Of course! I understand. You want to transform the offline page into a mini-game arcade. Instead of launching directly into the Math Runner game, clicking the "Play" button should first show a menu with three options: the existing Math Runner, a classic Snake game, and a simple Nokia-style Cricket game. All of this should be contained within a single React component file.
I've integrated the new games and the game selection menu into your existing code structure. Both the Snake and Cricket games are fully functional and styled to match the aesthetic of your original component.
Here is the complete, single-file React component with all three games included:

```
import React, { useState, useEffect, useRef, useCallback } from
"react";
import { Wifi, WifiOff, Play, RotateCcw, Mic, ArrowLeft, Delete,
Check, Gamepad2, BrainCircuit, Tally1 } from "lucide-react";

// --- In-memory storage for high scores (persists during session
only) ---
let storedHighScores = {
  math: 0,
  snake: 0,
  cricket: 0,
};

//
======================================================================
======
// === HELPER FUNCTIONS & CONSTANTS
//
======================================================================
======

// --- Math Game Helper ---
function generateEquation(currentScore = 0) {
  let operations = ["+", "-", "×"];
  let maxNum = 20;
  let multMax = 12;
  if (currentScore >= 30) {
    operations.push("÷");
    maxNum = 50;
    multMax = 20;
  } else if (currentScore >= 20) {
    operations.push("÷");
    maxNum = 30;
    multMax = 15;
  } else if (currentScore >= 10) {
    maxNum = 35;
    multMax = 15;
  }
  const operation = operations[Math.floor(Math.random() *
operations.length)];
```

```
  let num1, num2, answer;
  if (operation === "÷") {
    const divisor = Math.floor(Math.random() * 10) + 2;
    const result = Math.floor(Math.random() * 12) + 1;
    num1 = divisor * result;
    num2 = divisor;
    answer = result;
  } else {
    if (operation === "×") {
      num1 = Math.floor(Math.random() * multMax) + 1;
      num2 = Math.floor(Math.random() * multMax) + 2;
    } else {
      num1 = Math.floor(Math.random() * maxNum) + 1;
      num2 = Math.floor(Math.random() * maxNum) + 1;
    }
    if (operation === "+") answer = num1 + num2;
    else if (operation === "-") {
      if (currentScore > 5 && Math.random() > 0.5) [num1, num2] =
[num2, num1];
      answer = num1 - num2;
    } else answer = num1 * num2;
  }
  const displayOperation = operation === "×" ? "×" : operation === "÷"
? "÷" : operation;
  return { num1, num2, operation: displayOperation, answer };
}

// --- Snake Game Constants ---
const GRID_SIZE = 20;
const INITIAL_SNAKE = [{ x: 10, y: 10 }];
const INITIAL_FOOD = { x: 15, y: 15 };
const DIRECTIONS = {
  ArrowUp: { x: 0, y: -1 },
  ArrowDown: { x: 0, y: 1 },
  ArrowLeft: { x: -1, y: 0 },
  ArrowRight: { x: 1, y: 0 },
};

//
======================================================================
======
// === GAME COMPONENTS
//
======================================================================
======

// --- 1. Math Runner Game Component (Your Original Code) ---
const MathRunner = ({ onBack }) => {
```

```
const [score, setScore] = useState(0);
const [equation, setEquation] = useState(() => generateEquation(0));
const [userAnswer, setUserAnswer] = useState("");
const [timeLeft, setTimeLeft] = useState(30);
const [isPlaying, setIsPlaying] = useState(false);
const [gameOver, setGameOver] = useState(false);
const [highScore, setHighScore] = useState(storedHighScores.math);
const [feedback, setFeedback] = useState("");

useEffect(() => {
  storedHighScores.math = highScore;
}, [highScore]);

useEffect(() => {
  if (isPlaying && timeLeft > 0) {
    const timer = setTimeout(() => setTimeLeft(timeLeft - 1), 1000);
