import React, { useState, useEffect, useCallback } from 'react';

import { Button } from '@/components/ui/button';

import { Input } from '@/components/ui/input';

import { Textarea } from '@/components/ui/textarea';

import { Label } from '@/components/ui/label';

import { Card, CardHeader, CardContent, CardFooter } from '@/components/ui/card';

import {

    User,

    LogIn,

    LogOut,

    FileText,

    Mail,

    Edit,

    Save,

    X,

    CheckCircle,

    AlertTriangle,

    Loader2,

    Home,

    ArrowRight

} from 'lucide-react';

import { motion, AnimatePresence } from 'framer-motion';

import { cn } from '@/lib/utils';

// Mock API functions (replace with your actual API calls)

const api = {

    signup: async (userData: any) => {

        // Simulate API call delay

        await new Promise(resolve => setTimeout(resolve, 500));

        if (userData.email === 'test@example.com') {

            throw new Error('Email already taken');

        }

        return { success: true, message: 'Signup successful', user: { ...userData, id: 'user123' } };

    },

    login: async (credentials: any) => {

        await new Promise(resolve => setTimeout(resolve, 500));

        if (credentials.email !== 'test@example.com' || credentials.password !== 'password') {

            throw new Error('Invalid credentials');

        }

        return { success: true, message: 'Login successful', token: 'fake-token', user: { id: 'user123', email: 'test@example.com', name: 'Test User' } };

    },

    logout: async () => {

        await new Promise(resolve => setTimeout(resolve, 200));

        return { success: true, message: 'Logged out' };

    },

    createBlog: async (blogData: any, token: string) => {

        await new Promise(resolve => setTimeout(resolve, 800));

        if (!token) throw new Error('Unauthorized');

        return { success: true, message: 'Blog created', blog: { ...blogData, id: 'blog456', authorId: 'user123', createdAt: new Date() } };

    },

    getBlogs: async () => {

        await new Promise(resolve => setTimeout(resolve, 300));

        return {

            success: true, blogs: [

                { id: 'blog1', title: 'First Blog Post', content: 'This is the content of the first blog post.', author: 'Test User', authorId: 'user123', createdAt: new Date() },

                { id: 'blog2', title: 'Second Blog Post', content: 'This is another blog post.', author: 'Another User', authorId: 'user456', createdAt: new Date() },

            ]

        };

    },

    getBlog: async (id: string) => {

        await new Promise(resolve => setTimeout(resolve, 400));

        if (id === 'nonexistent') throw new Error('Blog not found');

        return {

            success: true, blog: {

                id: id,

                title: 'Sample Blog Post',

                content: 'This is a sample blog post content.  It can contain rich text and formatting.  You can use this space to express your thoughts and ideas.\n\n\*\*Key Features:\*\*\n\n\* Easy to use editor\n\* Markdown support\n\* Save drafts\n\* Publish instantly\n\n```javascript\nconst hello = () => {\n  console.log("Hello, world!");\n};\n```',

                author: 'Test User',

                authorId: 'user123',

                createdAt: new Date()

            }

        };

    },

    getUser: async (id: string) => {

        await new Promise(resolve => setTimeout(resolve, 300));

        if (id === 'nonexistent') throw new Error('User not found');

        return {

            success: true, user: {

                id: id,

                name: 'Test User',

                email: 'test@example.com',

                bio: 'This is a sample bio.',

                profilePicture: null, // You might store a URL here

                joinedAt: new Date(),

            }

        };

    },

    updateUser: async (id: string, userData: any, token: string) => {

        await new Promise(resolve => setTimeout(resolve, 600));

        if (!token) throw new Error('Unauthorized');

        if (id !== 'user123') throw new Error('User not found'); // Only allow updating current user

        return { success: true, message: 'Profile updated', user: { ...userData, id: 'user123' } };

    },

    contact: async (messageData: any) => {

        await new Promise(resolve => setTimeout(resolve, 500));

        console.log('Contact form submitted:', messageData); // Simulate sending email

        return { success: true, message: 'Message sent!' };

    }

};

// Animation Variants

const pageVariants = {

    hidden: { opacity: 0, y: 20 },

    visible: { opacity: 1, y: 0, transition: { duration: 0.5 } },

    exit: { opacity: 0, y: -20, transition: { duration: 0.3 } }

};

