- (10p) Kilépés korrekt kezelése

### **E) Kódminőség (10p)**

- (5p) Strukturált kód
  - (5p) Kommentek, tiszta event-kezelés
- 

### **Extra (max +10p)**

- (+5p) Spectator mode
- (+5p) Chess.js használata + validáció