    return () => clearTimeout(timer);
  } else if (timeLeft === 0) {
    endGame();
  }
}, [timeLeft, isPlaying]);

const startGame = () => {
  setScore(0);
  setTimeLeft(30);
  setGameOver(false);
  setIsPlaying(true);
  setEquation(generateEquation(0));
  setUserAnswer("");
  setFeedback("");
};

const endGame = () => {
  setIsPlaying(false);
  setGameOver(true);
  if (score > highScore) setHighScore(score);
};

const handleSubmit = (answer) => {
  if (!isPlaying) return;
  const numAnswer = parseInt(answer);
  if (isNaN(numAnswer)) return;
  if (numAnswer === equation.answer) {
    const newScore = score + 1;
    setScore(newScore);
    setFeedback("✓");
    setTimeout(() => {
      setEquation(generateEquation(newScore));
```

```
        setUserAnswer("");
        setFeedback("");
      }, 300);
    } else {
      setFeedback("✗");
      setTimeout(() => endGame(), 500);
    }
  };

  const handleNumberClick = (num) => {
    if (!isPlaying) return;
    setUserAnswer(userAnswer + num);
  };

  const handleNegativeToggle = () => {
    if (!isPlaying) return;
    if (userAnswer.startsWith("-"))
setUserAnswer(userAnswer.substring(1));
    else setUserAnswer("-" + userAnswer);
  };

  const handleClear = () => {
    setUserAnswer("");
    setFeedback("");
  };

  const handleEnter = () => {
    if (userAnswer && userAnswer !== "-") handleSubmit(userAnswer);
  };

  return (
    <div className="flex flex-col items-center justify-center w-full
min-h-screen bg-white p-4 relative">
        <button onClick={() => onBack(score, highScore)}
className="fixed top-4 left-4 flex items-center gap-2 text-gray-600
hover:text-black z-30">
            <ArrowLeft className="w-6 h-6" /> <span className="text-sm
font-medium">Back to Games</span>
        </button>
        {/* ... (rest of your MathRunner JSX is identical) ... */}
        <div className="w-full max-w-lg md:max-w-xl lg:max-w-2xl flex
flex-col items-center space-y-4 md:space-y-6">
            <div className="text-center mt-6 md:mt-8">
              <h1 className="text-3xl sm:text-4xl md:text-5xl
lg:text-6xl font-bold mb-1 flex justify-center items-center">RE<Mic
className="w-6 h-6 sm:w-8 sm:h-8 md:w-10 md:h-10 inline-block mx-1"
/>LEX</h1>
              <p className="text-sm sm:text-base md:text-lg
```

```
text-gray-500">Solve equations as fast as you can</p>
            </div>
            <div className="flex justify-around text-center border
border-black/10 rounded-xl p-3 md:p-4 lg:p-5 w-full">
              <div className="flex-1"><p className="text-xs md:text-sm
text-gray-500">SCORE</p><p className="text-xl sm:text-2xl md:text-3xl
lg:text-4xl font-bold">{score}</p></div>
              <div className="border-l border-black/10" />
              <div className="flex-1"><p className="text-xs md:text-sm
text-gray-500">TIME</p><p className="text-xl sm:text-2xl md:text-3xl
lg:text-4xl font-bold">{timeLeft}s</p></div>
              <div className="border-l border-black/10" />
              <div className="flex-1"><p className="text-xs md:text-sm
text-gray-500">BEST</p><p className="text-xl sm:text-2xl md:text-3xl
lg:text-4xl font-bold">{highScore}</p></div>
            </div>
            {!isPlaying && !gameOver ? (
              <div className="text-center space-y-4 py-8 w-full"><div
className="text-5xl sm:text-6xl mb-4">⚡</div><p className="text-sm
sm:text-base text-gray-600 mb-4">Test your math skills.<br />Solve
equations before time runs out.</p><button onClick={startGame}
className="px-6 sm:px-8 py-2 sm:py-3 bg-black text-white text-sm
sm:text-base font-medium rounded-full hover:bg-gray-800
transition-colors">START</button></div>
            ) : gameOver ? (
              <div className="text-center space-y-4 py-8 w-full"><div