// --- Components ---

// Reusable Alert Component

const AlertMessage = ({ message, type, onClose }: { message: string, type: 'success' | 'error' | 'info', onClose: () => void }) => {

    let icon;

    let colorClass;

    switch (type) {

        case 'success':

            icon = <CheckCircle className="h-5 w-5" />;

            colorClass = 'text-green-500 bg-green-100 border-green-300';

            break;

        case 'error':

            icon = <AlertTriangle className="h-5 w-5" />;

            colorClass = 'text-red-500 bg-red-100 border-red-300';

            break;

        default:

            icon = <AlertTriangle className="h-5 w-5" />;

            colorClass = 'text-blue-500 bg-blue-100 border-blue-300';

    }

    return (

        <motion.div

            initial={{ opacity: 0, y: -10 }}

            animate={{ opacity: 1, y: 0 }}

            exit={{ opacity: 0, y: -10 }}

            className={`p-4 rounded-md border shadow-md ${colorClass} flex items-center justify-between`}

        >

            <div className="flex items-center gap-2">

                {icon}

                <p>{message}</p>

            </div>

            <Button variant="ghost" size="icon" onClick={onClose} className="text-opacity-70 hover:text-opacity-100">

                <X className="h-4 w-4" />

            </Button>

        </motion.div>

    );

};

// Loading Indicator

const Loading = () => (

    <div className="flex justify-center items-center h-24">

        <Loader2 className="animate-spin h-8 w-8 text-gray-500" />

    </div>

);

// --- Pages ---

// 1. Landing Page

const LandingPage = ({ onNavigate }: { onNavigate: (page: string) => void }) => {

    return (

        <motion.div

            variants={pageVariants}

            initial="hidden"

            animate="visible"

            exit="exit"

            className="container mx-auto px-4 py-8 text-center"

        >

            <h1 className="text-4xl font-bold text-gray-800 mb-4">Welcome to Our Blog</h1>

            <p className="text-lg text-gray-600 mb-8">

                Discover insightful articles and share your own perspectives.

            </p>

            <div className="flex flex-col sm:flex-row justify-center gap-4">

                <Button onClick={() => onNavigate('blogs')} className="bg-blue-500 hover:bg-blue-600 text-white px-6 py-3 rounded-md">

                    <FileText className="mr-2 h-4 w-4" />

                    Read Blogs

                </Button>

                <Button onClick={() => onNavigate('signup')} className="bg-green-500 hover:bg-green-600 text-white px-6 py-3 rounded-md">

                    <User className="mr-2 h-4 w-4" />

                    Sign Up & Write

                </Button>

            </div>

            <div className="mt-12">

                <h2 className="text-2xl font-semibold text-gray-700 mb-4">Featured Blogs</h2>

                <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">

                    {/\* Mock Blog Cards - Replace with dynamic data \*/}

                    <Card>

                        <CardHeader>

                            <h3 className="text-xl font-semibold">Blog Post 1</h3>

                            <p className="text-gray-500">Author: Test User</p>

                        </CardHeader>

                        <CardContent>

                            <p className="text-gray-700">Summary of the first blog post...</p>

                        </CardContent>

                        <CardFooter>

                            <Button variant="link" onClick={() => onNavigate('blogs/blog1')}>

                                Read More <ArrowRight className="ml-1 h-4 w-4" />

                            </Button>

                        </CardFooter>

                    </Card>

                    <Card>

                        <CardHeader>

                            <h3 className="text-xl font-semibold">Blog Post 2</h3>

                            <p className="text-gray-500">Author: Another User</p>

                        </CardHeader>

                        <CardContent>

                            <p className="text-gray-700">Summary of the second blog post...</p>

                        </CardContent>

                        <CardFooter>

                            <Button variant="link" onClick={() => onNavigate('blogs/blog2')}>

                                Read More <ArrowRight className="ml-1 h-4 w-4" />

                            </Button>

                        </CardFooter>

                    </Card>

                    <Card>

                        <CardHeader>

                            <h3 className="text-xl font-semibold">Blog Post 3</h3>

                            <p className="text-gray-500">Author: Test User</p>

                        </CardHeader>

                        <CardContent>

                            <p className="text-gray-700">Summary of the third blog post...</p>

                        </CardContent>

                        <CardFooter>

                            <Button variant="link" onClick={() => onNavigate('blogs/blog1')}>

                                Read More <ArrowRight className="ml-1 h-4 w-4" />

                            </Button>

                        </CardFooter>

                    </Card>

                </div>

            </div>

        </motion.div>

    );

