**CHỈNH SỬA BẢN CHƠI 1 MÌNH**

# FILE INDEX.HTML

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Sudoku</title>

    <link rel="stylesheet" href="{{ url\_for('static', filename='css/style.css') }}">

</head>

<body>

    <div class="container">

        <h1>Sudoku</h1>

        <div class="menu">

            <button onclick="window.location.href='/practice'">Chơi Offline</button>

            <button onclick="window.location.href='/singleplayer'">Chơi Online</button>

        </div>

    </div>

</body>

</html>

# FILE PRACTICE.HTML

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Sudoku Practice</title>

    <link rel="stylesheet" href="{{ url\_for('static', filename='css/style.css') }}">

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css">

    <script src="{{ url\_for('static', filename='js/script.js') }}"></script>

</head>

<body>

    <div class="container">

        <div class="header">

            <h1>Sudoku</h1>

            <div class="level">

                Độ khó: <span id="difficulty"></span>

                <span id="stars"></span>

            </div>

            <div class="timer">00:00</div>

        </div>

        <div id="difficulty-menu" class="menu">

            <button onclick="startGame('easy')">Dễ</button>

            <button onclick="startGame('medium')">Trung bình</button>

            <button onclick="startGame('hard')">Khó</button>

        </div>

        <table id="game-board"></table>

        <div class="toolbar">

            <button onclick="eraseCell()"><i class="fas fa-eraser"></i></button>

            <button onclick="undo()">Hoàn tác</button>

        </div>

        <div id="result-modal" class="modal">

            <div class="modal-content">

                <span class="close" onclick="closeModal()">&times;</span>

                <h2 id="result-title"></h2>

                <p id="result-message"></p>

                <button onclick="closeModal()">Chơi lại</button>

            </div>

        </div>

    </div>

</body>

</html>

# FILE SINGLEPALYER.HTML

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Sudoku</title>

    <link rel="stylesheet" href="{{ url\_for('static', filename='css/style.css') }}">

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">

    <style>

        #game-area {

            display: none; /\* Ẩn mặc định, chỉ hiển thị sau khi tải xong \*/

        }

    </style>

</head>

<body>

    <div class="container">

        <div class="header">

            <h1>Sudoku</h1>

            <div class="info">

                <div class="level"></div>

                <div class="share"></div>

            </div>

            <div class="timer">00:00</div>

        </div>

        <div id="difficulty-menu" class="menu">

            <button onclick="startGame('easy')">Dễ</button>

            <button onclick="startGame('medium')">Trung bình</button>

            <button onclick="startGame('hard')">Khó</button>

        </div>

        <div id="game-area">  <table id="game-board"></table>

        </div>

        <div class="toolbar">

            <button onclick="eraseCell()"><i class="fas fa-eraser"></i></button>

            <button onclick="undo()">Hoàn tác</button>

        </div>

        <div id="result-modal" class="modal">

            <div class="modal-content">

                <span class="close" onclick="closeModal()">&times;</span>

                <h2 id="result-title"></h2>

                <p id="result-message"></p>

                <button onclick="closeModal()">Chơi lại</button>

            </div>

        </div>

        <div id="waiting" style="display: none;">Đang tải trò chơi mới...</div>

    </div>

    <script src="{{ url\_for('static', filename='js/online.js') }}"></script>

    <script>

        document.addEventListener('DOMContentLoaded', () => {

            document.getElementById('difficulty-menu').style.display = 'block';

        });

    </script>

</body>

</html>

# FILE ONLINE.JS

let board;

let solution;

let timerInterval;

let startTime;

let incorrectAttempts = 0;

const MAX\_INCORRECT\_ATTEMPTS = 3;

let currentGameLevel = 'easy'; // Mức độ hiện tại của ván chơi

let history = []; // Mảng lưu trữ các bước đi

function initializeBoard() {

    const gameBoard = document.getElementById('game-board');

    gameBoard.innerHTML = '';

    const table = document.createElement('table');

    for (let i = 0; i < 9; i++) {

        const row = document.createElement('tr');

        for (let j = 0; j < 9; j++) {

            const cell = document.createElement('td');

            const input = document.createElement('input');

            input.type = 'text';

            input.classList.add('cell');

            input.maxLength = 1;

            input.dataset.row = i;

            input.dataset.col = j;

            if (board[i][j] !== 0) {

                input.value = board[i][j];

                input.disabled = true;

