

```

"use client"

import { useState } from "react"
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@components/ui/tabs"
import { Button } from "@components/ui/button"
import { Input } from "@components/ui/input"
import { ScrollArea } from "@components/ui/scroll-area"
import { Card, CardContent } from "@components/ui/card"
import { Search, Save, Download, Zap } from "lucide-react"
import PokemonCard from "@components/pokemon-card"
import TeamPreview from "@components/team-preview"
import PokemonDetails from "@components/pokemon-details"
import { Pokemon } from "@lib/types"
import PokemonBattler from "../pokemon-battler"
import { mockPokemonList } from "@lib/mock-data"
import { fetchSpecificPokemonData } from "@lib/utils"

/**
 * TeamBuilder component for constructing and managing a Pokémon team.
 * @param pokemonList - The list of available Pokémon to choose from.
 * @returns A React component for building a Pokémon team or managing an existing
one.
 */
export default function TeamBuilder({ pokemonList }: { pokemonList: Pokemon[] }) {

  // State to manage the Pokémon team, search query, selected Pokémon, and battle
mode
  const [team, setTeam] = useState<any[]>([])
  const [searchQuery, setSearchQuery] = useState("")
  const [selectedPokemon, setSelectedPokemon] = useState<Pokemon | null>(null)
  const [battleMode, setBattleMode] = useState(false)

  // Filter the Pokémon list based on the search query
  const filteredPokemon = pokemonList.filter(
    (pokemon) =>
      pokemon.name.toLowerCase().includes(searchQuery.toLowerCase().replace(" ",
"")) ||
      pokemon.types.some((type) =>
type.toLowerCase().includes(searchQuery.toLowerCase()))
  )

  // Function to add a Pokémon to the team
  const addToTeam = async (pokemon: Pokemon) => {
    // If the Pokémon has only one move with the name "Error", fetch its specific
data
    // This is a placeholder check to ensure the Pokémon has valid data before
adding it to the team
    if (
      pokemon.moves.length === 1 &&
      pokemon.moves[0].name === "Error"
    ) {

```

```

    await fetchSpecificPokemonData(pokemon)
  }

  if (team.length < 6) {
    const clone = pokemon.clone()
    ; (clone as any).id = crypto.randomUUID()
    setTeam([...team, clone])
  }
}

// Function to remove a Pokémon from the team
const removeFromTeam = (pokemonToRemove: Pokemon) => {
  const index = team.findIndex((pokemon) => pokemon === pokemonToRemove)
  if (index !== -1) {
    const newTeam = [...team]
    newTeam.splice(index, 1)
    setTeam(newTeam)
  }
}

// Function to select a Pokémon for details view
const selectPokemon = async (pokemon: Pokemon) => {
  // If the Pokémon has only one move with the name "Error", fetch its specific
data
  // This is a placeholder check to ensure the Pokémon has valid data before
selecting it
  if (
    pokemon.moves.length === 1 &&
    pokemon.moves[0].name === "Error"
  ) {
    await fetchSpecificPokemonData(pokemon)
  }
  setSelectedPokemon(pokemon)
}

// Log the team to the console for debugging
console.log(team)
// Based on the battle mode, render either the team builder or the Pokémon
battler
return (
  !battleMode ? (
    <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-4 py-8">
        <div className="text-center mb-8">
          <h1 className="text-3xl font-bold mb-2">Pokémon Team Builder</h1>
          <p className="text-muted-foreground">Build your ultimate Pokémon
team</p>
        </div>

        <div className="grid grid-cols-1 lg:grid-cols-12 gap-6">
          <Card className="lg:col-span-8 shadow-lg border-0 bg-white/80 dark:bg-
slate-800/80 backdrop-blur-sm">

