```
"use client"

import { use, useEffect, useState, type JSX } from "react"
import { Button } from "@/components/ui/button"
import { Card, CardContent } from "@/components/ui/card"
import { Progress } from "@/components/ui/progress"
import { Badge } from "@/components/ui/badge"
import {
    Zap, Shield,
    Swords, Package, User2, LogOut, Bird, Bug, Circle,
    Droplet, Eye, Flame, Ghost, Globe, HelpCircle, Leaf,
    Moon, Skull, Snowflake, Sparkles, Mountain,
    Star,
    ArrowLeft
} from "lucide-react"
import { SimplePokeballIcon } from "@/components/ui/pokeball-icon"
import type { Pokemon, Pokemon_in_battle, Moves } from "@/lib/types"
import { motion, AnimatePresence } from "framer-motion"
import { calculateTypeEffectiveness, PokemonType } from "@/lib/typeEffectiveness"
import battlefieldImage from "@/assets/pokemon-battlefield.png";

import React from "react"

// All possible menu options
type MenuOption = "main" | "attack" | "pokemon" | "items" | "run"

// Props for the PokemonBattler component
type PokemonBattlerProps = {
    FullUserTeam: Pokemon[];
    FullOpponentTeam: Pokemon[];
    onEndofBattle: () => void;
}

/**
 * PokemonBattler component for handling battles between two Pokémon teams.
 * This component manages the battle state, animations, and interactions between
the player's team and the opponent's team.
 * It includes functionality for selecting moves, switching Pokémon, and
displaying battle text.
 * @param FullUserTeam - The full team of the user, containing Pokémon objects
 * @param FullOpponentTeam - The full team of the opponent, containing Pokémon
objects
 * @param onEndofBattle - Callback function to be called when the battle ends
 * @returns
 */
export default function pokemon_battle({
    FullUserTeam,
    FullOpponentTeam,
    onEndofBattle
}: PokemonBattlerProps) {

    // Making the teams battle-ready
    // This includes setting current HP, PP, and other battle-related properties
    // This is done using useMemo to avoid unnecessary recalculations on every
render
    const initialUserTeam = React.useMemo(
        () => FullUserTeam.map((p) => p.makeBattleReady()),
        [FullUserTeam]
    )
    const initialOpponentTeam = React.useMemo(
        () => FullOpponentTeam.map((p) => p.makeBattleReady()),
        [FullOpponentTeam]
    )
    // Team States to manage the player's and opponent's Pokémon teams
    const [playerTeam, setPlayerTeam] = useState<Pokemon_in_battle[]>
(initialUserTeam)
    const [opponentTeam, setOpponentTeam] = useState<Pokemon_in_battle[]>
(initialOpponentTeam)

    // Active Pokémon Indices
    const [activePlayerIndex, setActivePlayerIndex] = useState(0)
    const [activeOpponentIndex, setActiveOpponentIndex] = useState(0)

    // Active Pokémon from the current teams
    const playerPokemon = playerTeam[activePlayerIndex]
    const opponentPokemon = opponentTeam[activeOpponentIndex]

    // Battle Text initial with current player Pokémon
    const [battleText, setBattleText] = useState(() => `What will
${playerPokemon.name} do?`)

    // Battle States needed to manage the Battle Flow
    const [isPlayerTurn, setIsPlayerTurn] = useState(true)
    const [isAnimating, setIsAnimating] = useState(false)
    const [currentMenu, setCurrentMenu] = useState<MenuOption>("main")
    const [showRunConfirmation, setShowRunConfirmation] = useState(false)

    // Animation States for Player
    const [isSwitchingPlayer, setIsSwitchingPlayer] = useState(false)
    const [switchingInPlayer, setSwitchingInPlayer] = useState<Pokemon_in_battle |
null>(null)
    const [switchingOutPlayer, setSwitchingOutPlayer] = useState<Pokemon_in_battle
| null>(null)
    const [switchDirectionPlayer, setSwitchDirectionPlayer] = useState<"in" |
"out">("out")

    // Animation States for Opponent
    const [isSwitchingOpponent, setIsSwitchingOpponent] = useState(false)
    const [switchingInOpponent, setSwitchingInOpponent] =
useState<Pokemon_in_battle | null>(null)
    const [switchingOutOpponent, setSwitchingOutOpponent] =
useState<Pokemon_in_battle | null>(null)
    const [switchDirectionOpponent, setSwitchDirectionOpponent] = useState<"in" |
"out">("out")
```

```
    // Pending Action State to manage the next action in the battle
    type PendingAction = null |
    { type: "playerMoveFirst"; move: Moves } |
    { type: "opponentMoveFirst"; move?: Moves } |
    { type: "playerMoveSecond"; move: Moves } |
    { type: "opponentMoveSecond"; move?: Moves } |
    { type: "switchPokemon"; newPokemon: Pokemon_in_battle } |
    { type: "switchOpponent"; newPokemon: Pokemon_in_battle } |
    { type: "ForcedSwitch"; newPokemon: Pokemon_in_battle }

    const [pendingAction, setPendingAction] = useState<PendingAction>(null)
    const delay = (ms: number) => new Promise((res) => setTimeout(res, ms));

    /**
     * Updates a Pokémon in the team.
     * @param team - The current team of Pokémon.
     * @param setTeam - The state setter function for the team.
     * @param index - The index of the Pokémon to update.
     * @param updated - The updated Pokémon data.
     */
    function updateTeamPokemon(
        team: Pokemon_in_battle[],
        setTeam: React.Dispatch<React.SetStateAction<Pokemon_in_battle[]>>,
        index: number,
        updated: Pokemon_in_battle
    ) {
        console.log("Logs from updateTeamPokemon: ", team, setTeam, index,
updated);
        setTeam(prev =>
            prev.map((p, i) => (i === index ? updated : p))
        );
    }

    /**
     * Performs a switch between Pokémon in the battle.
     * @param newPokemon - The new Pokémon to switch in.
     * @param isPlayer - Indicates if the switch is for the player or the
opponent.
     * @param forced - Indicates if the switch is forced (e.g., after fainting).
     * @returns A promise that resolves when the switch is complete.