            } else {

                input.addEventListener('input', handleInput);

                input.addEventListener('focus', () => highlightRelated(i, j));

                input.addEventListener('blur', clearHighlight);

            }

            cell.appendChild(input);

            row.appendChild(cell);

        }

        table.appendChild(row);

    }

    gameBoard.appendChild(table);

    displayLevelInfo(currentGameLevel);

    startTimer();

}

function handleInput(event) {

    const input = event.target;

    const row = parseInt(input.dataset.row);

    const col = parseInt(input.dataset.col);

    let value = input.value ? parseInt(input.value) : 0;

    if (isNaN(value) || value < 1 || value > 9) {

        input.value = '';

        return;

    }

    if (isValidMove(row, col, value)) {

        history.push({ row, col, prevValue: board[row][col], newValue: value }); // Lưu vào history

        board[row][col] = value;

        input.classList.remove('error');

        if (isGameOver()) {

            stopTimer();

            showModal("Bạn đã thắng!", "Chúc mừng!");

        }

    } else {

        input.classList.add('error');

        incorrectAttempts++;

        if (incorrectAttempts >= MAX\_INCORRECT\_ATTEMPTS) {

            stopTimer();

            showModal("Hết lượt", "Bạn đã nhập sai quá nhiều lần.");

        }

    }

}

function isValidMove(row, col, value) {

    // Kiểm tra hàng

    if (board[row].includes(value) && board[row][col] !== value) {

        return false;

    }

    // Kiểm tra cột

    for (let i = 0; i < 9; i++) {

        if (board[i][col] === value && i !== row) {

            return false;

        }

    }

    // Kiểm tra ô 3x3

    const startRow = Math.floor(row / 3) \* 3;

    const startCol = Math.floor(col / 3) \* 3;

    for (let i = 0; i < 3; i++) {

        for (let j = 0; j < 3; j++) {

            const r = startRow + i;

            const c = startCol + j;

            if (board[r][c] === value && (r !== row || c !== col)) {

                return false;

            }

        }

    }

    return true;

}

function isGameOver() {

    for (let i = 0; i < 9; i++) {

        for (let j = 0; j < 9; j++) {

            if (board[i][j] === 0) {

                return false;

            }

        }

    }

    return true;

}

function clearError(row, col) {

    const input = document.querySelector(`input[data-row="${row}"][data-col="${col}"]`);

    if (input) {

        input.classList.remove('error');

    }

}

function startTimer() {

    startTime = new Date();

    const timerElement = document.querySelector('.timer');

    timerInterval = setInterval(() => {

        const elapsedTime = Math.floor((new Date() - startTime) / 1000);

        timerElement.textContent = formatTime(elapsedTime);

    }, 1000);

}

function stopTimer() {

    clearInterval(timerInterval);

}

function formatTime(time) {

    const minutes = Math.floor(time / 60).toString().padStart(2, '0');

    const seconds = (time % 60).toString().padStart(2, '0');

    return `${minutes}:${seconds}`;

}

function showModal(title, message) {

    document.getElementById('result-title').textContent = title;

    document.getElementById('result-message').textContent = message;

    document.getElementById('result-modal').style.display = 'block';

}

function closeModal() {

    document.getElementById('result-modal').style.display = 'none';

    startGame(currentGameLevel); // Bắt đầu lại game với mức độ hiện tại

}

function eraseCell() {

    const focusedElement = document.activeElement;

    if (focusedElement.tagName === 'INPUT' && !focusedElement.disabled) {

        const row = parseInt(focusedElement.dataset.row);

        const col = parseInt(focusedElement.dataset.col);

        history.push({ row, col, prevValue: board[row][col], newValue: '' }); // Lưu vào history

        board[row][col] = 0;

        focusedElement.value = '';

        clearError(row, col);

    }

}

function undo() {

    if (history.length > 0) {

        const lastMove = history.pop();

        const input = document.querySelector(`input[data-row="${lastMove.row}"][data-col="${lastMove.col}"]`);

        if (input) {

            input.value = lastMove.prevValue !== '' ? lastMove.prevValue : '';

            board[lastMove.row][lastMove.col] = lastMove.prevValue;

            input.classList.remove('error');