className="text-5xl sm:text-6xl mb-4">🎯</div><p className="text-xl
sm:text-2xl font-bold">Score: {score}</p><p className="text-sm
sm:text-base text-gray-600">{score >= highScore && score > 0 ? "New
high score!" : "Keep practicing!"}</p><button onClick={startGame}
className="flex items-center gap-2 mx-auto px-5 sm:px-6 py-2 sm:py-3
bg-black text-white text-sm sm:text-base font-medium rounded-full
hover:bg-gray-800 transition-colors"><RotateCcw className="w-4 h-4"
/>RETRY</button></div>
            ) : (
              <>
                <div className="text-center py-8 md:py-10 lg:py-12
border border-black/10 rounded-xl relative w-full overflow-hidden"><p
className="text-xs sm:text-sm md:text-base text-gray-500 mb-3
md:mb-4">EQUATION</p><p className="text-3xl sm:text-4xl md:text-5xl
lg:text-6xl font-bold mb-6 md:mb-8">{equation.num1}
{equation.operation} {equation.num2} = ?</p><div className="h-14
sm:h-16 md:h-20 flex items-center justify-center"><p
className="text-3xl sm:text-4xl md:text-5xl lg:text-6xl font-bold
min-w-[100px] md:min-w-[140px] lg:min-w-[180px] border-b-4
border-black pb-2 text-center">{userAnswer || " "}</p></div>{feedback
&& (<div className={`absolute inset-0 flex items-center justify-center
text-7xl sm:text-8xl md:text-9xl ${feedback === "✓" ? "text-green-500"
: "text-red-500"}`}>{feedback}</div>)}</div>
                <div className="grid grid-cols-3 gap-2 md:gap-4
lg:gap-5 w-full mt-4 max-w-md md:max-w-lg mx-auto">
                    {[1, 2, 3, 4, 5, 6, 7, 8, 9].map((num) => (<button
key={num} onClick={() => handleNumberClick(num.toString())}
className="aspect-square border-2 border-black text-xl sm:text-2xl
md:text-3xl lg:text-4xl font-bold rounded-xl hover:bg-black
hover:text-white transition-all active:scale-95 flex items-center
justify-center">{num}</button>))}
                    <button onClick={handleClear}
className="aspect-square border-2 border-red-500 bg-red-500 text-white
text-lg md:text-xl font-bold rounded-xl hover:bg-red-500
hover:text-white transition-all active:scale-95 flex items-center
justify-center"><Delete className="w-6 h-6 md:w-8 md:h-8 lg:w-9
lg:h-9" /></button>
                    <button onClick={() => handleNumberClick("0")}
className="aspect-square border-2 border-black text-xl sm:text-2xl
md:text-3xl lg:text-4xl font-bold rounded-xl hover:bg-black
hover:text-white transition-all active:scale-95 flex items-center
justify-center">0</button>
                    <button onClick={handleEnter}
className="aspect-square border-2 border-green-600 bg-green-600
text-white text-xl md:text-2xl font-bold rounded-xl hover:bg-green-700
hover:border-green-700 transition-all active:scale-95 flex
items-center justify-center"><Check className="h-6 w-6 md:h-8 md:w-8"
/></button>
                </div>
                <button onClick={handleNegativeToggle}
className="w-full max-w-md md:max-w-lg mx-auto py-3 md:py-4 border-2
border-black text-lg sm:text-xl md:text-2xl font-bold rounded-xl
hover:bg-black hover:text-white transition-all active:scale-95
mt-2">+/-</button>
              </>
            )}
            <div className="text-center pt-4"><a
href="https://tone2vibe.in" target="_blank" rel="noopener noreferrer"
className="text-xs text-gray-400 hover:text-black
transition-colors">tone2vibe.in</a></div>
        </div>
    </div>
  );
};

// --- 2. Snake Game Component ---
const SnakeGame = ({ onBack }) => {
    const [snake, setSnake] = useState(INITIAL_SNAKE);
    const [food, setFood] = useState(INITIAL_FOOD);
    const [direction, setDirection] = useState(DIRECTIONS.ArrowRight);
    const [isPlaying, setIsPlaying] = useState(false);
    const [gameOver, setGameOver] = useState(false);
    const [score, setScore] = useState(0);