     */
    const performSwitch = async ({
        newPokemon,
        isPlayer,
        forced = false,
    }: {
        newPokemon: Pokemon_in_battle;
        isPlayer: boolean;
        forced?: boolean;
    }) => {
        // Checks which team and active index to use based on whether it's the
player or opponent
        const team = isPlayer ? playerTeam : opponentTeam;
        const activeIndex = isPlayer ? activePlayerIndex : activeOpponentIndex;
        const currentPokemon = team[activeIndex];

        // Checks if the new Pokémon is valid for switching
        if (!newPokemon || newPokemon.id === currentPokemon.id ||
newPokemon.currentHP <= 0) {
            console.log("Invalid Pokémon for switching");
            return;
        }

        // Checks if the current Pokémon is fainted and if a forced switch is
allowed
        if (forced && currentPokemon.currentHP > 0) {
            console.log("Forced switch not allowed when Pokémon is still alive");
            return;
        }

        // Start switch animation
        if (isPlayer) {
            setIsSwitchingPlayer(true);
            setSwitchingOutPlayer(currentPokemon);
            setSwitchingInPlayer(newPokemon);
            setSwitchDirectionPlayer("out");
        } else {
            setIsSwitchingOpponent(true);
            setSwitchingOutOpponent(currentPokemon);
            setSwitchingInOpponent(newPokemon);
            setSwitchDirectionOpponent("out");
        }

        // Battle text for switching Pokémon
        setBattleText(`Come back, ${currentPokemon.name}!`);
        await delay(1000);

        // Update the current Battle state to reflect the switch
        if (isPlayer) {
            setSwitchDirectionPlayer("in");
        } else {
            setSwitchDirectionOpponent("in");
        }

        setBattleText(`Go, ${newPokemon.name}!`);
        await delay(1000);

        // Update the team with the new Pokémon
        const newIndex = team.findIndex((p) => p.id === newPokemon.id);
        if (newIndex !== -1) {
            if (isPlayer) {
                setActivePlayerIndex(newIndex);
                setIsSwitchingPlayer(false);
                setSwitchingOutPlayer(null);
                setSwitchingInPlayer(null);
                setCurrentMenu("main");
            } else {
                setActiveOpponentIndex(newIndex);
                setIsSwitchingOpponent(false);
                setSwitchingOutOpponent(null);
                setSwitchingInOpponent(null);
            }
        }

        if (isPlayer) {
            setBattleText(`What will ${newPokemon.name} do?`);
        }
        setIsAnimating(false);
    };

    /**
     * Performs a attack move in the battle. It updates the Pokémon's stats,
calculates damage, and handles animations.
     * * This function calculates the damage based on the move's power, type
effectiveness, and Pokémon stats.
     * * It handles critical hits, STAB (Same Type Attack Bonus), and type
effectiveness.
     * @param attacker - The Pokémon performing the attack.
     * @param defender - The Pokémon receiving the attack.
     * @param move - The move being used.
     * @param isPlayerAttacker - Whether the attacker is the player.
     * @returns The result of the attack move.
     */
    const performMove = async (
        attacker: Pokemon_in_battle,
        defender: Pokemon_in_battle,
        move: Moves,
        isPlayerAttacker: boolean
    ) => {
        // Start attack animation
        setIsAnimating(true);

        // Calculate the damage based on the move's power, type effectiveness, and
Pokémon stats
        // This includes critical hits, STAB (Same Type Attack Bonus), and random
damage variation
        const criticalHit = Math.random() < 0.0625;
        const criticalMultiplier = criticalHit ? 1.5 : 1;

        const stab = attacker.types.includes(move.type) ? 1.5 : 1;
        const typeEffectiveness = calculateTypeEffectiveness(
            (move.type as string).toLowerCase() as PokemonType,
            (defender.types as string[]).map((type) => type.toLowerCase() as
PokemonType)
        );
        const random = (Math.floor(Math.random() * (255 - 217 + 1)) + 217) / 255;
// 0.85-1.0

        let damage = 0;
        // Calculate damage based on the move's damage class
        if (move.damageClass === "physical") {
            damage = Math.floor(
                ((((2 * attacker.lvl * criticalMultiplier) / 5 + 2) * move.power *
(attacker.attack / defender.defense)) / 50 + 2)
                * stab * typeEffectiveness * random
            );
        } else if (move.damageClass === "special") {
            damage = Math.floor(
                ((((2 * attacker.lvl * criticalMultiplier) / 5 + 2) * move.power *
(attacker.special_attack / defender.special_defense)) / 50 + 2)
                * stab * typeEffectiveness * random
            );
        }

        console.log("Calculated damage:", damage);

        // Ensure that the newHp for the defender is not negative
        const newHp = Math.max(0, defender.currentHP - damage);

        // Update the teams and Pokémon states based on the attack
        const attackerIndex = isPlayerAttacker ? activePlayerIndex :
activeOpponentIndex;
        const defenderIndex = isPlayerAttacker ? activeOpponentIndex :
activePlayerIndex;
        const attackerTeam = isPlayerAttacker ? playerTeam : opponentTeam;
        const defenderTeam = isPlayerAttacker ? opponentTeam : playerTeam;
        const setAttackerTeam = isPlayerAttacker ? setPlayerTeam :
setOpponentTeam;
        const setDefenderTeam = isPlayerAttacker ? setOpponentTeam :
setPlayerTeam;

        const updatedAttacker = attacker.clone();
        // Update the attacker's PP for the used move
        updatedAttacker.setCurrentPP(move.name, Math.max(0,
updatedAttacker.getCurrentPP(move.name) - 1));
        updateTeamPokemon(attackerTeam, setAttackerTeam, attackerIndex,
updatedAttacker);

        const isStatusMove = move.damageClass === "status";

        if (!isStatusMove) {
            const updatedDefender = defender.clone();
            updatedDefender.setCurrentHP(newHp);
            updateTeamPokemon(defenderTeam, setDefenderTeam, defenderIndex,
updatedDefender);
        }

        // Kampftext und Animation
        setBattleText(`${attacker.name} used ${move.name}!`);
        await delay(1000);

        if (criticalHit) {
            setBattleText(`A critical hit!`);
            await delay(1000);
        }
        if (!isStatusMove) {
            if (typeEffectiveness > 1) {
                setBattleText(`It's super effective!`);
                await delay(1000);
            } else if (typeEffectiveness < 1) {
                setBattleText(`It's not very effective...`);
                await delay(1000);
            } else {
                setBattleText(`${defender.name} took ${damage} damage!`);
                await delay(1000);
            }
        }

        // Fainting Handling
        if (newHp === 0) {
            setBattleText(`${defender.name} fainted!`);
            console.log(`${defender.name} fainted!`);
            await delay(1000);

            // Check if there are any available Pokémon left in the defender's
team
            const availableDefenderPokemon = defenderTeam.filter(
                (p) => p.currentHP > 0 && p.id !== defender.id
            );

            if (availableDefenderPokemon.length > 0) {
                // If there are available Pokémon, switch to one of them
                const nextPokemon =
availableDefenderPokemon[Math.floor(Math.random() *
availableDefenderPokemon.length)];
                setBattleText(isPlayerAttacker ? "Opponent is switching
Pokémon..." : "Please choose your next Pokémon.");
                await delay(1000);

                // Automatically switch for the opponent
                if (isPlayerAttacker) {
                    console.log("Opponent switching Pokémon:", nextPokemon.name);
                    setPendingAction({ type: "switchOpponent", newPokemon:
nextPokemon });
                    return newHp;
                } else {
                    // Set the switching state for the player so that the Menu
opens
                    setCurrentMenu("pokemon");
                    console.log("Player must choose next Pokémon.");
                    return newHp;
                }
            } else {
                // If no Pokémon left, end the battle
                setBattleText(isPlayerAttacker ? "Opponent has no Pokémon left!