};

// 2. Sign Up Page

const SignUpPage = ({ onNavigate, onLogin }: { onNavigate: (page: string) => void, onLogin: (user: any, token: string) => void }) => {

    const [name, setName] = useState('');

    const [email, setEmail] = useState('');

    const [password, setPassword] = useState('');

    const [loading, setLoading] = useState(false);

    const [error, setError] = useState<string | null>(null);

    const [success, setSuccess] = useState<string | null>(null);

    const handleSubmit = async (e: React.FormEvent) => {

        e.preventDefault();

        setLoading(true);

        setError(null);

        setSuccess(null);

        try {

            const res = await api.signup({ name, email, password });

            setSuccess(res.message);

            // Simulate auto-login after signup

            onLogin(res.user, 'fake-token'); //  Replace 'fake-token'

            onNavigate('blogs'); // Redirect to blogs page after successful signup and login

        } catch (err: any) {

            setError(err.message);

        } finally {

            setLoading(false);

        }

    };

    return (

        <motion.div

            variants={pageVariants}

            initial="hidden"

            animate="visible"

            exit="exit"

            className="container mx-auto px-4 py-8 flex justify-center items-center"

        >

            <Card className="w-full max-w-md">

                <CardHeader>

                    <h2 className="text-2xl font-semibold text-gray-800">Sign Up</h2>

                </CardHeader>

                <CardContent>

                    <form onSubmit={handleSubmit} className="space-y-6">

                        <div>

                            <Label htmlFor="name" className="block text-sm font-medium text-gray-700">Name</Label>

                            <Input

                                id="name"

                                type="text"

                                value={name}

                                onChange={(e) => setName(e.target.value)}

                                required

                                className="mt-1"

                                placeholder="Your Name"

                            />

                        </div>

                        <div>

                            <Label htmlFor="email" className="block text-sm font-medium text-gray-700">Email</Label>

                            <Input

                                id="email"

                                type="email"

                                value={email}

                                onChange={(e) => setEmail(e.target.value)}

                                required

                                className="mt-1"

                                placeholder="you@example.com"

                            />

                        </div>

                        <div>

                            <Label htmlFor="password" className="block text-sm font-medium text-gray-700">Password</Label>

                            <Input

                                id="password"

                                type="password"

                                value={password}

                                onChange={(e) => setPassword(e.target.value)}

                                required

                                className="mt-1"

                                placeholder="Password"

                            />

                        </div>

                        <Button type="submit" className="w-full" disabled={loading}>

                            {loading ? <><Loader2 className="mr-2 h-4 w-4 animate-spin" /> Please wait</> : 'Sign Up'}

                        </Button>

                    </form>

                    <div className="mt-4 text-center">

                        <Button variant="link" onClick={() => onNavigate('login')}>

                            Already have an account? Log in

                        </Button>

                    </div>

                </CardContent>

                <AnimatePresence>

                    {error && (

                        <AlertMessage message={error} type="error" onClose={() => setError(null)} />

                    )}

                </AnimatePresence>

                <AnimatePresence>

                    {success && (

                        <AlertMessage message={success} type="success" onClose={() => setSuccess(null)} />

                    )}

                </AnimatePresence>

            </Card>

        </motion.div>

    );

};

// 3. Login Page

const LoginPage = ({ onNavigate, onLogin }: { onNavigate: (page: string) => void, onLogin: (user: any, token: string) => void }) => {

    const [email, setEmail] = useState('');

    const [password, setPassword] = useState('');

    const [loading, setLoading] = useState(false);

    const [error, setError] = useState<string | null>(null);

    const handleSubmit = async (e: React.FormEvent) => {

        e.preventDefault();

        setLoading(true);

        setError(null);

        try {

            const res = await api.login({ email, password });

            onLogin(res.user, res.token);

            onNavigate('blogs'); // Redirect to blogs page after successful login

        } catch (err: any) {

            setError(err.message);

        } finally {

            setLoading(false);

        }

    };

    return (

        <motion.div

            variants={pageVariants}

            initial="hidden"

            animate="visible"

            exit="exit"

            className="container mx-auto px-4 py-8 flex justify-center items-center"