    const [highScore, setHighScore] =
useState(storedHighScores.snake);

    const gameLoop = useRef(null);
    const speed = useRef(200);

    const generateFood = (currentSnake) => {
        let newFood;
        do {
            newFood = {
                x: Math.floor(Math.random() * GRID_SIZE),
                y: Math.floor(Math.random() * GRID_SIZE),
            };
        } while (currentSnake.some(segment => segment.x === newFood.x
&& segment.y === newFood.y));
        return newFood;
    };

    const startGame = () => {
        setSnake(INITIAL_SNAKE);
        setFood(generateFood(INITIAL_SNAKE));
        setDirection(DIRECTIONS.ArrowRight);
        setScore(0);
        speed.current = 200;
        setGameOver(false);
        setIsPlaying(true);
    };

    const endGame = () => {
        setIsPlaying(false);
        setGameOver(true);
        if (score > highScore) {
            setHighScore(score);
            storedHighScores.snake = score;
        }
    };

    const moveSnake = useCallback(() => {
        setSnake(prevSnake => {
            const newSnake = [...prevSnake];
            const head = { ...newSnake[0] };
            head.x += direction.x;
            head.y += direction.y;

            // Wall collision
            if (head.x < 0 || head.x >= GRID_SIZE || head.y < 0 ||
head.y >= GRID_SIZE) {
                endGame();
                return prevSnake;
            }

            // Self collision
            for (let i = 1; i < newSnake.length; i++) {
                if (head.x === newSnake[i].x && head.y ===
newSnake[i].y) {
                    endGame();
                    return prevSnake;
                }
            }

            newSnake.unshift(head);

            // Food collision
            if (head.x === food.x && head.y === food.y) {
                setScore(s => s + 1);
                setFood(generateFood(newSnake));
                if(speed.current > 50) speed.current -= 5; // Increase
speed
            } else {
                newSnake.pop();
            }

            return newSnake;
        });
    }, [direction, food]);

    useEffect(() => {
        const handleKeyDown = (e) => {
            if (DIRECTIONS[e.key]) {
                setDirection(d => {
                    // Prevent reversing
                    if ((d.x === -DIRECTIONS[e.key].x && d.x !== 0) ||
(d.y === -DIRECTIONS[e.key].y && d.y !== 0)) {
                        return d;
                    }
                    return DIRECTIONS[e.key];
                });
            }
        };
        window.addEventListener("keydown", handleKeyDown);
        return () => window.removeEventListener("keydown",
handleKeyDown);
    }, []);
    useEffect(() => {
        if (isPlaying) {
            gameLoop.current = setInterval(moveSnake, speed.current);
            return () => clearInterval(gameLoop.current);
        }
    }, [isPlaying, moveSnake, speed.current]);

    return (
        <div className="flex flex-col items-center justify-center
w-full min-h-screen bg-white p-4 relative">
            <button onClick={() => onBack(score, highScore)}
className="fixed top-4 left-4 flex items-center gap-2 text-gray-600
hover:text-black z-30">
                <ArrowLeft className="w-6 h-6" /> <span
className="text-sm font-medium">Back to Games</span>
            </button>
            <div className="w-full max-w-lg md:max-w-xl lg:max-w-2xl
flex flex-col items-center space-y-4 md:space-y-6">
                <div className="text-center mt-6 md:mt-8">
                    <h1 className="text-3xl sm:text-4xl md:text-5xl
lg:text-6xl font-bold mb-1">Snake</h1>
                    <p className="text-sm sm:text-base md:text-lg
text-gray-500">Use arrow keys to move</p>
                </div>