You win!" : "All your Pokémon fainted!");
                setBattleText(isPlayerAttacker ? "You win!" : "You lost!");
                console.log(isPlayerAttacker ? "Player wins!" : "Player lost.");
                setPendingAction(null);
                setIsAnimating(false);
                return newHp;
            }
        }
        setIsAnimating(false);
    };

    // Log pending actions for debugging
    useEffect(() => {
        console.log(pendingAction);

    }, [pendingAction]);

    useEffect(() => {
        // If there is no pending action, do nothing
        if (!pendingAction) return;

        // Helper to process player's move
        const processPlayerMove = async (move: Moves, first?: boolean) => {
            const result = await performMove(playerTeam[activePlayerIndex],
opponentTeam[activeOpponentIndex], move, true);
            // If this is the first move of the turn, set the pending action for
the opponent's move
            if (first && result !== 0) {
                setIsPlayerTurn(false);
                setPendingAction({ type: "opponentMoveSecond" });
                setIsAnimating(true);
            } else {
                setBattleText(`What will ${playerTeam[activePlayerIndex].name} do?
`);
                setIsPlayerTurn(true);
            }
        }

        // Helper to process opponent's move
        const processOpponentMove = async (move?: Moves, first?: boolean) => {
            const activeOpponentPokemon = opponentTeam[activeOpponentIndex];
            const randomMove =
activeOpponentPokemon.moveset[Math.floor(Math.random() *
activeOpponentPokemon.moveset.length)];
            const result = await performMove(activeOpponentPokemon,
playerTeam[activePlayerIndex], randomMove, false);

            if (first && move && result !== 0) {
                setIsPlayerTurn(true);
                setPendingAction({ type: "playerMoveSecond", move: move });
                setIsAnimating(true);
            } else{
                setBattleText(`What will ${playerTeam[activePlayerIndex].name} do?
`);
                setIsPlayerTurn(true);
                setIsAnimating(false);
            }
        };
        // Helper to process player switching Pokémon
        const processSwitchPokemon = async (newPokemon: Pokemon_in_battle) => {
            await performSwitch({ newPokemon, isPlayer: true });
            setIsPlayerTurn(false); // Spieler war dran, hat gewechselt, jetzt
Gegner dran
            setPendingAction({ type: "opponentMoveSecond" });
        };

        // Helper to process opponent switching Pokémon
        const processSwitchOpponent = async (newPokemon: Pokemon_in_battle) => {
            console.log("Opponent switching Pokémon:", newPokemon.name);
            await performSwitch({ newPokemon, isPlayer: false });
            setIsAnimating(false);
            setIsPlayerTurn(true);
            setBattleText(`What will ${playerTeam[activePlayerIndex].name} do?`);
        };

        // Helper to process forced switch (player's Pokémon fainted)
        const processForcedSwitch = async (newPokemon: Pokemon_in_battle) => {
            const availablePlayerPokemon = playerTeam.filter(
                (p) => p.currentHP > 0 && p.id !== playerPokemon.id
            );
            if (availablePlayerPokemon.length === 0) {
                setBattleText("All your Pokémon have fainted! You lose!");
                console.log("Player lost, no Pokemon left");
                return;
            } else {
                // If there are available Pokémon, allow switching
                await performSwitch({ newPokemon, isPlayer: true, forced: true });
                setIsPlayerTurn(true);
                setBattleText(`What will ${newPokemon.name} do?`);

            }
        };

        // Process the pending action based on its type
        (async () => {
            console.log("Pending action:", pendingAction);
            if (pendingAction.type === "playerMoveFirst") {
                await processPlayerMove(pendingAction.move, true);
            } else if (pendingAction.type === "opponentMoveSecond") {
                await processOpponentMove(pendingAction.move, false);
            } else if (pendingAction.type === "opponentMoveFirst") {
                await processOpponentMove(pendingAction.move, true);
            } else if (pendingAction.type === "playerMoveSecond") {
                await processPlayerMove(pendingAction.move, false);
            } else if (pendingAction.type === "switchPokemon") {
                await processSwitchPokemon(pendingAction.newPokemon);
            } else if (pendingAction.type === "switchOpponent") {
                await processSwitchOpponent(pendingAction.newPokemon);
            } else if (pendingAction.type === "ForcedSwitch") {
                await processForcedSwitch(pendingAction.newPokemon);
            }
        })();
    }, [pendingAction]);


    // Handles the player's move selection and updates the battle state
accordingly
    const handlePlayerMove = (move: Moves) => {
        if (!isPlayerTurn || isAnimating) return
        console.log("Player's turn to move:", move.name);
        setPendingAction({ type: "playerMoveFirst", move })
        setIsAnimating(true)
        setCurrentMenu("main")
        setBattleText(`${playerPokemon.name} used ${move.name}!`)
    }

    // Handles the opponent's move selection and updates the battle state
accordingly
    const handleOpponentMove = (move: Moves) => {
        if (isAnimating) return;
        setIsPlayerTurn(false)
        console.log("Opponent's turn to move:", move.name);
        setPendingAction({ type: "opponentMoveFirst", move })
        setIsAnimating(true)
        setCurrentMenu("main")
        setBattleText(`${opponentPokemon.name} used ${move.name}!`)
    }

    // Handles the menu selection and updates the current menu state
    const handleMenuSelect = (menu: MenuOption) => {
        if (menu === "items") {
            setBattleText("Your bag is empty!")
            return
        }

        if (menu === "run" || (menu === "main" && currentMenu === "main")) {
            setShowRunConfirmation(true)
            setBattleText("Are you sure you want to give up?")
            return
        }

        setCurrentMenu(menu)
        if (menu === "main") {
            setBattleText("What will " + playerPokemon.name + " do?")