        }

    } else {

        alert("Không còn bước nào để hoàn tác.");

    }

}

function highlightRelated(row, col) {

    clearHighlight();

    const table = document.querySelector('table');

    const cells = table.querySelectorAll('input.cell');

    const selectedValue = parseInt(cells[row \* 9 + col].value) || 0;

    cells[row \* 9 + col].style.border = '2px solid #007bff';

    cells.forEach((cell, index) => {

        const cellRow = Math.floor(index / 9);

        const cellCol = index % 9;

        const cellValue = parseInt(cell.value) || 0;

        if (cellValue === selectedValue && selectedValue !== 0) {

            cell.classList.add('highlighted');

        }

        if (cellRow === row || cellCol === col) {

            cell.classList.add('highlighted-row-col');

        }

    });

}

function clearHighlight() {

    const cells = document.querySelectorAll('input.cell');

    cells.forEach(cell => {

        cell.classList.remove('highlighted');

        cell.classList.remove('highlighted-row-col');

        cell.style.border = '';

    });

}

function startGame(level) {

    currentGameLevel = level;

    history = [];

    document.getElementById('waiting').style.display = 'block';

    document.getElementById('game-area').style.display = 'none'; // Ẩn bảng

    document.getElementById('difficulty-menu').style.display = 'none';

    displayLevelInfo(level);

    // GỌI API lấy bảng Sudoku mới

    fetch(`/generate\_sudoku?level=${level}`)

        .then(response => {

            if (!response.ok) {

                throw new Error('Network response was not ok');

            }

            return response.json();

        })

        .then(data => {

            board = data.board;

            solution = data.solution;

            initializeBoard();

            startTimer();

            incorrectAttempts = 0;

            document.getElementById('waiting').style.display = 'none';

            document.getElementById('game-area').style.display = 'block'; // Hiển thị bảng

        })

        .catch(error => {

            console.error('Lỗi khi tải bảng mới:', error);

            alert('Không thể tải bảng Sudoku mới. Vui lòng thử lại sau.');

            document.getElementById('waiting').style.display = 'none';

        });

}

function displayLevelInfo(level) {

    const difficultyElement = document.querySelector('.level');

    const starsElement = document.querySelector('.share');

    if (difficultyElement && starsElement) {

        difficultyElement.textContent = level.charAt(0).toUpperCase() + level.slice(1);

        let stars = '';

        for (let i = 0; i < (level === 'easy' ? 1 : level === 'medium' ? 2 : 3); i++) {

            stars += ' ★ ';

        }

        starsElement.textContent = stars;

    }

}

# FILE SCRIPT.JS

let board = [];

let solution = [];

let timerInterval;

let startTime;

let incorrectAttempts = 0;

const MAX\_INCORRECT\_ATTEMPTS = 3;

let currentLevel = 'easy'; // Mức độ mặc định

let history = []; // Mảng lưu trữ các bước đi để hoàn tác

// document.addEventListener('DOMContentLoaded', () => {  // Comment hoặc xóa dòng này

//     startGame('easy'); // Bắt đầu với mức độ dễ khi trang tải xong

// });

function startGame(level) {

    currentLevel = level;

    incorrectAttempts = 0;

    history = []; // Reset lịch sử khi bắt đầu game mới

    document.getElementById('difficulty-menu').style.display = 'none'; // Ẩn menu độ khó

    document.getElementById('game-board').innerHTML = ''; // Xóa bảng cũ

    fetch('/generate\_sudoku?level=' + level) // Gọi API để tạo bảng

        .then(response => response.json())

        .then(data => {

            board = data.board;

            solution = data.solution;

            initializeBoard(); // Khởi tạo bảng trên trang

            startTimer(); // Bắt đầu đếm giờ

            displayLevelInfo(level); // Hiển thị thông tin độ khó

        });

}

function startGame(level) {

    currentLevel = level;

    incorrectAttempts = 0;

    history = []; // Reset lịch sử khi bắt đầu game mới

    document.getElementById('difficulty-menu').style.display = 'none'; // Ẩn menu độ khó

    document.getElementById('game-board').innerHTML = ''; // Xóa bảng cũ

    fetch('/generate\_sudoku?level=' + level) // Gọi API để tạo bảng

        .then(response => response.json())