        >

            <Card className="w-full max-w-md">

                <CardHeader>

                    <h2 className="text-2xl font-semibold text-gray-800">Log In</h2>

                </CardHeader>

                <CardContent>

                    <form onSubmit={handleSubmit} className="space-y-6">

                        <div>

                            <Label htmlFor="email" className="block text-sm font-medium text-gray-700">Email</Label>

                            <Input

                                id="email"

                                type="email"

                                value={email}

                                onChange={(e) => setEmail(e.target.value)}

                                required

                                className="mt-1"

                                placeholder="you@example.com"

                            />

                        </div>

                        <div>

                            <Label htmlFor="password" className="block text-sm font-medium text-gray-700">Password</Label>

                            <Input

                                id="password"

                                type="password"

                                value={password}

                                onChange={(e) => setPassword(e.target.value)}

                                required

                                className="mt-1"

                                placeholder="Password"

                            />

                        </div>

                        <Button type="submit" className="w-full" disabled={loading}>

                            {loading ? <><Loader2 className="mr-2 h-4 w-4 animate-spin" /> Please wait</> : 'Log In'}

                        </Button>

                    </form>

                    <div className="mt-4 text-center">

                        <Button variant="link" onClick={() => onNavigate('signup')}>

                            Don't have an account? Sign up

                        </Button>

                    </div>

                </CardContent>

                <AnimatePresence>

                    {error && (

                        <AlertMessage message={error} type="error" onClose={() => setError(null)} />

                    )}

                </AnimatePresence>

            </Card>

        </motion.div>

    );

};

// 4. Blog Writing Page

const BlogWritingPage = ({ onNavigate, user, token }: { onNavigate: (page: string) => void, user: any, token: string | null }) => {

    const [title, setTitle] = useState('');

    const [content, setContent] = useState('');

    const [loading, setLoading] = useState(false);

    const [error, setError] = useState<string | null>(null);

    const [success, setSuccess] = useState<string | null>(null);

    const handleSubmit = async (e: React.FormEvent) => {

        e.preventDefault();

        setLoading(true);

        setError(null);

        setSuccess(null);

        if (!token) {

            setError('You must be logged in to create a blog post.');

            setLoading(false);

            return;

        }

        try {

            const res = await api.createBlog({ title, content }, token);

            setSuccess(res.message);

            setTitle('');  // Clear the form

            setContent('');

            onNavigate(`blogs/${res.blog.id}`); // Redirect to the newly created blog post

        } catch (err: any) {

            setError(err.message);

        } finally {

            setLoading(false);

        }

    };

    if (!user) {

        return (

            <motion.div

                variants={pageVariants}

                initial="hidden"

                animate="visible"

                exit="exit"

                className="container mx-auto px-4 py-8 text-center"

            >

                <p className="text-lg text-gray-700 mb-4">

                    Please <Button variant="link" onClick={() => onNavigate('login')}>log in</Button> to write a blog post.

                </p>

            </motion.div>

        );

    }

    return (

        <motion.div

            variants={pageVariants}

            initial="hidden"

            animate="visible"

            exit="exit"

            className="container mx-auto px-4 py-8"

        >

            <h1 className="text-3xl font-semibold text-gray-800 mb-6">Write a New Blog Post</h1>

            <form onSubmit={handleSubmit} className="space-y-6">

                <div>

                    <Label htmlFor="title" className="block text-sm font-medium text-gray-700">Title</Label>

                    <Input

                        id="title"

                        type="text"

                        value={title}

                        onChange={(e) => setTitle(e.target.value)}

                        required

                        className="mt-1"

                        placeholder="Blog Post Title"

                    />

                </div>

                <div>

                    <Label htmlFor="content" className="block text-sm font-medium text-gray-700">Content</Label>

                    <Textarea

                        id="content"

                        value={content}

                        onChange={(e) => setContent(e.target.value)}

                        required

                        className="mt-1 min-h-[300px]"

                        placeholder="Write your blog post content here..."