        }
    }

    // Handles the run action confirmation
    const handleRunConfirm = () => {
        setBattleText("You ran away from the battle!")
        setShowRunConfirmation(false)
        onEndofBattle()
    }

    // Handles the run action cancellation
    const handleRunCancel = () => {
        setShowRunConfirmation(false)
        setBattleText("What will " + playerPokemon.name + " do?")
    }

    // Handles the Pokémon selection from the player's team when switching Pokémon
is forced
    const handlePokemonSelect = (pokemon: Pokemon_in_battle) => {
        // Dead Pokémon cannot be selected
        if (pokemon.currentHP <= 0) return;

        const currentPlayer = playerTeam[activePlayerIndex];

        // If the current Pokémon is still alive and the same one is selected,
cancel
        if (currentPlayer.currentHP > 0 && pokemon.id === currentPlayer.id)
return;

        // If the current Pokémon is fainted, a forced switch occurs (e.g., after
fainting)
        if (currentPlayer.currentHP === 0) {
            setPendingAction({ type: "ForcedSwitch", newPokemon: pokemon });
        } else {
            // Normal switch through Pokemon menu
            switchPlayerPokemon(pokemon);
        }
    };

    // Handles the Player's Pokémon switch action through the Pokémon menu
    const switchPlayerPokemon = (newPokemon: Pokemon_in_battle) => {
        if (isAnimating || !isPlayerTurn) return
        console.log("Switching player Pokémon:", newPokemon.name);
        setPendingAction({ type: "switchPokemon", newPokemon })
    }

    // Handles the Opponent's Pokémon switch action through the Pokémon menu
    const switchOpponentPokemon = (newPokemon: Pokemon_in_battle) => {
        if (isAnimating || isPlayerTurn) return
        console.log("Switching opponent Pokémon:", newPokemon.name);
        setPendingAction({ type: "switchOpponent", newPokemon })
        setCurrentMenu("main")
    }

    // Returns the color class for a Pokémon type
    const getTypeColor = (type: string): string => {
        const colors: { [key: string]: string } = {
            Normal: "bg-gray-400 text-black",
            Fire: "bg-red-500 text-white",
            Water: "bg-blue-500 text-white",
            Electric: "bg-yellow-400 text-black",
            Grass: "bg-green-500 text-white",
            Ice: "bg-blue-300 text-black",
            Fighting: "bg-red-700 text-white",
            Poison: "bg-purple-500 text-white",
            Ground: "bg-yellow-700 text-black",
            Flying: "bg-indigo-400 text-white",
            Psychic: "bg-pink-400 text-white",
            Bug: "bg-lime-500 text-black",
            Rock: "bg-yellow-600 text-white",
            Ghost: "bg-purple-700 text-white",
            Dragon: "bg-purple-600 text-white",
            Dark: "bg-gray-800 text-white",
            Steel: "bg-gray-600 text-white",
            Fairy: "bg-pink-300 text-black",
        }

        return colors[type] || "bg-gray-500 text-white"
    }

    // Returns the color class for the HP bar based on current and max HP
    const getHPColor = (currentHP: number, maxHP: number): string => {
        const percentage = (currentHP / maxHP) * 100
        if (percentage > 50) return "bg-green-500"
        if (percentage > 25) return "bg-yellow-500"
        return "bg-red-500"
    }

    // Returns the icon for a Pokémon type
    const getTypeIcon = (type: string): JSX.Element => {
        const iconMap: { [key: string]: JSX.Element } = {
            Normal: <Circle className="w-4 h-4" />,
            Fire: <Flame className="w-4 h-4" />,
            Water: <Droplet className="w-4 h-4" />,
            Electric: <Zap className="w-4 h-4" />,
            Grass: <Leaf className="w-4 h-4" />,
            Ice: <Snowflake className="w-4 h-4" />,
            Fighting: <Swords className="w-4 h-4" />,
            Poison: <Skull className="w-4 h-4" />,
            Ground: <Globe className="w-4 h-4" />,
            Flying: <Bird className="w-4 h-4" />,
            Psychic: <Eye className="w-4 h-4" />,
            Bug: <Bug className="w-4 h-4" />,
            Rock: <Mountain className="w-4 h-4" />,
            Ghost: <Ghost className="w-4 h-4" />,
            Dragon: <Mountain className="w-4 h-4" />,
            Dark: <Moon className="w-4 h-4" />,
            Steel: <Shield className="w-4 h-4" />,
            Fairy: <Sparkles className="w-4 h-4" />,
            Stellar: <Star className="w-4 h-4" />,
        }

        return iconMap[type] || <HelpCircle className="w-4 h-4" />
    }

    // Sets the currently displayed Opponent Pokémon based on the switching state
    const displayedOpponent = isSwitchingOpponent
        ? switchDirectionOpponent === "in"
            ? switchingInOpponent
            : switchingOutOpponent
        : opponentTeam[activeOpponentIndex];

    // Sets depending on if the opponent is switching out a dead Pokémon
    const isOpponentFainting = isSwitchingOpponent && switchDirectionOpponent ===
"out";

    // Forces the displayed HP to 0 if the opponent is fainting
    const displayedCurrentHP = isOpponentFainting ? 0 :
displayedOpponent?.currentHP ?? 0;
    const displayedMaxHP = displayedOpponent?.maxHP ?? 1;

    return (
        <div className="min-h-screen bg-gradient-to-br from-slate-50 to-slate-100
dark:from-slate-900 dark:to-slate-800">
            <div className="container mx-auto px-2 py-4">
                <div className="max-w-3xl mx-auto relative">
                    {/* Opponent Team Status - Top Left */}
                    <div className="absolute top-4 left-4 z-10">
                        <Card className="bg-white/90 dark:bg-slate-800/90
backdrop-blur-sm border-0 shadow-lg">
                            <CardContent className="p-2">
                                <div className="text-xs text-muted-foreground mb-
1">Opponent</div>
                                <div className="flex gap-1">
                                    {/* Display for each Member on opponents team
a Pokeball
                                        and greys out when the representative
Pokémon fainted */}
                                    {opponentTeam.map((pokemon, index) => (
                                        <SimplePokeballIcon
                                            key={index}
                                            size={16}
                                            className={`transition-all duration-
300 ${pokemon.currentHP > 0
                                                ? "text-red-500 hover:scale-110"
                                                : "text-gray-400 dark:text-gray-
600 opacity-50"
                                                }`}
                                        />
                                    ))}
                                </div>
                            </CardContent>
                        </Card>
                    </div>

                    {/* Battle Arena */}
                    <Card className="shadow-2xl border-0 bg-white/80 dark:bg-
slate-800/80 backdrop-blur-sm mb-3">
                        <CardContent className="p-0">
                            <div
                                className="relative min-h-[350px] rounded-t-lg
overflow-hidden bg-cover bg-center bg-no-repeat"
                                style={{
                                    // Set the background image for the
battlefield
                                    backgroundImage: `url(${battlefieldImage})`,
                                    backgroundSize: "cover",
                                    backgroundPosition: "center",
                                }}
                            >
                                {/* Overlay for better readability */}
                                <div className="absolute inset-0 bg-black/10">
</div>

                                {/* Opponent Pokemon */}
                                <div className="relative z-10 flex justify-end pt-
6 pr-6 mb-6">
                                    <div className="flex items-center gap-3">
                                        {/* Animate the opponent's Pokémon
switching in and out as well as attacking
                                            if none of those Rendering Variables
are active it will only show the pokemon sprite */}
                                        <AnimatePresence mode="wait">
                                            {isSwitchingOpponent &&
                                                switchingOutOpponent?.id ===
displayedOpponent?.id &&
                                                switchDirectionOpponent === "out"
? (
                                                <motion.div
                                                    key="opponent-out"
                                                    initial={{ opacity: 1, y: 0 }}
                                                    animate={{ opacity: 0, y: -50
}}
                                                    exit={{ opacity: 0 }}
                                                    transition={{ duration: 0.5 }}
                                                    className="mt-8"
                                                >
                                                    <img
