Complete Authentication System

// =============================================================================

// FILE: lib/supabase.ts

// =============================================================================

import { createClient } from '@supabase/supabase-js'

const supabaseUrl = process.env.NEXT\_PUBLIC\_SUPABASE\_URL!

const supabaseAnonKey = process.env.NEXT\_PUBLIC\_SUPABASE\_ANON\_KEY!

export const supabase = createClient(supabaseUrl, supabaseAnonKey)

// =============================================================================

// FILE: contexts/AuthContext.tsx

// =============================================================================

'use client'

import React, { createContext, useContext, useEffect, useState } from 'react'

import { User, Session } from '@supabase/supabase-js'

import { supabase } from '@/lib/supabase'

import { useRouter } from 'next/navigation'

interface AuthContextType {

user: User | null

session: Session | null

loading: boolean

signIn: (email: string, password: string) => Promise<{ error?: string }>

signUp: (email: string, password: string, name: string) => Promise<{ error?: string }>

signOut: () => Promise<void>

updateProfile: (updates: { name?: string }) => Promise<{ error?: string }>

}

const AuthContext = createContext<AuthContextType>({} as AuthContextType)

export const useAuth = () => {

const context = useContext(AuthContext)

if (!context) {

throw new Error('useAuth must be used within an AuthProvider')

}

return context

}

export const AuthProvider: React.FC<{ children: React.ReactNode }> = ({ children }) => {

const [user, setUser] = useState<User | null>(null)

const [session, setSession] = useState<Session | null>(null)

const [loading, setLoading] = useState(true)

const router = useRouter()

useEffect(() => {

// Get initial session

const getInitialSession = async () => {

const { data: { session } } = await supabase.auth.getSession()

setSession(session)

setUser(session?.user ?? null)

setLoading(false)

}

getInitialSession()

// Listen for auth changes

const { data: { subscription } } = supabase.auth.onAuthStateChange(

async (event, session) => {

setSession(session)

setUser(session?.user ?? null)

setLoading(false)

// Handle auth events

if (event === 'SIGNED\_IN') {

router.push('/dashboard')

} else if (event === 'SIGNED\_OUT') {

router.push('/')

}

}

)

return () => subscription.unsubscribe()

}, [router])

const signIn = async (email: string, password: string) => {

try {

const { error } = await supabase.auth.signInWithPassword({

email,

password,

})

if (error) {

return { error: error.message }

}

return {}

} catch (error) {

return { error: 'An unexpected error occurred' }

}

}

const signUp = async (email: string, password: string, name: string) => {

try {

const { data, error } = await supabase.auth.signUp({

email,

password,

options: {

data: {

name: name,

},

},

})

if (error) {

return { error: error.message }

}

// Create user profile in our users table

if (data.user) {

const { error: profileError } = await supabase

.from('users')

.insert([

{

id: data.user.id,

email: email,

name: name,

subscription\_tier: 'starter',

subscription\_status: 'trial',

},

])

if (profileError) {

console.error('Error creating user profile:', profileError)

}

}

return {}

} catch (error) {

return { error: 'An unexpected error occurred' }

}

}

const signOut = async () => {

await supabase.auth.signOut()

}

const updateProfile = async (updates: { name?: string }) => {

try {

if (!user) return { error: 'No user logged in' }

const { error } = await supabase.auth.updateUser({

data: updates,

})

if (error) {

return { error: error.message }

}

// Also update our users table

const { error: dbError } = await supabase

.from('users')

.update(updates)

.eq('id', user.id)

if (dbError) {

return { error: dbError.message }

}

return {}

} catch (error) {

return { error: 'An unexpected error occurred' }

}

}

const value = {

user,

session,

loading,

signIn,

signUp,

signOut,

updateProfile,

}

return <AuthContext.Provider value={value}>{children}</AuthContext.Provider>

}

// =============================================================================

// FILE: components/auth/LoginForm.tsx

// =============================================================================

'use client'

import React, { useState } from 'react'

import { useAuth } from '@/contexts/AuthContext'