```

```

    <CardContent className="p-6">
      <Tabs defaultValue="pokemon" className="w-full">
        <TabsList className="grid grid-cols-1 mb-6 bg-slate-100 dark:bg-
slate-700">
          <TabsTrigger value="pokemon" className="text-sm font-medium">
            Pokémon Collection
          </TabsTrigger>
        </TabsList>

        { /* Pokémon tab content with search and filter functionality
*/}

        <TabsContent value="pokemon" className="space-y-4">
          <div className="flex items-center space-x-3">
            <div className="relative flex-1">
              <Search className="absolute left-3 top-2.5 h-4 w-4 text-
muted-foreground" />
              <Input
                placeholder="Search Pokémon or type..."
                className="pl-10 h-10 border-slate-200 dark:border-
slate-600 bg-white dark:bg-slate-800"
                value={searchQuery}
                onChange={(e) => setSearchQuery(e.target.value)}
              />
            </div>
            <Button variant="outline" size="sm">
              Filter
            </Button>
          </div>
          { /* Display the filtered Pokémon in a scrollable area */}
          <ScrollArea className="h-[500px] rounded-lg border border-
slate-200 dark:border-slate-600 bg-white dark:bg-slate-800">
            <div className="grid grid-cols-2 sm:grid-cols-3 md:grid-
cols-4 gap-4 p-6">
              {filteredPokemon.map((pokemon) => (
                <PokemonCard
                  key={pokemon.pdx_num}
                  pokemon={pokemon}
                  onAdd={() => addToTeam(pokemon)}
                  onSelect={() => selectPokemon(pokemon)}
                />
              ))}
            </div>
          </ScrollArea>
        </TabsContent>
      </Tabs>
    </CardContent>
  </Card>

  { /* Team preview and actions section */}
  <div className="lg:col-span-4 space-y-6">
    <Card className="shadow-md">
      <CardContent className="p-6">
        <div className="flex items-center justify-between mb-4">
          <h3 className="text-lg font-medium">Your Team</h3>

```

```

    <div className="flex space-x-2">
      { /* Button to save the current team configuration
       * or load a saved team configuration.
       * This is a placeholder for future functionality.
       */ }
      <Button variant="outline" size="sm"
        onClick={() => { }}>
        <Save className="h-4 w-4 mr-1" />
        Save
      </Button>
      <Button variant="outline" size="sm"
        onClick={() => { }}>
        <Download className="h-4 w-4 mr-1" />
        Load
      </Button>
    </div>
  </div>

  { /* Display the team preview with the selected Pokémon */ }
  <TeamPreview team={team} onRemove={removeFromTeam} onSelect=
{selectPokemon} />

  <div className="mt-6 space-y-3">
    <div className="flex items-center justify-between">
      <p className="text-sm text-muted-foreground">{team.length}/6
Pokémon selected</p>
      <div className="w-24 h-2 bg-slate-200 dark:bg-slate-700
rounded-full overflow-hidden">
        <div
          className="h-full bg-gradient-to-r from-blue-500 to-
purple-500 transition-all duration-300"
          style={{ width: `${(team.length / 6) * 100}%` }}
        />
      </div>
    </div>

    { /* Button to start the battle, disabled if no Pokémon are
selected */ }

    <Button
      variant="default"
      size="default"
      disabled={team.length === 0}
      className="w-full bg-gradient-to-r from-red-500 to-red-600
hover:from-red-600 hover:to-red-700 text-white border-0"
      onClick={() => setBattleMode(true)}
    >
      <Zap className="h-4 w-4 mr-2" />
      Start Battle
    </Button>
  </div>
</CardContent>
</Card>
{ /* Display selected Pokémon details if any */ }
{selectedPokemon && (

```

```
        <Card className="shadow-md">
          <CardContent className="p-6">
            <PokemonDetails pokemon={selectedPokemon} />
          </CardContent>
        </Card>
      )}
    </div>
  </div>
</div>
) : (
  // Render the Pokémon battler component when in battle mode
  <PokemonBattler FullUserTeam={team}
    FullOpponentTeam={[mockPokemonList[9], mockPokemonList[6],
mockPokemonList[3], mockPokemonList[1], mockPokemonList[0], mockPokemonList[4]]}
    onEndofBattle={() => { setBattleMode(false); }}
  />
)
)
}
```