                                                        src=
{displayedOpponent?.sprite || "/placeholder.svg"}
                                                        alt=
{displayedOpponent?.name}
                                                        className="w-32 h-32
object-contain drop-shadow-2xl"
                                                    />
                                                </motion.div>
                                            ) : isSwitchingOpponent &&
                                                switchDirectionOpponent === "in"
&&
                                                switchingInOpponent?.id ===
displayedOpponent?.id ? (
                                                <motion.div
                                                    key="opponent-in"
                                                    initial={{ opacity: 0, scale:
0.5 }}
                                                    animate={{ opacity: 1, scale:
1 }}
```

```
                                                    exit={{ opacity: 1 }}
                                                    transition={{ duration: 0.5 }}
                                                    className="mt-8"
                                                >
                                                    <img
                                                        src=
{displayedOpponent?.sprite || "/placeholder.svg"}
                                                        alt=
{displayedOpponent?.name}
                                                        className="w-32 h-32
object-contain drop-shadow-2xl"
                                                    />
                                                </motion.div>
                                            ) : (
                                                <motion.div
                                                    key="opponent-normal"
                                                    initial={{ opacity: 1 }}
                                                    animate={{
                                                        y: isAnimating &&
!isPlayerTurn ? [0, -10, 0] : 0,
                                                    }}
                                                    transition={{
                                                        duration: 0.5,
                                                        repeat: isAnimating &&
!isPlayerTurn ? 1 : 0,
                                                    }}
                                                    className="mt-8"
                                                >
                                                    <img
                                                        src=
{displayedOpponent?.sprite || "/placeholder.svg"}
                                                        alt=
{displayedOpponent?.name}
                                                        className="w-32 h-32
object-contain drop-shadow-2xl"
                                                    />
                                                </motion.div>
                                            )}
                                        </AnimatePresence>

                                        {/* Display the opponent's Pokémon card
with HP and types */}
                                        <Card className="mb-2 bg-white/95 dark:bg-
slate-800/95 backdrop-blur-sm border-0 shadow-lg">
                                            <CardContent className="p-3">
                                                <div className="flex items-center
gap-2 mb-1">
                                                    <div>
                                                        {/* Display the opponent's
Pokémon name, level, and types */}
                                                        <div className="font-bold
text-base">{displayedOpponent?.name}</div>
                                                        <div className="text-xs
text-muted-foreground">
```

```
                                                            Lv.
{displayedOpponent?.lvl}
                                                        </div>
                                                    </div>
                                                    <div className="flex gap-2">

{displayedOpponent?.types.map((type) => (
                                                            <Badge
                                                                key={type}
                                                                className=
{`${getTypeColor(type)} text-white border-0 shadow-md`}
                                                            >
                                                                {type}
                                                            </Badge>
                                                        ))}
                                                    </div>
                                                </div>

                                                <div className="w-40">
                                                    <div className="flex justify-
between text-xs mb-1">
                                                        {/* Display the opponent's
HP with current and max HP */}
                                                        <span>HP</span>
                                                        <span>

{displayedCurrentHP}/{displayedMaxHP}
                                                        </span>
                                                    </div>
                                                    <Progress
                                                        value={
                                                            ((displayedCurrentHP)
/ (displayedMaxHP)) * 100
                                                        }
                                                        className="h-2 shadow-sm"
                                                        indicatorClassName=
{getHPColor(

displayedOpponent?.currentHP ?? 0,

displayedOpponent?.maxHP ?? 1
                                                        )}
                                                    />
                                                </div>
                                            </CardContent>
                                        </Card>
                                    </div>
                                </div>

                                {/* Player Pokemon */}
                                <div className="relative z-10 flex justify-start
pl-6 pb-6">
                                    <div className="flex items-center gap-3">
                                        <Card className="bg-white/95 dark:bg-
```

```
slate-800/95 backdrop-blur-sm border-0 shadow-lg">
                                            <CardContent className="p-3">
                                                <div className="flex items-center
gap-2 mb-1">
                                                    <div className="flex gap-2">
                                                        {/* Display the player's
Pokémon types as badges */}

{playerPokemon.types.map((type) => (
                                                            <Badge key={type}
className={`${getTypeColor(type)} text-white border-0 shadow-md`}>
                                                                {type}
                                                            </Badge>
                                                        ))}
                                                    </div>
                                                    <div>
                                                        {/* Display the player's
Pokémon name and level */}
                                                        <div className="font-bold
text-base">{playerPokemon.name}</div>
                                                        <div className="text-xs
text-muted-foreground">Lv.{playerPokemon.lvl}</div>
                                                    </div>
                                                </div>
                                                <div className="w-40">
                                                    <div className="flex justify-
between text-xs mb-1">
                                                        {/* Display the player's
Pokémon HP with current and max HP */}
                                                        <span>HP</span>
                                                        <span>

{playerPokemon.currentHP}/{playerPokemon.maxHP}
                                                        </span>
                                                    </div>
                                                    <Progress
                                                        value=
{(playerPokemon.currentHP / playerPokemon.maxHP) * 100}
                                                        className="h-2 shadow-sm"
                                                        indicatorClassName=
{getHPColor(playerPokemon.currentHP, playerPokemon.maxHP)}