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

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

import { Mail, Lock, Loader2 } from 'lucide-react'

import toast from 'react-hot-toast'

interface LoginFormProps {

onSwitchToSignup: () => void

}

export const LoginForm: React.FC<LoginFormProps> = ({ onSwitchToSignup }) => {

const { signIn } = useAuth()

const [formData, setFormData] = useState({

email: '',

password: '',

})

const [isLoading, setIsLoading] = useState(false)

const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {

setFormData(prev => ({

...prev,

[e.target.name]: e.target.value,

}))

}

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

e.preventDefault()

setIsLoading(true)

const { error } = await signIn(formData.email, formData.password)

if (error) {

toast.error(error)

} else {

toast.success('Welcome back!')

}

setIsLoading(false)

}

return (

<div className="w-full max-w-md mx-auto">

<div className="bg-slate-800/50 backdrop-blur-sm border border-slate-700 rounded-lg p-6">

<h2 className="text-2xl font-bold text-white mb-6 text-center">

Welcome Back

</h2>

<form onSubmit={handleSubmit} className="space-y-4">

<div>

<label className="block text-sm font-medium text-slate-300 mb-2">

Email

</label>

<div className="relative">

<Mail className="absolute left-3 top-1/2 transform -translate-y-1/2 h-4 w-4 text-slate-400" />

<Input

type="email"

name="email"

value={formData.email}

onChange={handleChange}

placeholder="Enter your email"

className="pl-10 bg-slate-900/50 border-slate-600 text-white"

required

/>

</div>

</div>

<div>

<label className="block text-sm font-medium text-slate-300 mb-2">

Password

</label>

<div className="relative">

<Lock className="absolute left-3 top-1/2 transform -translate-y-1/2 h-4 w-4 text-slate-400" />

<Input

type="password"

name="password"

value={formData.password}

onChange={handleChange}

placeholder="Enter your password"

className="pl-10 bg-slate-900/50 border-slate-600 text-white"

required

/>

</div>

</div>

<Button

type="submit"

disabled={isLoading}

className="w-full bg-blue-600 hover:bg-blue-700 text-white"

>

{isLoading ? (

<>

<Loader2 className="mr-2 h-4 w-4 animate-spin" />

Signing In...

</>

) : (

'Sign In'

)}

</Button>

</form>

<div className="mt-6 text-center">

<p className="text-slate-400">

Don't have an account?{' '}

<button

onClick={onSwitchToSignup}

className="text-blue-400 hover:text-blue-300 font-medium"

>

Sign up

</button>

</p>

</div>

</div>

</div>

)

}

// =============================================================================

// FILE: components/auth/SignupForm.tsx

// =============================================================================

'use client'

import React, { useState } from 'react'

import { useAuth } from '@/contexts/AuthContext'

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

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

import { User, Mail, Lock, Loader2 } from 'lucide-react'

import toast from 'react-hot-toast'

interface SignupFormProps {

onSwitchToLogin: () => void

selectedPlan?: { id: string; name: string; price: string }

}

export const SignupForm: React.FC<SignupFormProps> = ({

onSwitchToLogin,

selectedPlan

}) => {

const { signUp } = useAuth()

const [formData, setFormData] = useState({

name: '',

email: '',

password: '',

confirmPassword: '',

})

const [isLoading, setIsLoading] = useState(false)

const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {

setFormData(prev => ({

...prev,

[e.target.name]: e.target.value,

}))

}

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

e.preventDefault()

if (formData.password !== formData.confirmPassword) {

toast.error("Passwords don't match")

return

}

if (formData.password.length < 8) {

toast.error('Password must be at least 8 characters')

return

}

setIsLoading(true)

const { error } = await signUp(formData.email, formData.password, formData.name)

if (error) {

toast.error(error)

} else {

toast.success('Account created! Please check your email to verify.')