                    />

                </div>

                <Button type="submit" className="w-full" disabled={loading}>

                    {loading ? <><Loader2 className="mr-2 h-4 w-4 animate-spin" /> Publishing...</> : 'Publish Blog Post'}

                </Button>

            </form>

            <AnimatePresence>

                {error && (

                    <AlertMessage message={error} type="error" onClose={() => setError(null)} />

                )}

            </AnimatePresence>

             <AnimatePresence>

                {success && (

                    <AlertMessage message={success} type="success" onClose={() => setSuccess(null)} />

                )}

            </AnimatePresence>

        </motion.div>

    );

};

// 5. Page to Display Published Blog

const BlogPage = ({ onNavigate, blogId }: { onNavigate: (page: string) => void, blogId: string }) => {

    const [blog, setBlog] = useState<any>(null); // Replace 'any' with a proper type

    const [loading, setLoading] = useState(true);

    const [error, setError] = useState<string | null>(null);

    useEffect(() => {

        const fetchBlog = async () => {

            setLoading(true);

            setError(null);

            try {

                const res = await api.getBlog(blogId);

                setBlog(res.blog);

            } catch (err: any) {

                setError(err.message);

            } finally {

                setLoading(false);

            }

        };

        fetchBlog();

    }, [blogId]);

    if (loading) {

        return (

            <motion.div

                variants={pageVariants}

                initial="hidden"

                animate="visible"

                exit="exit"

                className="container mx-auto px-4 py-8"

            >

                <Loading />

            </motion.div>

        );

    }

    if (error) {

        return (

            <motion.div

                variants={pageVariants}

                initial="hidden"

                animate="visible"

                exit="exit"

                className="container mx-auto px-4 py-8"

            >

                <AlertMessage message={error} type="error" onClose={() => setError(null)} />

            </motion.div>

        );

    }

    if (!blog) {

        return (

            <motion.div

                variants={pageVariants}

                initial="hidden"

                animate="visible"

                exit="exit"

                className="container mx-auto px-4 py-8"

            >

                <p className="text-lg text-gray-700">Blog post not found.</p>

            </motion.div>

        );

    }

    // Basic text formatting (replace with a markdown renderer for real content)

    const formattedContent = blog.content.split('\n\n').map((paragraph: string, index: number) => {

        // Very basic markdown-like formatting (for demo)

        let formattedParagraph = paragraph

            .replace(/\\*\\*(.\*?)\\*\\*/g, '<strong>$1</strong>')  // Bold

            .replace(/\\*(.\*?)\\*/g, '<em>$1</em>')      // Italics

            .replace(/```javascript([\s\S]\*?)```/g, '<pre class="bg-gray-100 p-4 rounded-md overflow-x-auto"><code>$1</code></pre>'); // Code block

        return <p key={index} className="text-gray-700 leading-relaxed" dangerouslySetInnerHTML={{ \_\_html: formattedParagraph }} />;

    });

    return (

        <motion.div

            variants={pageVariants}

            initial="hidden"

            animate="visible"

            exit="exit"

            className="container mx-auto px-4 py-8"

        >

            <h1 className="text-3xl font-bold text-gray-800 mb-4">{blog.title}</h1>

            <p className="text-sm text-gray-500 mb-2">By {blog.author} | {new Date(blog.createdAt).toLocaleDateString()}</p>

            <div className="mt-6 space-y-4">

                {formattedContent}

            </div>

        </motion.div>

    );

};

// 6. Contact Us Page

const ContactUsPage = ({ onNavigate }: { onNavigate: (page: string) => void }) => {

    const [name, setName] = useState('');

    const [email, setEmail] = useState('');

    const [message, setMessage] = useState('');

    const [loading, setLoading] = useState(false);

    const [error, setError] = useState<string |null>(null);

    const [success, setSuccess] = useState<string | null>(null);

    const handleSubmit = async (e: React.FormEvent) => {

        e.preventDefault();

        setLoading(true);

        setError(null);

        setSuccess(null);

        try {

            const res = await api.contact({ name, email, message });

            setSuccess(res.message);

            setName('');

            setEmail('');

            setMessage('');

        } catch (err: any) {

            setError(err.message);

        } finally {

            setLoading(false);

        }

    };

    return (

        <motion.div

            variants={pageVariants}

            initial="hidden"

            animate="visible"

            exit="exit"

            className="container mx-auto px-4 py-8 flex justify-center items-center"

        >

            <Card className="w-full max-w-md">

                <CardHeader>

                    <h2 className="text-2xl font-semibold text-gray-800">Contact Us</h2>

                </CardHeader>

                <CardContent>

                    <form onSubmit={handleSubmit} className="space-y-6">

                        <div>

                            <Label htmlFor="name" className="block text-sm font-medium text-gray-700">Name</Label>