                                                    />
                                                </div>
                                            </CardContent>
                                        </Card>
                                        {/* Animate the player's Pokémon switching
in and out as well as attacking
                                            if none of those Rendering Variables
are active it will only show the pokemon sprite */}
                                        <AnimatePresence mode="wait">
                                            {isSwitchingPlayer &&
switchingOutPlayer?.id === playerPokemon.id && switchDirectionPlayer === "out" ? (
                                                <motion.div
                                                    key="player-out"
```

```
                                                          initial={{ opacity: 1, y: 0 }}
                                                          animate={{ opacity: 0, y: 50
}}
                                                          exit={{ opacity: 0 }}
                                                          transition={{ duration: 0.5 }}
                                                      >
                                                          <img
                                                              src=
{playerPokemon.sprite_back || "/placeholder.svg"}
                                                              alt={playerPokemon.name}
                                                              className="w-32 h-32
object-contain drop-shadow-2xl"
                                                          />
                                                      </motion.div>
                                                  ) : isSwitchingPlayer &&
                                                      switchingInPlayer &&
                                                      switchDirectionPlayer === "in" &&
                                                      switchingOutPlayer?.id ===
playerPokemon.id ? (
                                                      <motion.div
                                                          key="player-in"
                                                          initial={{ opacity: 0, scale:
0.5 }}
                                                          animate={{ opacity: 1, scale:
1 }}
                                                          exit={{ opacity: 1 }}
                                                          transition={{ duration: 0.5 }}
                                                      >
                                                          <img
                                                              src=
{switchingInPlayer.sprite_back || "/placeholder.svg"}
                                                              alt=
{switchingInPlayer.name}
                                                              className="w-32 h-32
object-contain drop-shadow-2xl"
                                                          />
                                                      </motion.div>
                                                  ) : (
                                                      <motion.div
                                                          key="player-normal"
                                                          initial={{ opacity: 1 }}
                                                          animate={{
                                                              y: isAnimating &&
isPlayerTurn ? [0, -10, 0] : 0,
                                                          }}
                                                          transition={{
                                                              duration: 0.5,
                                                              repeat: isAnimating &&
isPlayerTurn ? 1 : 0,
                                                          }}
                                                      >
                                                          <img
                                                              src=
{playerPokemon.sprite_back || "/placeholder.svg"}
```

```
                                                                 alt={playerPokemon.name}
                                                                 className="w-32 h-32
object-contain drop-shadow-2xl"
                                                             />
                                                         </motion.div>
                                                     )}
                                                 </AnimatePresence>
                                             </div>
                                         </div>
                                     </div>
                                 </CardContent>
                             </Card>

                             {/* Battle Interface with Team Status */}
                             <div className="relative">
                                 {/* Player Team Status - Right Side */}
                                 <div className="absolute -right-20 top-4 z-10">
                                     <Card className="bg-white/90 dark:bg-slate-800/90
backdrop-blur-sm border-0 shadow-lg">
                                         <CardContent className="p-2">
                                             <div className="text-xs text-muted-foreground
mb-1 text-center">Your Team</div>
                                             <div className="flex flex-col gap-1">
                                                 {/* Display for each Member on player's
team a Pokeball
                                                     that greys out when the representativ
Pokemon fainted */}
                                                 {playerTeam.map((pokemon, index) => (
                                                     <span
                                                         key={index}
                                                         onClick={() => {
                                                         }}
                                                         className="inline-block"
                                                         style={{ cursor: pokemon.currentHP
> 0 && pokemon.id !== playerPokemon.id ? "pointer" : "default" }}
                                                     >
                                                         <SimplePokeballIcon
                                                             size={18}
                                                             className={`transition-all
duration-300 ${pokemon.currentHP > 0
                                                                 ? pokemon.id ===
playerPokemon.id
                                                                     ? "text-green-500
scale-110"
                                                                     : "text-red-500
hover:scale-110 cursor-pointer"
                                                                 : "text-gray-400
dark:text-gray-600 opacity-50"
                                                                 }`}
                                                         />
                                                     </span>
                                                 ))}
                                             </div>
                                         </CardContent>
```

```
                            </Card>
                        </div>

                        <Card className="shadow-xl border-0 bg-white/90 dark:bg-
slate-800/90 backdrop-blur-sm">
                            <CardContent className="p-4">
                                {/* Battle Text */}
                                <Card className="bg-slate-100/80 dark:bg-slate-
700/80 backdrop-blur-sm border-0 shadow-sm mb-4">
                                    <CardContent className="py-2 px-3">
                                        {/* Display the battle text, which updates
based on the current action */}
                                        <p className="text-base font-medium min-h-
[24px]">{battleText}</p>
                                    </CardContent>
                                </Card>

                                {/* Main Menu */}
                                {isPlayerTurn &&
                                    !isAnimating &&
                                    !isSwitchingPlayer &&
                                    playerPokemon.currentHP > 0 &&
                                    opponentPokemon.currentHP > 0 &&
                                    currentMenu === "main" && (
                                        <div className="grid grid-cols-2 gap-3">
                                            { /* Main menu with options for
attack, Pokémon, items, and run */}
                                            <Button
                                                onClick={() =>
handleMenuSelect("attack")}
                                                variant="outline"
                                                className="h-14 text-left flex
