COLOCVIU REACT – PROIECT „SUDOKU”

Se cere realizarea unui joc de Sudoku în React, alcătuit din componente organizate logic și ușor de întreținut. Rezolvă fiecare task în ordinea prezentată, respectând cerințele de bază și folosind React pentru gestionarea interfeței și a stării aplicației.

FIECARE TASK ARE 1 PUNCT. SE ACORDĂ 2 PUNCTE DIN OFICIU!

TASK 1: CREAREA COMPONENTEI “SUDOKUBOARD”

- Proiectează un layout pentru un **board Sudoku 9×9**.
- Structura minimă: fiecare rând conține 9 celule. Poți crea un component dedicat `SudokuBoard` care afișează aceste celule.
- Stilizează grila astfel încât subgrid-urile 3×3 să fie vizibile (ex. prin borduri mai groase).

Cerință: Prezintă pe ecran componentele astfel încât să existe 81 de celule. Nu este necesară logica încă; doar un layout coerent și lizibil.

TASK 2: DEFINIREA STĂRII JOCULUI (BOARD DATA)

- Creează o reprezentare internă a tablei de Sudoku (ex. un array bidimensional de cifre).
- Alege un puzzle inițial (9×9) cu valori (0 = liber, 1-9 = cifre).
- Depozitează structura în `useState` într-un component “container” (de ex. `Game` sau chiar `App`), astfel încât să poată fi transmisă mai departe către `SudokuBoard`.

Cerință: Demonstrează că datele din `useState` se propagă corect în `SudokuBoard` și fiecare celulă afișează valoarea corespunzătoare (0 sau alt simbol pentru spații libere).

TASK 3: COMPONENTELE "CELL" ȘI AFIȘAREA DATELOR

- Împarte fiecare celulă (din grila 9×9) într-un component "Cell" individual.
- Fiecare `Cell` primește valoarea (de ex. `0` sau `5` etc.) și o afișează în mod corespunzător.
- Stilizează celulele (dimensiune, margini) pentru claritate.

Cerință: Fiecare cell are propriul component și afișează **valoarea** din starea principală. Verifică vizual că board-ul este identic cu datele inițiale.

TASK 4: EDITAREA CELULELOR

- Permite utilizatorului să modifice conținutul unei celule.
- Poți alege între:
  - O casetă de tip `<input>` pentru introducerea cifrelor,
  - Sau o funcție onClick + un mic meniu cu cifre (1-9).
- La fiecare modificare, actualizează starea din componenta principală (array-ul bidimensional).

Cerință: Când utilizatorul introduce o cifră validă, board-ul să reflecte imediat noua valoare.

TASK 5: VALIDARE MINIMĂ LA INTRODUCERE

- Impune regula ca doar cifre între 1 și 9 să fie acceptate ca intrare (fără litere, zero sau alte caractere).
- Dacă se introduce ceva invalid, ignoră sau semnalează eroarea (decizi tu cum).
- Deocamdată, nu te preocupă conflictele pe rând/coloană, ci doar validarea de bază.

Cerință: Testează că atunci când introduci text invalid, nu se afectează starea finală a tablei.

TASK 6: DETECTAREA CONFLICTELOR (OPȚIONAL, RECOMANDAT)

- După fiecare editare, verifică dacă cifra introdusă apare deja pe același rând, coloană sau în subgrid-ul 3×3.
- Dacă e un conflict, evidențiază celula (de ex. fundal roșu) sau afișează un mesaj.
- Poți decide dacă permiți totuși introducerea conflictuală sau o blochezi.

Cerință: La introducerea unei valori ce există deja pe linie/coloană/subgrid, semnalizează clar conflictul.

TASK 7: VERIFICAREA CONDIȚIEI DE FINAL (PUZZLE COMPLET)

- Creează un buton "Check" sau verifică automat dacă:
    1. Toate celulele sunt umplute (niciun 0).
    2. Nu există conflicte de rând/coloană/subgrid.
- Dacă e complet și valid, afișează un mesaj ("Felicitări, Sudoku complet!").
- (Opțional) Dezactivează editarea sau lasă posibilitatea de modificare.

Cerință: Asigură-te că la un Sudoku corect, apare mesajul de succes, iar la unul incomplet sau conflictual, testul eșuează.

TASK 8: RESET ȘI/ SAU PUZZLE NOU

- Adaugă un buton "Reset" care reinițializează datele cu puzzle-ul inițial.
- (Opțional) Afișează un puzzle aleator (dacă ai o listă predefinită de puzzle-uri) sau oferă utilizatorului mai multe variante.

Cerință: Demonstrează că jocul poate fi refăcut rapid și că datele revin la forma inițială (sau se încarcă un alt puzzle).

OBSERVAȚII FINALE

- **Organizează fișierele** logic (componente în fișiere separate).
- **Predă proiectul** cu instrucțiuni de rulare și un scurt README, explicând cum ai structurat aplicația.
- Respectă principiile React (state management, props, componentizare).

Succes!