```
import React, { useState, useEffect } from "react";
import { base44 } from "@/api/base44Client";
import { useQuery, useMutation, useQueryClient } from
"@tanstack/react-query";
import { Button } from "@/components/ui/button";
import { Card, CardContent, CardHeader, CardTitle } from
"@/components/ui/card";
import { Input } from "@/components/ui/input";
import { Label } from "@/components/ui/label";
import { Textarea } from "@/components/ui/textarea";
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue
} from "@/components/ui/select";

import { Badge } from "@/components/ui/badge";
import { format } from "date-fns";
import { Smile, Frown, Meh, TrendingUp, Calendar, Activity, Moon, Zap }
from "lucide-react";
import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip,
ResponsiveContainer, BarChart, Bar } from "recharts";

const moodEmojis = {
 excellent: { emoji: "😄", color: "text-green-500", bg: "bg-green-100"
},
 good: { emoji: "🙂", color: "text-blue-500", bg: "bg-blue-100" },
 okay: { emoji: "😐", color: "text-yellow-500", bg: "bg-yellow-100" },
 low: { emoji: "😕", color: "text-orange-500", bg: "bg-orange-100" },
 struggling: { emoji: "😢", color: "text-red-500", bg: "bg-red-100" }
};

export default function MoodTracking() {
 const [showForm, setShowForm] = useState(false);
 const [user, setUser] = useState(null);
 const queryClient = useQueryClient();

 const [formData, setFormData] = useState({
   date: format(new Date(), "yyyy-MM-dd"),
   mood: "okay",
   stress_level: 5,
   energy_level: 5,
   sleep_quality: 5,
   notes: "",
   activities: []
 });
```

```
useEffect(() => {
  const fetchUser = async () => {
    try {
      const userData = await base44.auth.me();
      setUser(userData);
    } catch (error) {
      base44.auth.redirectToLogin();
    }
  };
  fetchUser();
}, []);

const { data: moodEntries = [], isLoading } = useQuery({
  queryKey: ['moodEntries'],
  queryFn: async () => {
    const entries = await base44.entities.MoodEntry.filter({
created_by: user?.email }, '-date');
    return entries;
  },
  enabled: !!user
});

const createMoodMutation = useMutation({
  mutationFn: (data) => base44.entities.MoodEntry.create(data),
  onSuccess: () => {
    queryClient.invalidateQueries(['moodEntries']);
    setShowForm(false);
    resetForm();
  }
});

const resetForm = () => {
  setFormData({
    date: format(new Date(), "yyyy-MM-dd"),
    mood: "okay",
    stress_level: 5,
    energy_level: 5,
    sleep_quality: 5,
    notes: "",
    activities: []
  });
};
```

```
 const handleSubmit = (e) => {
   e.preventDefault();
   createMoodMutation.mutate(formData);
 };

 const chartData = moodEntries.slice(0, 7).reverse().map(entry => ({
   date: format(new Date(entry.date), 'MMM dd'),
   stress: entry.stress_level,
   energy: entry.energy_level,
   sleep: entry.sleep_quality
 }));

 const moodDistribution = Object.keys(moodEmojis).map(mood => ({
   mood: mood.charAt(0).toUpperCase() + mood.slice(1),
   count: moodEntries.filter(e => e.mood === mood).length
 }));

 const averages = {
   stress: (moodEntries.reduce((sum, e) => sum + (e.stress_level || 0),
0) / (moodEntries.length || 1)).toFixed(1),
   energy: (moodEntries.reduce((sum, e) => sum + (e.energy_level || 0),
0) / (moodEntries.length || 1)).toFixed(1),
   sleep: (moodEntries.reduce((sum, e) => sum + (e.sleep_quality || 0),
0) / (moodEntries.length || 1)).toFixed(1)
 };

 if (!user) return null;

 return (
   <div className="py-12 px-4">
     <div className="max-w-7xl mx-auto space-y-8">
       {/* Header */}
       <div className="flex flex-col md:flex-row justify-between
items-start md:items-center gap-4">
         <div>
           <h1 className="text-4xl font-bold bg-gradient-to-r
from-purple-600 to-teal-600 bg-clip-text text-transparent">
             Mood Tracking Dashboard
           </h1>
           <p className="text-gray-600 mt-2">Track your emotional
well-being and identify patterns</p>
         </div>
```

```
          <Button
            onClick={() => setShowForm(!showForm)}
            className="bg-gradient-to-r from-purple-500 to-teal-500
text-white shadow-lg"
          >
            {showForm ? "Cancel" : "+ Log Today's Mood"}
          </Button>
        </div>

        {/* Mood Entry Form */}
        {showForm && (
          <Card className="border-none shadow-xl">
            <CardHeader>
              <CardTitle>How are you feeling today?</CardTitle>
            </CardHeader>
            <CardContent>
              <form onSubmit={handleSubmit} className="space-y-6">
                <div className="grid md:grid-cols-2 gap-6">
                  <div className="space-y-2">
                    <Label>Date</Label>
                    <Input
                      type="date"
                      value={formData.date}
                      onChange={(e) => setFormData({ ...formData, date:
e.target.value })}
                      required
                    />
                  </div>

                  <div className="space-y-2">
                    <Label>Overall Mood</Label>
                    <Select value={formData.mood} onValueChange={(value)
=> setFormData({ ...formData, mood: value })}>
                      <SelectTrigger>
                        <SelectValue />
                      </SelectTrigger>
                      <SelectContent>
                        {Object.entries(moodEmojis).map(([key, { emoji
}]) => (
                          <SelectItem key={key} value={key}>
                            {emoji} {key.charAt(0).toUpperCase() +
key.slice(1)}
                          </SelectItem>
                        ))}
                      </SelectContent>
                    </Select>
                  </div>
                </div>

                <div className="space-y-4">
                  <div>
                    <Label>Stress Level:
{formData.stress_level}/10</Label>
                    <Input
                      type="range"
                      value={formData.stress_level}
                      onChange={(e) => setFormData({ ...formData,
stress_level: Number(e.target.value) })}
                      min={1}
                      max={10}
                      step={1}
                      className="mt-2"
                    />
                  </div>

                  <div>
                    <Label>Energy Level:
{formData.energy_level}/10</Label>
                    <Input
                      type="range"
                      value={formData.energy_level}
                      onChange={(e) => setFormData({ ...formData,
energy_level: Number(e.target.value) })}
```