items-center justify-start gap-3 hover:bg-blue-50/80 dark:hover:bg-slate-700/80
border-slate-200 dark:border-slate-600 bg-white/60 dark:bg-slate-800/60 backdrop-
blur-sm shadow-md hover:shadow-lg transition-all duration-200"
                                            >
                                                <div className="p-2 rounded-lg bg-
gradient-to-r from-red-500 to-orange-500 shadow-sm">
                                                    <Swords className="h-4 w-4
text-white" />
                                                </div>
                                                <span className="font-semibold
text-base">Attack</span>
                                            </Button>

                                            <Button
                                                onClick={() =>
handleMenuSelect("pokemon")}
                                                variant="outline"
                                                className="h-14 text-left flex
items-center justify-start gap-3 hover:bg-blue-50/80 dark:hover:bg-slate-700/80
border-slate-200 dark:border-slate-600 bg-white/60 dark:bg-slate-800/60 backdrop-
blur-sm shadow-md hover:shadow-lg transition-all duration-200"
                                            >
```

```
                                                <div className="p-2 rounded-lg bg-
gradient-to-r from-green-500 to-emerald-500 shadow-sm">
                                                    <User2 className="h-4 w-4
text-white" />
                                                </div>
                                                <span className="font-semibold
text-base">Pokémon</span>
                                            </Button>

                                            <Button
                                                onClick={() =>
handleMenuSelect("items")}
                                                variant="outline"
                                                className="h-14 text-left flex
items-center justify-start gap-3 hover:bg-blue-50/80 dark:hover:bg-slate-700/80
border-slate-200 dark:border-slate-600 bg-white/60 dark:bg-slate-800/60 backdrop-
blur-sm shadow-md hover:shadow-lg transition-all duration-200"
                                            >
                                                <div className="p-2 rounded-lg bg-
gradient-to-r from-blue-500 to-cyan-500 shadow-sm">
                                                    <Package className="h-4 w-4
text-white" />
                                                </div>
                                                <span className="font-semibold
text-base">Items</span>
                                            </Button>

                                            <Button
                                                onClick={() =>
handleMenuSelect("run")}
                                                variant="outline"
                                                className="h-14 text-left flex
items-center justify-start gap-3 hover:bg-blue-50/80 dark:hover:bg-slate-700/80
border-slate-200 dark:border-slate-600 bg-white/60 dark:bg-slate-800/60 backdrop-
blur-sm shadow-md hover:shadow-lg transition-all duration-200"
                                            >
                                                <div className="p-2 rounded-lg bg-
gradient-to-r from-gray-500 to-slate-600 shadow-sm">
                                                    <LogOut className="h-4 w-4
text-white" />
                                                </div>
                                                <span className="font-semibold
text-base">Run</span>
                                            </Button>
                                        </div>
                                    )}

                                {/* Attack Menu */}
                                {isPlayerTurn &&
                                    !isAnimating &&
                                    !isSwitchingPlayer &&
                                    playerPokemon.currentHP > 0 &&
                                    opponentPokemon.currentHP > 0 &&
                                    currentMenu === "attack" && (
                    <>
                      <div className="grid grid-cols-2 gap-3
mb-3">
                        {/* Display the player's Pokémon
moves with type icons and PP as well as Power */}
                        {playerPokemon.moveset.map((move,
index) => {
                          const iconElement =
getTypeIcon(move.type)
                          const currentPP =
playerPokemon.getCurrentPP(move.name)
                          const isDisabled = currentPP
<= 0

                          return (
                            <Button
                              key={index}
                              onClick={() => {

console.log("Button clicked");

console.log("playerPokemon:", playerPokemon);

console.log("opponentPokemon:", opponentPokemon);
                                const
isPlayerFaster = playerPokemon.speed > opponentPokemon.speed;

console.log("isPlayerFaster:", isPlayerFaster);
                                if (!isDisabled) {
                                  if
(isPlayerFaster) {

console.log("handlePlayerMove aufgerufen");

handlePlayerMove(move);
                                  } else {

console.log("handleOpponentMove aufgerufen");

handleOpponentMove(move);
                                  }
                                }
                              }}
                              variant="outline"
                              disabled={isDisabled}
                              className={`h-14 text-
left flex items-center justify-start gap-3
                                ${isDisabled
                                  ? "opacity-50
cursor-not-allowed"
                                  : "hover:bg-
blue-50/80 dark:hover:bg-slate-700/80"
                                }
                                border-slate-200
dark:border-slate-600 bg-white/60 dark:bg-slate-800/60 backdrop-blur-sm shadow-md
hover:shadow-lg transition-all duration-200`}
                                                >
                                                    <div className={`p-2
rounded-lg ${getTypeColor(move.type)} shadow-sm`}>
                                                        {iconElement}
                                                    </div>
                                                    <div className="flex
flex-col items-start">
                                                        <span
className="font-semibold text-sm">{move.name}</span>
                                                        <span
className="text-xs text-muted-foreground">
                                                            {move.power >
0 ? `${move.power} Power` : "Status"} • {move.type} • {move.damageClass} •
{currentPP}/{move.pp}
                                                        </span>
                                                    </div>
                                                </Button>
                                            )
                                        })}
                                    </div>
                                    {/* Back button to return to the main
menu */}
                                    <Button
                                        onClick={() =>
setCurrentMenu("main")}
                                        variant="outline"
                                        className="w-full text-sm h-8 bg-
slate-100 dark:bg-slate-700 hover:bg-slate-200 dark:hover:bg-slate-600"
                                    >
                                        Back
                                    </Button>
                                </>
                            )}

                        {/* Pokemon Menu */}
                        {!isSwitchingPlayer && opponentPokemon.currentHP >
0 && currentMenu === "pokemon" && (
                            <>
                                {/* Display the player's Pokémon team with
selectable options
                                    Already Fainted Pokemon are displayed
greyed out
                                    The currently selected Pokemon is
greened out */}
                                <div className="grid grid-cols-3 gap-4 mb-
4">
                                    {playerTeam.map((pokemon, index) => {
                                        const isSelectable =
                                            pokemon.currentHP > 0 &&
                                            (playerPokemon.currentHP === 0
|| pokemon.id !== playerPokemon.id);
```

```
                                                    return (
                                                        // Display each Pokémon in the
team with the sprite on top of a Pokeball
                                                        // If the Pokémon is
selectable, it can be clicked to switch and the image scales up on hover
                                                        <motion.div
                                                            key={index}
                                                            whileHover={isSelectable ?