                            <Input

                                id="name"

                                type="text"

                                value={name}

                                onChange={(e) => setName(e.target.value)}

                                required

                                className="mt-1"

                                placeholder="Your Name"

                            />

                        </div>

                        <div>

                            <Label htmlFor="email" className="block text-sm font-medium text-gray-700">Email</Label>

                            <Input

                                id="email"

                                type="email"

                                value={email}

                                onChange={(e) => setEmail(e.target.value)}

                                required

                                className="mt-1"

                                placeholder="you@example.com"

                            />

                        </div>

                        <div>

                            <Label htmlFor="message" className="block text-sm font-medium text-gray-700">Message</Label>

                            <Textarea

                                id="message"

                                value={message}

                                onChange={(e) => setMessage(e.target.value)}

                                required

                                className="mt-1"

                                placeholder="Your message..."

                            />

                        </div>

                        <Button type="submit" className="w-full" disabled={loading}>

                            {loading ? <><Loader2 className="mr-2 h-4 w-4 animate-spin" /> Sending...</> : 'Send Message'}

                        </Button>

                    </form>

                </CardContent>

                <AnimatePresence>

                    {error && (

                        <AlertMessage message={error} type="error" onClose={() => setError(null)} />

                    )}

                </AnimatePresence>

                <AnimatePresence>

                    {success && (

                        <AlertMessage message={success} type="success" onClose={() => setSuccess(null)} />

                    )}

                </AnimatePresence>

            </Card>

        </motion.div>

    );

};

// 7. User Profile Section

const UserProfilePage = ({ onNavigate, user, token }: { onNavigate: (page: string) => void, user: any, token: string | null }) => {

    const [profile, setProfile] = useState<any>(null); // Replace 'any' with a proper type

    const [loading, setLoading] = useState(true);

    const [error, setError] = useState<string | null>(null);

    useEffect(() => {

        const fetchProfile = async () => {

            setLoading(true);

            setError(null);

            if (!user) {

                setError('Please log in to view your profile.');

                setLoading(false);

                return;

            }

            try {

                const res = await api.getUser(user.id);

                setProfile(res.user);

            } catch (err: any) {

                setError(err.message);

            } finally {

                setLoading(false);

            }

        };

        fetchProfile();

    }, [user]);

    if (loading) {

        return (

            <motion.div

                variants={pageVariants}

                initial="hidden"

                animate="visible"

                exit="exit"

                className="container mx-auto px-4 py-8"

            >

                <Loading />

            </motion.div>

        );

    }

    if (error) {

        return (

            <motion.div

                variants={pageVariants}

                initial="hidden"

                animate="visible"

                exit="exit"

                className="container mx-auto px-4 py-8"

            >

                <AlertMessage message={error} type="error" onClose={() => setError(null)} />

            </motion.div>

        );

    }

    if (!profile) {

        return (

            <motion.div

                variants={pageVariants}

                initial="hidden"

                animate="visible"

                exit="exit"

                className="container mx-auto px-4 py-8"

            >

                <p className="text-lg text-gray-700">No profile information available.</p>

            </motion.div>

        );

    }

    return (

        <motion.div

            variants={pageVariants}

            initial="hidden"

            animate="visible"

            exit="exit"

            className="container mx-auto px-4 py-8"

        >

            <h1 className="text-3xl font-semibold text-gray-800 mb-6">Your Profile</h1>

            <Card>

                <CardHeader>

                    <div className="flex items-center gap-4">

                        {profile.profilePicture ? (

                            <img src={profile.profilePicture} alt="Profile" className="h-16 w-16 rounded-full" />

                        ) : (

                            <User className="h-16 w-16 text-gray-500" />

                        )}

                        <div>

                            <h2 className="text-xl font-semibold">{profile.name}</h2>

                            <p className="text-gray-500">Joined: {new Date(profile.joinedAt).toLocaleDateString()}</p>