}

setIsLoading(false)

}

return (

<div className="w-full max-w-md mx-auto">

<div className="bg-slate-800/50 backdrop-blur-sm border border-slate-700 rounded-lg p-6">

<h2 className="text-2xl font-bold text-white mb-6 text-center">

Create Your Account

</h2>

{selectedPlan && (

<div className="mb-4 p-3 bg-blue-600/20 border border-blue-500/30 rounded-lg">

<p className="text-blue-400 text-sm text-center">

Selected Plan: {selectedPlan.name} - {selectedPlan.price}

</p>

</div>

)}

<form onSubmit={handleSubmit} className="space-y-4">

<div>

<label className="block text-sm font-medium text-slate-300 mb-2">

Full Name

</label>

<div className="relative">

<User className="absolute left-3 top-1/2 transform -translate-y-1/2 h-4 w-4 text-slate-400" />

<Input

type="text"

name="name"

value={formData.name}

onChange={handleChange}

placeholder="Enter your full name"

className="pl-10 bg-slate-900/50 border-slate-600 text-white"

required

/>

</div>

</div>

<div>

<label className="block text-sm font-medium text-slate-300 mb-2">

Email

</label>

<div className="relative">

<Mail className="absolute left-3 top-1/2 transform -translate-y-1/2 h-4 w-4 text-slate-400" />

<Input

type="email"

name="email"

value={formData.email}

onChange={handleChange}

placeholder="Enter your email"

className="pl-10 bg-slate-900/50 border-slate-600 text-white"

required

/>

</div>

</div>

<div>

<label className="block text-sm font-medium text-slate-300 mb-2">

Password

</label>

<div className="relative">

<Lock className="absolute left-3 top-1/2 transform -translate-y-1/2 h-4 w-4 text-slate-400" />

<Input

type="password"

name="password"

value={formData.password}

onChange={handleChange}

placeholder="Create a password"

className="pl-10 bg-slate-900/50 border-slate-600 text-white"

required

/>

</div>

</div>

<div>

<label className="block text-sm font-medium text-slate-300 mb-2">

Confirm Password

</label>

<div className="relative">

<Lock className="absolute left-3 top-1/2 transform -translate-y-1/2 h-4 w-4 text-slate-400" />

<Input

type="password"

name="confirmPassword"

value={formData.confirmPassword}

onChange={handleChange}

placeholder="Confirm your password"

className="pl-10 bg-slate-900/50 border-slate-600 text-white"

required

/>

</div>

</div>

<Button

type="submit"

disabled={isLoading}

className="w-full bg-blue-600 hover:bg-blue-700 text-white"

>

{isLoading ? (

<>

<Loader2 className="mr-2 h-4 w-4 animate-spin" />

Creating Account...

</>

) : (

'Create Account'

)}

</Button>

</form>

<div className="mt-6 text-center">

<p className="text-slate-400">

Already have an account?{' '}

<button

onClick={onSwitchToLogin}

className="text-blue-400 hover:text-blue-300 font-medium"

>

Sign in

</button>

</p>

</div>

</div>

</div>

)

}

// =============================================================================

// FILE: middleware.ts (Next.js App Router)

// =============================================================================

import { createMiddlewareClient } from '@supabase/auth-helpers-nextjs'

import { NextResponse } from 'next/server'

import type { NextRequest } from 'next/server'

export async function middleware(req: NextRequest) {

const res = NextResponse.next()

const supabase = createMiddlewareClient({ req, res })

// Refresh session if expired

const { data: { session } } = await supabase.auth.getSession()

// Protect dashboard routes

if (req.nextUrl.pathname.startsWith('/dashboard')) {

if (!session) {

return NextResponse.redirect(new URL('/', req.url))

}

}

// Redirect authenticated users away from auth pages

if (req.nextUrl.pathname === '/' && session) {

return NextResponse.redirect(new URL('/dashboard', req.url))

}

return res

}

export const config = {

matcher: ['/((?!api|\_next/static|\_next/image|favicon.ico).\*)'],

}