                 <div className="flex justify-around text-center
border border-black/10 rounded-xl p-3 md:p-4 w-full max-w-sm">
                    <div className="flex-1"><p className="text-xs
md:text-sm text-gray-500">SCORE</p><p className="text-xl sm:text-2xl
md:text-3xl font-bold">{score}</p></div>
                    <div className="border-l border-black/10" />
                    <div className="flex-1"><p className="text-xs
md:text-sm text-gray-500">BEST</p><p className="text-xl sm:text-2xl
md:text-3xl font-bold">{highScore}</p></div>
                </div>

                <div className="relative border-2 border-black/20
bg-gray-50 aspect-square w-full max-w-sm" style={{ display: 'grid',
gridTemplateColumns: `repeat(${GRID_SIZE}, 1fr)`}}>
                     {!isPlaying ? (
                        <div className="absolute inset-0 bg-white/80
backdrop-blur-sm flex flex-col items-center justify-center z-10
text-center p-4">
                            {gameOver ? (
                                <>
                                    <div className="text-5xl
sm:text-6xl mb-4">🕹️</div>
                                    <p className="text-xl sm:text-2xl
font-bold">Game Over!</p>
                                    <p className="text-sm sm:text-base
text-gray-600">{score >= highScore && score > 0 ? "New high score!" :
"Better luck next time."}</p>
                                    <button onClick={startGame}
className="mt-4 flex items-center gap-2 mx-auto px-5 py-2 bg-black
text-white text-sm font-medium rounded-full"><RotateCcw className="w-4
h-4" />RETRY</button>
                                </>
                            ) : (
                                <>
                                    <div className="text-5xl
sm:text-6xl mb-4">🐍</div>
                                    <p className="text-sm sm:text-base
text-gray-600 mb-4">Eat the food, avoid the walls and your own
tail!</p>
                                    <button onClick={startGame}
className="px-8 py-3 bg-black text-white text-base font-medium
rounded-full">START</button>
                                </>
                            )}
                        </div>
                    ) : null}

                    {snake.map((segment, index) => (
                        <div key={index} className="bg-black
rounded-sm" style={{ gridColumn: segment.x + 1, gridRow: segment.y +
1, opacity: 1 - (index * 0.02) }}/>
                    ))}
                    <div className="bg-red-500 rounded-full" style={{
gridColumn: food.x + 1, gridRow: food.y + 1 }}/>
                </div>
            </div>
        </div>
    );
};


// --- 3. Cricket Game Component ---
const CricketGame = ({ onBack }) => {
    const [score, setScore] = useState(0);
    const [wickets, setWickets] = useState(0);
    const [gameState, setGameState] = useState('idle'); // idle,
bowling, result
    const [message, setMessage] = useState('Click "Bowl" to start');
    const [highScore, setHighScore] =
useState(storedHighScores.cricket);
    const ballTimer = useRef(null);
    const totalWickets = 5;

    const resetBall = () => {
        setGameState('idle');
        setMessage('Ready for the next ball!');
    };

    const handleBowl = () => {
        if (gameState !== 'idle') return;
        setGameState('bowling');
        setMessage('...');

        const hitTime = Math.random() * 2000 + 1000; // Ball arrives
in 1-3 seconds
        ballTimer.current = setTimeout(() => {
            setGameState('result');
            setMessage('OUT! Too slow!');
            setWickets(w => w + 1);
        }, hitTime);
    };

    const handleHit = () => {
        if (gameState !== 'bowling') return;
        clearTimeout(ballTimer.current);