                        </div>

                    </div>

                </CardHeader>

                <CardContent>

                    <div className="space-y-4">

                        <div>

                            <Label className="block text-sm font-medium text-gray-700">Email</Label>

                            <p className="text-gray-700">{profile.email}</p>

                        </div>

                        <div>

                            <Label className="block text-sm font-medium text-gray-700">Bio</Label>

                            <p className="text-gray-700">{profile.bio || 'No bio available.'}</p>

                        </div>

                    </div>

                </CardContent>

                <CardFooter>

                    <Button onClick={() => onNavigate('profile/edit')} className="w-full">

                        <Edit className="mr-2 h-4 w-4" /> Edit Profile

                    </Button>

                </CardFooter>

            </Card>

        </motion.div>

    );

};

// 8. Profile Update Feature

const ProfileUpdatePage = ({ onNavigate, user, token }: { onNavigate: (page: string) => void, user: any, token: string | null }) => {

    const [name, setName] = useState('');

    const [email, setEmail] = useState('');

    const [bio, setBio] = useState('');

    const [loading, setLoading] = useState(false);

    const [error, setError] = useState<string | null>(null);

    const [success, setSuccess] = useState<string | null>(null);

    useEffect(() => {

        if (user) {

            setName(user.name);

            setEmail(user.email);

            setBio(user.bio || '');

        }

    }, [user]);

    const handleSubmit = async (e: React.FormEvent) => {

        e.preventDefault();

        setLoading(true);

        setError(null);

        setSuccess(null);

        if (!token) {

            setError('You must be logged in to update your profile.');

            setLoading(false);

            return;

        }

        try {

            const res = await api.updateUser(user.id, { name, email, bio }, token);

            setSuccess(res.message);

            onNavigate('profile'); // Redirect to profile page after update

        } catch (err: any) {

            setError(err.message);

        } finally {

            setLoading(false);

        }

    };

    if (!user) {

        return (

            <motion.div

                variants={pageVariants}

                initial="hidden"

                animate="visible"

                exit="exit"

                className="container mx-auto px-4 py-8 text-center"

            >

                <p className="text-lg text-gray-700 mb-4">

                    Please <Button variant="link" onClick={() => onNavigate('login')}>log in</Button> to edit your profile.

                </p>

            </motion.div>

        );

    }

    return (

        <motion.div

            variants={pageVariants}

            initial="hidden"

            animate="visible"

            exit="exit"

            className="container mx-auto px-4 py-8 flex justify-center items-center"

        >

            <Card className="w-full max-w-md">

                <CardHeader>

                    <h2 className="text-2xl font-semibold text-gray-800">Edit Profile</h2>

                </CardHeader>

                <CardContent>

                    <form onSubmit={handleSubmit} className="space-y-6">

                        <div>

                            <Label htmlFor="name" className="block text-sm font-medium text-gray-700">Name</Label>

                            <Input

                                id="name"

                                type="text"

                                value={name}

                                onChange={(e) => setName(e.target.value)}

                                required

                                className="mt-1"

                                placeholder="Your Name"

                            />

                        </div>

                        <div>

                            <Label htmlFor="email" className="block text-sm font-medium text-gray-700">Email</Label>

                            <Input

                                id="email"

                                type="email"

                                value={email}

                                onChange={(e) => setEmail(e.target.value)}

                                required

                                className="mt-1"

                                placeholder="you@example.com"

                            />

                        </div>

                        <div>

                            <Label htmlFor="bio" className="block text-sm font-medium text-gray-700">Bio</Label>

                            <Textarea

                                id="bio"

                                value={bio}

                                onChange={(e) => setBio(e.target.value)}

                                className="mt-1"

                                placeholder="Tell us about yourself..."

                            />

                        </div>

                        <Button type="submit" className="w-full" disabled={loading}>

                            {loading ? <><Loader2 className="mr-2 h-4 w-4 animate-spin" /> Updating...</> : 'Save Profile'}

                        </Button>

                    </form>

                </CardContent>

                <AnimatePresence>

                    {error && (

                        <AlertMessage message={error} type="error" onClose={() => setError(null)} />

                    )}

                </AnimatePresence>

                <AnimatePresence>

                    {success && (

                        <AlertMessage message={success} type="success" onClose={() => setSuccess(null)} />

                    )}

                </AnimatePresence>

            </Card>

        </motion.div>

    );

};

// Main App Component

const App = () => {

    const [currentPage, setCurrentPage] = useState<'home' | 'signup' | 'login' | 'blogs' | 'blogs/:id' | 'contact' | 'profile' | 'profile/edit'>('home');

    const [user, setUser] = useState<any>(null);  // Replace 'any'

    const [token, setToken] = useState<string | null>(null);

    const [blogId, setBlogId] = useState<string>('');