{ scale: 1.05 } : {}}
                                                            onClick={() =>
isSelectable && handlePokemonSelect(pokemon)}
                                                            className={`flex flex-col
items-center justify-center p-3 rounded-lg shadow-md transition-all duration-200
${pokemon.currentHP === 0
                                                                ? "opacity-50 cursor-
not-allowed"
                                                                : pokemon.id ===
playerPokemon.id
                                                                    ? "bg-green-100/80
dark:bg-green-900/30 border border-green-300 dark:border-green-700"
                                                                    : "cursor-pointer
bg-white/80 dark:bg-slate-800/80 hover:bg-blue-50/80 dark:hover:bg-blue-900/20"
                                                                }`}
                                                        >
                                                            <div className="relative
w-20 h-20">
                                                                <SimplePokeballIcon
                                                                    size={80}
                                                                    className=
{pokemon.id === playerPokemon.id ? "text-green-500" : "text-red-500"}
                                                                />
                                                                <img
                                                                    src=
{pokemon.sprite || "/placeholder.svg"}
                                                                    alt={pokemon.name}

className="absolute top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2 w-12 h-12
object-contain"
                                                                />
                                                            </div>
                                                            {/* Display the Pokémon's
name and HP bar */}
                                                            <span className="text-sm
font-medium mt-2">{pokemon.name}</span>
                                                            <div className="w-full mt-
1">
                                                                <Progress
                                                                    value=
{(pokemon.currentHP / pokemon.maxHP) * 100}
                                                                    className={`h-1.5
${(pokemon.currentHP / pokemon.maxHP) > 0.5
                                                                        ? "bg-green-
200 dark:bg-green-800"
                                                                        :
```

```
pokemon.currentHP / pokemon.maxHP > 0.25
                                                                  ? "bg-
yellow-200 dark:bg-yellow-800"
                                                                  : "bg-red-
200 dark:bg-red-800"
                                                              }`}
                                                      />
                                                  </div>
                                                  <span className="text-xs
text-muted-foreground mt-1">

{pokemon.currentHP}/{pokemon.maxHP} HP
                                                  </span>
                                              </motion.div>
                                          );
                                      })}
                                  </div>
                                  {/* Back button to return to the main menu
*/}
                                  <Button
                                      onClick={() => {
                                          if (playerPokemon.currentHP > 0) {
                                              setCurrentMenu("main")
                                          }
                                      }}
                                      variant="outline"
                                      className="w-full text-sm h-8 bg-
slate-100 dark:bg-slate-700 hover:bg-slate-200 dark:hover:bg-slate-600"
                                      disabled={playerPokemon.currentHP ===
0}
                                  >
                                      {playerPokemon.currentHP === 0 ?
"Choose a Pokémon" : "Back"}
                                  </Button>
                              </>
                          )}

                          {/* Run Confirmation Menu */}
                          {/* Display Menu to make sure that the player
wants to run away */}
                          {isPlayerTurn &&
                              !isAnimating &&
                              !isSwitchingPlayer &&
                              playerPokemon.currentHP > 0 &&
                              opponentPokemon.currentHP > 0 &&
                              showRunConfirmation && (
                                  <div className="space-y-4">
                                      <div className="text-center">
                                          <p className="text-lg font-
semibold mb-4">Are you sure you want to give up?</p>
                                      </div>
                                      <div className="grid grid-cols-2 gap-
3">
                                          { /* Buttons to confirm or cancel
```

```
the run action
                                                     Confirmation ends the battle
while cancellation returns to the main menu */
                                                }

                                                <Button
                                                    onClick={handleRunConfirm}
                                                    variant="outline"
                                                    className="h-14 text-center
flex items-center justify-center gap-3 hover:bg-red-50/80 dark:hover:bg-red-900/20
border-red-200 dark:border-red-600 bg-white/60 dark:bg-slate-800/60 backdrop-blur-
sm shadow-md hover:shadow-lg transition-all duration-200"
                                                >
                                                    <div className="p-2 rounded-lg
bg-gradient-to-r from-red-500 to-red-600 shadow-sm">
                                                        <LogOut className="h-4 w-4
text-white" />
                                                    </div>
                                                    <span className="font-semibold
text-base text-red-600 dark:text-red-400">Yes, give up</span>
                                                </Button>
                                                <Button
                                                    onClick={handleRunCancel}
                                                    variant="outline"
                                                    className="h-14 text-center
flex items-center justify-center gap-3 hover:bg-green-50/80 dark:hover:bg-green-
900/20 border-green-200 dark:border-green-600 bg-white/60 dark:bg-slate-800/60
backdrop-blur-sm shadow-md hover:shadow-lg transition-all duration-200"
                                                >
                                                    <div className="p-2 rounded-lg
bg-gradient-to-r from-green-500 to-green-600 shadow-sm">
                                                        <Shield className="h-4 w-4
text-white" />
                                                    </div>
                                                    <span className="font-semibold
text-base text-green-600 dark:text-green-400">
                                                        No, keep fighting
                                                    </span>
                                                </Button>
                                            </div>
                                        </div>
                                    )}

                                {/* Waiting State */}
                                {/* Display a waiting state when it's not the
player's turn or when an animation is in progress */}
                                {(!isPlayerTurn || isAnimating ||
isSwitchingPlayer) &&
                                    playerPokemon.currentHP > 0 &&
                                    opponentPokemon.currentHP > 0 &&
                                    !showRunConfirmation &&
                                    currentMenu !== "pokemon" && (
                                        <div className="text-center py-6">
                                            <div className="inline-flex items-
```

```
center gap-2 text-muted-foreground">
                                                <div className="w-2 h-2 bg-blue-
500 rounded-full animate-pulse"></div>
                                                <div className="w-2 h-2 bg-blue-
500 rounded-full animate-pulse delay-75"></div>
                                                <div className="w-2 h-2 bg-blue-
500 rounded-full animate-pulse delay-150"></div>
                                                <span className="ml-2">
                                                    {isSwitchingPlayer
                                                        ? "Switching Pokémon..."
                                                        : isAnimating
                                                            ? "Battle in
progress..."
                                                            : "Opponent's
turn..."}
                                                </span>
                                            </div>
                                        </div>
                                    )}

                                {/* Game Over State */}
                                {/* Display a button to return to the teambuilder
if either team has no Pokémon left */}
                                {(playerTeam.every((p) => p.currentHP === 0) ||
opponentTeam.every((p) => p.currentHP === 0)) && (
                                    <div className="text-center">
                                        <Button
                                            onClick={() => onEndofBattle()}
                                            className="bg-gradient-to-r from-
green-500 to-emerald-600 hover:from-green-600 hover:to-emerald-700 text-white
border-0 shadow-lg hover:shadow-xl transition-all duration-200 px-6 py-2 text-
base"
                                        >
                                            <ArrowLeft className="h-4 w-4 mr-2" />
                                            Back to Teambuilder
                                        </Button>
                                    </div>
                                )}
                            </CardContent>
                        </Card>
                    </div>
                </div>
            </div>
        </div >
    )
}
```