        .then(data => {

            board = data.board;

            solution = data.solution;

            initializeBoard(); // Khởi tạo bảng trên trang

            startTimer(); // Bắt đầu đếm giờ

            displayLevelInfo(level); // Hiển thị thông tin độ khó

        });

}

function initializeBoard() {

    const gameBoard = document.getElementById('game-board');

    const table = document.createElement('table');

    for (let i = 0; i < 9; i++) {

        const row = document.createElement('tr');

        for (let j = 0; j < 9; j++) {

            const cell = document.createElement('td');

            const input = document.createElement('input');

            input.type = 'text';

            input.classList.add('cell');

            input.maxLength = 1; // Chỉ cho phép nhập 1 ký tự

            input.dataset.row = i; // Lưu chỉ số hàng

            input.dataset.col = j; // Lưu chỉ số cột

            if (board[i][j] !== 0) {

                input.value = board[i][j];

                input.disabled = true; // Ô số ban đầu không cho sửa

            } else {

                input.addEventListener('input', handleInput); // Xử lý khi nhập

                input.addEventListener('focus', () => highlightRelated(i, j)); // Highlight khi chọn ô

                input.addEventListener('blur', clearHighlight); // Xóa highlight khi bỏ chọn ô

            }

            cell.appendChild(input);

            row.appendChild(cell);

        }

        table.appendChild(row);

    }

    gameBoard.appendChild(table);

}

function handleInput(event) {

    const input = event.target;

    const row = parseInt(input.dataset.row);

    const col = parseInt(input.dataset.col);

    let value = input.value ? parseInt(input.value) : 0; // Chuyển đổi sang số

    // Kiểm tra đầu vào

    if (isNaN(value) || value < 0 || value > 9) {

        input.value = ''; // Xóa giá trị không hợp lệ

        return;

    }

    // Lưu bước đi vào lịch sử

    history.push({ row, col, prevValue: board[row][col], newValue: value });

    board[row][col] = value; // Cập nhật bảng

    if (solution[row][col] === value) {

        input.classList.remove('error'); // Xóa hiệu ứng lỗi

        if (checkWinCondition()) {

            stopTimer();

            showModal('Chúc mừng!', `Bạn đã hoàn thành trong ${document.querySelector('.timer').textContent}`);

        }

    } else {

        input.classList.add('error'); // Thêm hiệu ứng lỗi

        incorrectAttempts++;

        if (incorrectAttempts >= MAX\_INCORRECT\_ATTEMPTS) {

            stopTimer();

            showModal('Game Over', 'Bạn đã nhập sai quá 3 lần.');

        }

    }

}

function checkWinCondition() {

    for (let i = 0; i < 9; i++) {

        for (let j = 0; j < 9; j++) {

            if (board[i][j] !== solution[i][j]) {

                return false; // Chưa hoàn thành

            }

        }

    }

    return true; // Đã hoàn thành

}

function startTimer() {

    startTime = new Date();

    const timerElement = document.querySelector('.timer');

    timerInterval = setInterval(() => {

        const elapsedTime = Math.floor((new Date() - startTime) / 1000);

        timerElement.textContent = formatTime(elapsedTime);

    }, 1000);

}

function stopTimer() {

    clearInterval(timerInterval);

}

function formatTime(time) {

    const minutes = Math.floor(time / 60).toString().padStart(2, '0');

    const seconds = (time % 60).toString().padStart(2, '0');

    return `${minutes}:${seconds}`;

}

function showModal(title, message) {

    document.getElementById('result-title').textContent = title;

    document.getElementById('result-message').textContent = message;

    document.getElementById('result-modal').style.display = 'block';

}

function closeModal() {

    document.getElementById('result-modal').style.display = 'none';

}

function eraseCell() {

    const focusedElement = document.activeElement;

    if (focusedElement.tagName === 'INPUT' && !focusedElement.disabled) {

        const row = parseInt(focusedElement.dataset.row);

        const col = parseInt(focusedElement.dataset.col);

        const prevValue = board[row][col]; // Lưu giá trị trước khi xóa

        focusedElement.value = '';

        board[row][col] = 0;

        history.push({ row, col, prevValue: prevValue, newValue: 0 }); // Lưu vào lịch sử

        focusedElement.classList.remove('error'); // Xóa hiệu ứng lỗi

    }

}

function undo() {

    if (history.length > 0) {

        const lastMove = history.pop();

        const input = document.querySelector(`input[data-row="${lastMove.row}"][data-col="${lastMove.col}"]`);

        if (input) {

            input.value = lastMove.prevValue !== 0 ? lastMove.prevValue : '';

            board[lastMove.row][lastMove.col] = lastMove.prevValue;

            input.classList.remove('error'); // Xóa hiệu ứng lỗi

        }

    } else {

        alert('Không còn bước nào để hoàn tác.');

