File: client/src/pages/Subscription.tsx

Type: tsx | Size: 11107 bytes

```
import React, { useState, useEffect } from 'react';
import { motion } from 'framer-motion';
import { loadStripe } from '@stripe/stripe-js';
import { Elements, PaymentElement, useStripe, useElements } from '@stripe/react-stripe-js';
import { useToast } from '@/hooks/use-toast';
import { SubscriptionPlan, premiumFeatures } from '@shared/schema';
import HeartLoader from '@/components/HeartLoader';
import { apiRequest } from '@/lib/queryClient';

// Make sure to call loadStripe outside of a component's render to avoid recreating the Stripe object
const stripePromise = loadStripe(import.meta.env.VITE_STRIPE_PUBLIC_KEY);

interface CheckoutFormProps {
  clientSecret: string;
  onSuccess: () => void;
  onError: (message: string) => void;
}

const CheckoutForm = ({ clientSecret, onSuccess, onError }: CheckoutFormProps) => {
  const stripe = useStripe();
  const elements = useElements();
  const [isProcessing, setIsProcessing] = useState(false);
  const { toast } = useToast();

  const handleSubmit = async (e: React.FormEvent<HTMLFormElement>) => {
    e.preventDefault();

    if (!stripe || !elements) {
      return;
    }

    setIsProcessing(true);

    const { error } = await stripe.confirmPayment({
      elements,
      confirmParams: {
        return_url: window.location.origin + '/subscription-success',
      },
      redirect: 'if_required',
    });

    if (error) {
      onError(error.message || 'An unexpected error occurred');
      setIsProcessing(false);
    } else {
      // Payment successful
      onSuccess();
      setIsProcessing(false);
    }
  };

  return (
    <form onSubmit={handleSubmit} className="space-y-6">
      <PaymentElement />

      <div className="text-xs text-mystical-lavender/70 italic">
        <p>For testing, use card number: 4242 4242 4242 4242</p>
        <p>Any future expiry date and any 3-digit CVC</p>
      </div>

      <motion.button
        type="submit"
        disabled={!stripe || isProcessing}
        className={`w-full py-3 rounded-lg font-medium ${
          isProcessing ? 'bg-mystical-purple/50' : 'bg-mystical-purple'
        } text-white transition-all`}
        whileHover={{ scale: isProcessing ? 1 : 1.02 }}
        whileTap={{ scale: isProcessing ? 1 : 0.98 }}
      >
        {isProcessing ? (
          <div className="flex items-center justify-center">
            <div className="w-5 h-5 border-2 border-white border-t-transparent rounded-full animate-spin mr-2"...
            Processing...
          </div>
        ) : (
          'Subscribe Now'
        )}
      </motion.button>
    </form>
  );
};

export default function Subscription() {
  const [plans, setPlans] = useState<SubscriptionPlan[]>([]);
  const [selectedPlan, setSelectedPlan] = useState<string>('');
  const [clientSecret, setClientSecret] = useState<string>('');
  const [isLoading, setIsLoading] = useState(true);
  const [isPremium, setIsPremium] = useState(false);
  const [email, setEmail] = useState('');
  const { toast } = useToast();

  // Fetch subscription plans
  useEffect(() => {
    const fetchPlans = async () => {
      try {
        const response = await fetch('/api/subscription-plans');
        const data = await response.json();
        setPlans(data);
        if (data.length > 0) {
          setSelectedPlan(data[0].id);
        }
        setIsLoading(false);
      } catch (error) {
        console.error('Error fetching plans:', error);
        toast({
          title: 'Error',
          description: 'Failed to load subscription plans',
          variant: 'destructive',
        });
        setIsLoading(false);
      }
    };

    fetchPlans();
  }, [toast]);

  // Check if user is already premium
  useEffect(() => {
    const checkPremiumStatus = async () => {
      try {
        // This is just a mock check since we don't have real authentication
        // In a real app, you would check the logged-in user's status
        setIsPremium(false);
      } catch (error) {
        console.error('Error checking premium status:', error);
      }
    };

    checkPremiumStatus();
  }, []);

  const createSubscription = async () => {
    if (!selectedPlan || !email.trim()) {
      toast({
        title: 'Missing Information',
        description: 'Please provide your email address',
        variant: 'destructive',
      });
      return;
    }

    setIsLoading(true);
    try {
      // In a real app, you would use the actual user ID
      const userId = 1; // Mock user ID for development

      const response = await apiRequest('POST', '/api/get-or-create-subscription', {
        userId,
        planId: selectedPlan,
        email,
      });

      const data = await response.json();

      if (!data.clientSecret) {
        throw new Error('No client secret returned');
      }

      setClientSecret(data.clientSecret);
    } catch (error) {
      console.error('Error creating subscription:', error);
      toast({
        title: 'Error',
        description: 'Failed to create subscription',
        variant: 'destructive',
      });
    } finally {
      setIsLoading(false);
    }
  };

  const handleSuccess = () => {
    setIsPremium(true);
    toast({
      title: 'Success!',
      description: 'Your premium subscription is now active',
      variant: 'default',
    });
  };

  const handleError = (message: string) => {
    toast({
      title: 'Payment Failed',
      description: message,
      variant: 'destructive',
    });
  };

  if (isLoading) {
    return (
      <div className="flex justify-center items-center min-h-[70vh]">
        <HeartLoader isVisible={true} message="Loading subscription options..." />
      </div>
    );
  }

  return (
    <section className="px-4 py-6">
      <div className="text-center mb-8">
        <h2 className="font-serif text-3xl text-mystical-pink">Premium Membership</h2>
        <p className="text-mystical-lavender mt-2">Unlock the full power of the Love Oracle</p>
      </div>

      {isPremium ? (
        <div className="bg-mystical-gradient backdrop-blur-md rounded-xl p-5 text-center">
          <div className="w-20 h-20 mx-auto bg-mystical-gold/20 rounded-full flex items-center justify-center ...
