```
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 { BookOpen, Plus, Search, Calendar, Tag, Lock, Pencil, Trash2 }
from "lucide-react";

const moodTags = {
 grateful: { label: "Grateful", color: "bg-green-100 text-green-800",
icon: "🙏" },
 reflective: { label: "Reflective", color: "bg-blue-100 text-blue-800",
icon: "💭" },
 challenging: { label: "Challenging", color: "bg-red-100 text-red-800",
icon: "💪" },
 hopeful: { label: "Hopeful", color: "bg-yellow-100 text-yellow-800",
icon: "🌟" },
 anxious: { label: "Anxious", color: "bg-orange-100 text-orange-800",
icon: "😰" },
 peaceful: { label: "Peaceful", color: "bg-teal-100 text-teal-800",
icon: "☮️" },
 other: { label: "Other", color: "bg-gray-100 text-gray-800", icon:
"📝" }
};

export default function Journal() {
 const [showForm, setShowForm] = useState(false);
 const [user, setUser] = useState(null);
 const [editingEntry, setEditingEntry] = useState(null);
 const [searchQuery, setSearchQuery] = useState("");
 const [selectedTag, setSelectedTag] = useState("all");
 const queryClient = useQueryClient();

 const [formData, setFormData] = useState({
```

```
    title: "",
    content: "",
    date: format(new Date(), "yyyy-MM-dd"),
    mood_tag: "reflective",
    is_private: true
  });

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

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

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

  const updateEntryMutation = useMutation({
    mutationFn: ({ id, data }) =>
base44.entities.JournalEntry.update(id, data),
    onSuccess: () => {
      queryClient.invalidateQueries(['journalEntries']);
```

```
      setEditingEntry(null);
      setShowForm(false);
      resetForm();
    }
  });

  const deleteEntryMutation = useMutation({
    mutationFn: (id) => base44.entities.JournalEntry.delete(id),
    onSuccess: () => {
      queryClient.invalidateQueries(['journalEntries']);
    }
  });

  const resetForm = () => {
    setFormData({
      title: "",
      content: "",
      date: format(new Date(), "yyyy-MM-dd"),
      mood_tag: "reflective",
      is_private: true
    });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    if (editingEntry) {
      updateEntryMutation.mutate({ id: editingEntry.id, data: formData
});
    } else {
      createEntryMutation.mutate(formData);
    }
  };

  const handleEdit = (entry) => {
    setEditingEntry(entry);
    setFormData({
      title: entry.title || "",
      content: entry.content,
      date: entry.date,
      mood_tag: entry.mood_tag,
      is_private: entry.is_private
    });
    setShowForm(true);
```

```
 };

 const filteredEntries = journalEntries.filter(entry => {
   const matchesSearch =
entry.title?.toLowerCase().includes(searchQuery.toLowerCase()) ||

entry.content?.toLowerCase().includes(searchQuery.toLowerCase());
   const matchesTag = selectedTag === "all" || entry.mood_tag ===
selectedTag;
   return matchesSearch && matchesTag;
 });

 if (!user) return null;

 return (
   <div className="py-12 px-4">
     <div className="max-w-6xl 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 flex
items-center gap-3">
             <BookOpen className="w-10 h-10 text-purple-600" />
             My Journal
           </h1>
           <p className="text-gray-600 mt-2">Your private space for
thoughts and reflections</p>
         </div>
         <Button
           onClick={() => {
             setShowForm(!showForm);
             if (showForm) {
               setEditingEntry(null);
               resetForm();
             }
           }}
           className="bg-gradient-to-r from-purple-500 to-teal-500
text-white shadow-lg"
         >
           <Plus className="w-5 h-5 mr-2" />
           {showForm ? "Cancel" : "New Entry"}
```

```
          </Button>
        </div>

        {/* Stats */}
        <div className="grid md:grid-cols-3 gap-6">
          <Card className="border-none shadow-lg">
            <CardContent className="p-6">
              <div className="flex items-center justify-between">
                <div>
                  <p className="text-sm text-gray-500">Total Entries</p>
                  <p className="text-3xl font-bold
text-purple-600">{journalEntries.length}</p>
                </div>
                <BookOpen className="w-10 h-10 text-purple-400" />
              </div>
            </CardContent>
          </Card>

          <Card className="border-none shadow-lg">
            <CardContent className="p-6">
              <div className="flex items-center justify-between">
                <div>
                  <p className="text-sm text-gray-500">This Month</p>
                  <p className="text-3xl font-bold text-teal-600">
                    {journalEntries.filter(e =>
                      new Date(e.date).getMonth() === new
Date().getMonth()
                    ).length}
                  </p>
                </div>
                <Calendar className="w-10 h-10 text-teal-400" />
              </div>
            </C
```