    }

}

function highlightRelated(row, col) {

    clearHighlight(); // Xóa highlight cũ

    const table = document.querySelector('table');

    const cells = table.querySelectorAll('input.cell');

    const selectedValue = parseInt(cells[row \* 9 + col].value) || 0;

    // Highlight ô được chọn

    cells[row \* 9 + col].style.border = '2px solid #007bff';

    cells.forEach((cell, index) => {

        const cellRow = Math.floor(index / 9);

        const cellCol = index % 9;

        const cellValue = parseInt(cell.value) || 0;

        if (cellValue === selectedValue && selectedValue !== 0) {

            cell.classList.add('highlighted'); // Highlight ô cùng số

        }

        if (cellRow === row || cellCol === col) {

            cell.classList.add('highlighted-row-col'); // Highlight hàng và cột

        }

    });

}

function clearHighlight() {

    const cells = document.querySelectorAll('input.cell');

    cells.forEach(cell => {

        cell.classList.remove('highlighted');

        cell.classList.remove('highlighted-row-col');

        cell.style.border = ''; // Reset viền

    });

}

function displayLevelInfo(level) {

    const difficultyElement = document.getElementById('difficulty');

    const starsElement = document.getElementById('stars');

    difficultyElement.textContent = level.charAt(0).toUpperCase() + level.slice(1); // Viết hoa chữ cái đầu

    let stars = '';

    for (let i = 0; i < (level === 'easy' ? 1 : level === 'medium' ? 2 : 3); i++) {

        stars += '★';

    }

    starsElement.textContent = stars;

}

# FILE APP.PY

from flask import Flask, render\_template, jsonify, request

import random

app = Flask(\_\_name\_\_)

def is\_valid(board, row, col, num):

    """Kiểm tra tính hợp lệ của số trong bảng Sudoku."""

    for x in range(9):

        if board[row][x] == num and x != col:

            return False

    for x in range(9):

        if board[x][col] == num and x != row:

            return False

    start\_row, start\_col = 3 \* (row // 3), 3 \* (col // 3)

    for i in range(3):

        for j in range(3):

            if board[i + start\_row][j + start\_col] == num and (i + start\_row, j + start\_col) != (row, col):

                return False

    return True

def solve\_sudoku(board):

    """Giải bảng Sudoku (đệ quy, quay lui)."""

    for row in range(9):

        for col in range(9):

            if board[row][col] == 0:

                for num in range(1, 10):

                    if is\_valid(board, row, col, num):

                        board[row][col] = num

                        if solve\_sudoku(board):

                            return True

                        board[row][col] = 0

                return False

    return True

def generate\_full\_board():

    """Tạo một bảng Sudoku hoàn chỉnh."""

    board = [[0 for \_ in range(9)] for \_ in range(9)]

    solve\_sudoku(board)

    return board

def generate\_sudoku(level):

    """Tạo bảng Sudoku với độ khó khác nhau."""

    board = generate\_full\_board()

    solution\_board = [row[:] for row in board]

    game\_board = [row[:] for row in board]

    cells\_to\_remove = {'easy': 40, 'medium': 50, 'hard': 60}.get(level, 50)

    while cells\_to\_remove > 0:

        row = random.randint(0, 8)

        col = random.randint(0, 8)

        if game\_board[row][col] != 0:

            game\_board[row][col] = 0

            cells\_to\_remove -= 1

    return {'board': game\_board, 'solution': solution\_board}

@app.route('/')

def index():

    return render\_template('index.html')

@app.route('/practice')

def practice():

    return render\_template('practice.html')

@app.route('/singleplayer')

def singleplayer():

    return render\_template('singleplayer.html')

@app.route('/generate\_sudoku')

def generate\_sudoku\_api():

    level = request.args.get('level', 'easy')

    sudoku\_data = generate\_sudoku(level)

    return jsonify(sudoku\_data)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)