    // Function to navigate between pages

    const navigate = useCallback((page: string, id?: string) => {

        setCurrentPage(page as any); // Type assertion

        if (id) {

            setBlogId(id);

        }

    }, []);

    // Function to handle login

    const handleLogin = (userData: any, userToken: string) => {

        setUser(userData);

        setToken(userToken);

        // Store user data and token (optional)

        localStorage.setItem('user', JSON.stringify(userData));

        localStorage.setItem('token', userToken);

    };

    // Function to handle logout

    const handleLogout = async () => {

        // Invalidate session on the server (optional)

        try {

            await api.logout();

        } catch (error) {

            console.error("Logout error:", error);

        }

        setUser(null);

        setToken(null);

        // Clear stored data

        localStorage.removeItem('user');

        localStorage.removeItem('token');

        setCurrentPage('home');

    };

    // Check for stored session on initial load

    useEffect(() => {

        const storedUser = localStorage.getItem('user');

        const storedToken = localStorage.getItem('token');

        if (storedUser && storedToken) {

            try {

                setUser(JSON.parse(storedUser));

                setToken(storedToken);

            } catch (e) {

                console.error("Error parsing stored user data", e);

                // Clear the invalid data

                localStorage.removeItem('user');

                localStorage.removeItem('token');

            }

        }

    }, []);

    // Render different pages based on state

    const renderPage = () => {

        switch (currentPage) {

            case 'home':

                return <LandingPage onNavigate={navigate} />;

            case 'signup':

                return <SignUpPage onNavigate={navigate} onLogin={handleLogin} />;

            case 'login':

                return <LoginPage onNavigate={navigate} onLogin={handleLogin} />;

            case 'blogs':

                return <BlogWritingPage onNavigate={navigate} user={user} token={token} />;

            case 'blogs/:id':

                return <BlogPage onNavigate={navigate} blogId={blogId} />;

            case 'contact':

                return <ContactUsPage onNavigate={navigate} />;

            case 'profile':

                return <UserProfilePage onNavigate={navigate} user={user} token={token} />;

            case 'profile/edit':

                return <ProfileUpdatePage onNavigate={navigate} user={user} token={token} />;

            default:

                return <LandingPage onNavigate={navigate} />;

        }

    };

    return (

        <div className="min-h-screen bg-gray-100">

            {/\* Simple Navigation (Replace with a styled component) \*/}

            <nav className="bg-white shadow-md py-4 px-6 flex justify-between items-center">

                <Button variant="ghost" onClick={() => navigate('home')} className="text-xl font-semibold text-gray-800">

                    <Home className="mr-2 h-5 w-5 inline-block" />

                    My Blog

                </Button>

                <div className="flex gap-4">

                    {user ? (

                        <>

                            <Button variant="ghost" onClick={() => navigate('profile')} className="flex items-center">

                                <User className="mr-2 h-4 w-4" /> Profile

                            </Button>

                            <Button variant="ghost" onClick={() => navigate('blogs')} className="flex items-center">

                                <FileText className="mr-2 h-4 w-4" /> Write Blog

                            </Button>

                            <Button variant="ghost" onClick={handleLogout} className="flex items-center">

                                <LogOut className="mr-2 h-4 w-4" /> Log Out

                            </Button>

                        </>

                    ) : (

                        <>

                            <Button variant="ghost" onClick={() => navigate('login')} className="flex items-center">

                                <LogIn className="mr-2 h-4 w-4" /> Log In

                            </Button>

                            <Button variant="ghost" onClick={() => navigate('signup')} className="flex items-center">

                                <User className="mr-2 h-4 w-4" /> Sign Up

                            </Button>

                        </>

                    )}

                    <Button variant="ghost" onClick={() => navigate('contact')} className="flex items-center">

                        <Mail className="mr-2 h-4 w-4" /> Contact

                    </Button>

                </div>

            </nav>

            {/\* Page Content \*/}

            <main>

                {renderPage()}

            </main>

            {/\* Footer (Optional) \*/}

            <footer className="bg-gray-200 text-gray-600 text-center py-4 mt-12">

                <p>&copy; {new Date().getFullYear()} My Blog. All rights reserved.</p>

            </footer>

        </div>

    );

};

export default App;