            <i className="fas fa-crown text-mystical-gold text-3xl"></i>
          </div>
          <h3 className="text-2xl font-serif text-mystical-light mb-2">You're a Premium Member!</h3>
          <p className="text-mystical-lavender mb-4">
            Thank you for supporting Love Oracle. Enjoy all your premium benefits!
          </p>
          <ul className="space-y-2 text-left mb-6">
            {premiumFeatures.map((feature, index) => (
              <li key={index} className="flex items-start">
          <span className="text-mystical-gold mr-2">✓</span>
          <span className="text-mystical-light">{feature}</span>
        </li>
      ))}
    </ul>
  </div>
) : clientSecret ? (
  <div className="bg-mystical-gradient backdrop-blur-md rounded-xl p-5">
    <h3 className="text-xl font-serif text-mystical-light mb-4">Complete Your Subscription</h3>
    <Elements stripe={stripePromise} options={{ clientSecret }}>
      <CheckoutForm
        clientSecret={clientSecret}
        onSuccess={handleSuccess}
        onError={handleError}
      />
    </Elements>
  </div>
) : (
  <>
    {/* Plan Selection */}
    <div className="space-y-4 mb-8">
      {plans.map((plan) => (
        <motion.div
          key={plan.id}
          className={`border rounded-xl p-5 cursor-pointer transition-all ${
            selectedPlan === plan.id
              ? 'border-mystical-pink bg-mystical-gradient'
              : 'border-mystical-purple/30 bg-mystical-dark/50'
          }`}
          whileHover={{ scale: 1.02 }}
          whileTap={{ scale: 0.98 }}
          onClick={() => setSelectedPlan(plan.id)}
        >
          <div className="flex justify-between items-center mb-3">
            <h3 className="font-serif text-xl text-mystical-light">{plan.name}</h3>
            <div className="text-right">
              <span className="text-mystical-gold text-2xl font-semibold">${plan.price}</span>
              <span className="text-mystical-lavender text-sm">/{plan.interval}</span>
            </div>
          </div>
          <p className="text-mystical-lavender mb-3">{plan.description}</p>
          <ul className="space-y-1">
            {plan.features.map((feature, idx) => (
              <li key={idx} className="flex items-start text-sm">
                <span className="text-mystical-gold mr-2">✓</span>
                <span className="text-mystical-light">{feature}</span>
              </li>
            ))}
          </ul>
        </motion.div>
      ))}
    </div>

    {/* Email Input */}
    <div className="bg-mystical-gradient backdrop-blur-md rounded-xl p-5 mb-6">
      <h3 className="text-xl font-serif text-mystical-light mb-4">Your Information</h3>
      <div className="space-y-4">
        <div>
          <label htmlFor="email" className="block text-mystical-light mb-1">
            Email Address
          </label>
          <input
            type="email"
            id="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            className="w-full bg-white/10 border border-mystical-lavender/30 text-mystical-lavender roun...
            placeholder="your@email.com"
            required
          />
        </div>

        <motion.button
```

```
              onClick={createSubscription}
              className="w-full bg-mystical-purple text-white font-medium py-3 rounded-lg"
              whileHover={{ scale: 1.02 }}
              whileTap={{ scale: 0.98 }}
            >
              Continue to Payment
            </motion.button>
          </div>
        </div>

        {/* Benefits */}
        <div className="bg-mystical-dark border border-mystical-purple/30 rounded-xl p-5">
          <h3 className="text-xl font-serif text-mystical-pink mb-4">Premium Benefits</h3>
          <ul className="space-y-3">
            {premiumFeatures.map((feature, index) => (
              <li key={index} className="flex items-start">
                <div className="w-5 h-5 rounded-full bg-mystical-purple/20 flex items-center justify-center ...
                  <i className="fas fa-sparkle text-mystical-pink text-xs"></i>
                </div>
                <span className="text-mystical-lavender">{feature}</span>
              </li>
            ))}
          </ul>
        </div>
      </>
    )}
  </section>
 );
}
```