        const outcome = Math.random();
        let runs = 0;

        if (outcome < 0.1) {
            setMessage('OUT! You missed!');
            setWickets(w => w + 1);
        } else if (outcome < 0.4) {
            runs = 1;
            setMessage('1 RUN!');
        } else if (outcome < 0.6) {
            runs = 2;
            setMessage('2 RUNS!');
        } else if (outcome < 0.8) {
            runs = 4;
            setMessage('FOUR!');
        } else {
            runs = 6;
            setMessage('SIX! What a shot!');
        }
        setScore(s => s + runs);
        setGameState('result');
    };
    const startGame = () => {
        setScore(0);
        setWickets(0);
        setGameState('idle');
        setMessage('Click "Bowl" to start');
    }

    useEffect(() => {
        if(gameState === 'result') {
            if (wickets >= totalWickets) {
                setMessage(`GAME OVER! Final Score: ${score}`);
                if (score > highScore) {
                    setHighScore(score);
                    storedHighScores.cricket = score;
                }
            } else {
                setTimeout(resetBall, 1500);
            }
        }
    }, [gameState, wickets, score, highScore]);

    const isGameOver = wickets >= totalWickets;

    return (
        <div className="flex flex-col items-center justify-center
w-full min-h-screen bg-white p-4 relative">
             <button onClick={() => onBack(score, highScore)}
className="fixed top-4 left-4 flex items-center gap-2 text-gray-600
hover:text-black z-30">
                <ArrowLeft className="w-6 h-6" /> <span
className="text-sm font-medium">Back to Games</span>
            </button>
            <div className="w-full max-w-lg md:max-w-xl lg:max-w-2xl
flex flex-col items-center space-y-4 md:space-y-6">
                <div className="text-center mt-6 md:mt-8">
                    <h1 className="text-3xl sm:text-4xl md:text-5xl
lg:text-6xl font-bold mb-1">Timing Cricket</h1>
                    <p className="text-sm sm:text-base md:text-lg
text-gray-500">Hit the ball at the right time</p>
                </div>
                 <div className="flex justify-around text-center
border border-black/10 rounded-xl p-3 md:p-4 lg:p-5 w-full">
                    <div className="flex-1"><p className="text-xs
md:text-sm text-gray-500">SCORE</p><p className="text-xl sm:text-2xl
md:text-3xl lg:text-4xl font-bold">{score}</p></div>
                    <div className="border-l border-black/10" />
                    <div className="flex-1"><p className="text-xs
md:text-sm text-gray-500">WICKETS</p><p className="text-xl sm:text-2xl
md:text-3xl lg:text-4xl font-bold">{wickets}/{totalWickets}</p></div>
                    <div className="border-l border-black/10" />
                    <div className="flex-1"><p className="text-xs
md:text-sm text-gray-500">BEST</p><p className="text-xl sm:text-2xl
md:text-3xl lg:text-4xl font-bold">{highScore}</p></div>
                </div>

                <div className="text-center space-y-4 py-8 w-full flex
flex-col items-center">
                    <div className="h-48 w-full max-w-sm border-2
border-black/10 rounded-xl flex flex-col items-center justify-center
p-4">
                        <div className="text-5xl mb-4">
                            {gameState === 'bowling' && '🏏'}
                            {gameState !== 'bowling' && !isGameOver &&
'⏳'}
                            {isGameOver && '🏆'}
                        </div>
                        <p className="text-xl font-semibold
text-gray-800 h-14 flex items-center">{message}</p>
                    </div>

                    {isGameOver ? (
                         <button onClick={startGame} className="mt-4
flex items-center gap-2 mx-auto px-6 py-3 bg-black text-white
text-base font-medium rounded-full"><RotateCcw className="w-4 h-4"
/>RETRY</button>
                    ) : (
                        <div className="flex gap-4 mt-4">
                            <button onClick={handleBowl}
disabled={gameState !== 'idle'} className="px-8 py-4 bg-blue-600
text-white text-lg font-bold rounded-full disabled:bg-gray-400
disabled:cursor-not-allowed hover:bg-blue-700
transition-colors">BOWL</button>
                            <button onClick={handleHit}
disabled={gameState !== 'bowling'} className="px-8 py-4 bg-red-600
text-white text-lg font-bold rounded-full disabled:bg-gray-400
disabled:cursor-not-allowed hover:bg-red-700
transition-colors">HIT</button>
                        </div>
                    )}
                </div>
            </div>
        </div>
    );
};
```

```
//
=========================================================================
======
// === MAIN OFFLINE COMPONENT (ORCHESTRATOR)
//
=========================================================================
======
export default function Offline() {
  const [activeScreen, setActiveScreen] = useState('landing'); //
landing, menu, math, snake, cricket
  const [lastGameSummary, setLastGameSummary] = useState({ score: 0,
highScore: 0, played: false });

  const handleBackFromGame = (score, highScore) => {
    setLastGameSummary({ score, highScore, played: true });
    setActiveScreen('menu');
  };

  const GameMenu = () => (
    <div className="relative z-10 flex flex-col items-center w-full">
        <button onClick={() => setActiveScreen('landing')}
className="fixed top-4 left-4 flex items-center gap-2 text-gray-600
hover:text-black z-30">
            <ArrowLeft className="w-6 h-6" /> <span className="text-sm
font-medium">Back</span>
        </button>
        <h1 className="text-3xl sm:text-4xl font-bold mb-8">Choose a
Game</h1>
        <div className="grid grid-cols-1 md:grid-cols-3 gap-4 w-full
max-w-4xl">
            <button onClick={() => setActiveScreen('math')}
className="p-6 border border-black/10 rounded-xl text-left
hover:bg-gray-50 transition-colors space-y-2">
                <BrainCircuit className="w-8 h-8 text-blue-500" />
                <h2 className="text-xl font-bold">Math Runner</h2>
                <p className="text-sm text-gray-600">Solve equations
against the clock.</p>
                <p className="text-xs text-gray-400 mt-2">High Score:
{storedHighScores.math}</p>
            </button>
            <button onClick={() => setActiveScreen('snake')}
className="p-6 border border-black/10 rounded-xl text-left
hover:bg-gray-50 transition-colors space-y-2">
                <Gamepad2 className="w-8 h-8 text-green-500" />
                <h2 className="text-xl font-bold">Snake</h2>
                <p className="text-sm text-gray-600">Classic arcade
snake game.</p>
```

```
                <p className="text-xs text-gray-400 mt-2">High Score:
{storedHighScores.snake}</p>
            </button>
             <button onClick={() => setActiveScreen('cricket')}
className="p-6 border border-black/10 rounded-xl text-left
hover:bg-gray-50 transition-colors space-y-2">
                <Tally1 className="w-8 h-8 text-red-500" />
                <h2 className="text-xl font-bold">Timing Cricket</h2>
                <p className="text-sm text-gray-600">A simple
reaction-based cricket game.</p>
                <p className="text-xs text-gray-400 mt-2">High Score:
{storedHighScores.cricket}</p>
            </button>
        </div>
    </div>
  );

  const LandingScreen = () => (
    <div className="relative z-10 flex flex-col items-center max-w-2xl
mx-auto px-4 sm:px-6">
        <div className="relative mb-6 md:mb-8 flex items-center
justify-center gap-3 sm:gap-4 md:gap-5">
            <div className="relative"><div className="absolute inset-0
animate-ping"><div className="w-10 h-10 sm:w-12 sm:h-12 md:w-14
md:h-14 border-2 border-black/20 rounded-full" /></div><WifiOff
className="w-10 h-10 sm:w-12 sm:h-12 md:w-14 md:h-14 text-black
relative z-10" strokeWidth={1.5} /></div>
            <h1 className="text-3xl sm:text-4xl md:text-5xl
lg:text-6xl font-bold bg-gradient-to-r from-black via-gray-600
to-black bg-clip-text text-transparent
animate-[shimmer_3s_ease-in-out_infinite] bg-[length:200%_100%]
whitespace-nowrap">You're Offline</h1>
        </div>
        <p className="text-base sm:text-lg md:text-xl text-gray-600
mb-8 md:mb-12 max-w-md flex items-center gap-2 justify-center
text-center px-4 mt-4"><span>Challenge your mind with a tone2vibe
game</span></p>

        {lastGameSummary.played && (
            <div className="mb-6 text-center"><p className="text-sm
text-gray-500 mb-1">Last Score</p><p className="text-3xl
font-bold">{lastGameSummary.score}</p><p className="text-xs
text-gray-400 mt-1">Best: {lastGameSummary.highScore}</p></div>
        )}

        <button onClick={() => setActiveScreen('menu')}
className="group relative px-6 py-3 sm:px-8 sm:py-4 bg-black
text-white font-semibold rounded-full overflow-hidden transition-all
```

```
duration-300 hover:scale-105 hover:shadow-[0_0_40px_rgba(0,0,0,0.3)]
text-sm sm:text-base">
            <div className="absolute inset-0 bg-gradient-to-r
from-gray-900 via-black to-gray-900 opacity-0 group-hover:opacity-100
transition-opacity duration-300" /><span className="relative flex
items-center gap-2"><Play className="w-4 h-4 sm:w-5 sm:h-5"
fill="currentColor" />{lastGameSummary.played ? "Play Again" : "Start
Playing"}</span>
        </button>
        <div className="mt-12 md:mt-16 flex flex-col items-center
gap-3 text-xs sm:text-sm text-gray-400 px-4 text-center">
            <div className="flex items-center gap-2"><Wifi
className="w-3 h-3 sm:w-4 sm:h-4 flex-shrink-0" /><span>Reconnect
anytime to sync your progress</span></div>
            <a href="https://tone2vibe.in" target="_blank"
rel="noopener noreferrer" className="text-gray-500 hover:text-black
transition-colors duration-200">tone2vibe.in</a>
        </div>
    </div>
  );

  const renderContent = () => {
    switch (activeScreen) {
        case 'landing': return <LandingScreen />;
        case 'menu': return <GameMenu />;
        case 'math': return <MathRunner onBack={handleBackFromGame}
/>;
        case 'snake': return <SnakeGame onBack={handleBackFromGame}
/>;
        case 'cricket': return <CricketGame
onBack={handleBackFromGame} />;
        default: return <LandingScreen />;
    }
  };

  return (
    <div className="relative flex flex-col items-center justify-center
min-h-screen w-full bg-white overflow-hidden p-4">
      <div className="absolute inset-0
bg-[linear-gradient(to_right,#00000008_1px,transparent_1px),linear-gra
dient(to_bottom,#00000008_1px,transparent_1px)] bg-[size:4rem_4rem]
md:bg-[size:4rem_4rem]
[mask-image:radial-gradient(ellipse_60%_50%_at_50%_50%,#000_60%,transp
arent_100%)]" />
      <div className="absolute top-10 left-10 md:top-20 md:left-20
w-40 h-40 md:w-72 md:h-72 bg-black/5 rounded-full blur-3xl
animate-pulse" />
      <div className="absolute bottom-10 right-10 md:bottom-20
```

```
md:right-20 w-48 h-48 md:w-96 md:h-96 bg-black/3 rounded-full blur-3xl
animate-pulse delay-1000" />

      <div className="relative z-10 w-full h-full flex items-center
justify-center">
        {renderContent()}
      </div>

      <style>{`@keyframes shimmer { 0%, 100% { background-position: 0%
50%; } 50% { background-position: 100% 50%; } } .delay-1000 {
animation-delay: 1s; }`}</style>
    </div>
  );
}
```