**USER LOGIN**

import React, { useState } from 'react';

import { Link, useNavigate } from 'react-router-dom';

import { Eye, EyeOff, Users, Brain, ArrowLeft } from 'lucide-react';

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

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

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

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

import { Tabs, TabsContent, TabsList, TabsTrigger } from '@/components/ui/tabs';

import { ThemeToggle } from '@/components/ThemeToggle';

const UserLogin = () => {

  const [showPassword, setShowPassword] = useState(false);

  const [loginData, setLoginData] = useState({ email: '', password: '' });

  const [signupData, setSignupData] = useState({

    name: '',

    email: '',

    password: '',

    confirmPassword: ''

  });

  const navigate = useNavigate();

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

    e.preventDefault();

    try {

      const res = await fetch('/api/auth/login', {

        method: 'POST',

        headers: { 'Content-Type': 'application/json' },

        body: JSON.stringify({

          email: loginData.email,

          password: loginData.password

        })

      });

      const data = await res.json();

      if (res.ok) {

        // Save user info, redirect, etc.

        // e.g., localStorage.setItem('user', JSON.stringify(data));

        navigate('/user/dashboard');

      } else {

        alert(data.detail || 'Login failed');

      }

    } catch (err) {

      alert('Server error');

    }

  };

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

    e.preventDefault();

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

      alert('Passwords do not match!');

      return;

    }

    try {

      const res = await fetch('/api/auth/signup', {

        method: 'POST',

        headers: { 'Content-Type': 'application/json' },

        body: JSON.stringify({

          full\_name: signupData.name,

          email: signupData.email,

          password: signupData.password

        })

      });

      const data = await res.json();

      if (res.ok) {

        // Save user info, redirect, etc.

        // e.g., localStorage.setItem('user', JSON.stringify(data));

        navigate('/user/dashboard');

      } else {

        alert(data.detail || 'Sign up failed');

      }

    } catch (err) {

      alert('Server error');

    }

  };

  return (

    <div className="min-h-screen bg-gradient-to-br from-blue-50 via-purple-50 to-pink-50 dark:from-gray-900 dark:via-blue-900 dark:to-purple-900 flex items-center justify-center p-4">

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

        {/\* Back to Home and Theme Toggle \*/}

        <div className="flex items-center justify-between mb-6">

          <Link to="/" className="inline-flex items-center text-purple-600 dark:text-purple-400 hover:text-purple-700 dark:hover:text-purple-300 transition-colors">

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

            Back to Home

          </Link>

          <ThemeToggle />

        </div>

        {/\* Header \*/}

        <div className="text-center mb-8">

          <div className="flex items-center justify-center space-x-3 mb-4">

            <div className="w-12 h-12 bg-gradient-to-r from-purple-400 to-pink-400 rounded-lg flex items-center justify-center">

              <Brain className="w-7 h-7 text-white" />

            </div>

            <h1 className="text-2xl font-bold bg-gradient-to-r from-purple-600 to-pink-600 bg-clip-text text-transparent">

              ApexWolf

            </h1>

          </div>

          <h2 className="text-3xl font-bold text-gray-800 dark:text-gray-100 mb-2">User Portal</h2>

          <p className="text-gray-600 dark:text-gray-300">Access your personalized financial dashboard</p>

        </div>

        {/\* Login/Signup Card \*/}

        <Card className="bg-white/90 dark:bg-gray-800/90 border-purple-200 dark:border-gray-700 backdrop-blur-sm shadow-xl">

          <CardHeader className="text-center pb-4">

            <div className="w-16 h-16 bg-gradient-to-r from-purple-500 to-purple-600 rounded-full flex items-center justify-center mx-auto mb-4">

              <Users className="w-8 h-8 text-white" />

            </div>

            <CardTitle className="text-gray-800 dark:text-gray-100 text-xl">Welcome Back</CardTitle>

          </CardHeader>

          <CardContent>

            <Tabs defaultValue="login" className="w-full">

              <TabsList className="grid w-full grid-cols-2 bg-purple-100 dark:bg-gray-700">

                <TabsTrigger value="login" className="text-gray-700 dark:text-gray-300 data-[state=active]:bg-purple-500 data-[state=active]:text-white">

                  Sign In

                </TabsTrigger>

                <TabsTrigger value="signup" className="text-gray-700 dark:text-gray-300 data-[state=active]:bg-purple-500 data-[state=active]:text-white">

                  Sign Up

                </TabsTrigger>

              </TabsList>

              <TabsContent value="login">

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

                  <div className="space-y-2">

                    <Label htmlFor="email" className="text-gray-700 dark:text-gray-300">Email</Label>

                    <Input

                      id="email"

                      type="email"

                      placeholder="Enter your email"

                      value={loginData.email}

                      onChange={(e) => setLoginData({ ...loginData, email: e.target.value })}

                      className="bg-gray-50 dark:bg-gray-700 border-gray-300 dark:border-gray-600 text-gray-800 dark:text-gray-100 placeholder:text-gray-500 dark:placeholder:text-gray-400 focus:border-purple-400 dark:focus:border-purple-500"

                      required

                    />

                  </div>

                  <div className="space-y-2">

                    <Label htmlFor="password" className="text-gray-700 dark:text-gray-300">Password</Label>

                    <div className="relative">

                      <Input

                        id="password"

                        type={showPassword ? "text" : "password"}

                        placeholder="Enter your password"

                        value={loginData.password}

                        onChange={(e) => setLoginData({ ...loginData, password: e.target.value })}

                        className="bg-gray-50 dark:bg-gray-700 border-gray-300 dark:border-gray-600 text-gray-800 dark:text-gray-100 placeholder:text-gray-500 dark:placeholder:text-gray-400 focus:border-purple-400 dark:focus:border-purple-500 pr-10"

                        required

                      />

                      <button

                        type="button"

                        onClick={() => setShowPassword(!showPassword)}

                        className="absolute right-3 top-1/2 transform -translate-y-1/2 text-gray-500 dark:text-gray-400 hover:text-gray-700 dark:hover:text-gray-300"

                      >

                        {showPassword ? <EyeOff className="w-4 h-4" /> : <Eye className="w-4 h-4" />}

                      </button>

                    </div>

                  </div>

                  <Button

                    type="submit"

                    className="w-full bg-gradient-to-r from-purple-500 to-purple-600 hover:from-purple-600 hover:to-purple-700 text-white font-semibold py-3 transition-all duration-300 transform hover:scale-105"

                  >

                    Sign In to Dashboard

                  </Button>

                </form>

              </TabsContent>

              <TabsContent value="signup">

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

                  <div className="space-y-2">

                    <Label htmlFor="name" className="text-gray-700 dark:text-gray-300">Full Name</Label>

                    <Input

                      id="name"

                      type="text"

                      placeholder="Enter your full name"

                      value={signupData.name}

                      onChange={(e) => setSignupData({ ...signupData, name: e.target.value })}

                      className="bg-gray-50 dark:bg-gray-700 border-gray-300 dark:border-gray-600 text-gray-800 dark:text-gray-100 placeholder:text-gray-500 dark:placeholder:text-gray-400 focus:border-purple-400 dark:focus:border-purple-500"

                      required

                    />

                  </div>

                  <div className="space-y-2">

                    <Label htmlFor="signup-email" className="text-gray-700 dark:text-gray-300">Email</Label>

                    <Input

                      id="signup-email"

                      type="email"

                      placeholder="Enter your email"

                      value={signupData.email}

                      onChange={(e) => setSignupData({ ...signupData, email: e.target.value })}

                      className="bg-gray-50 dark:bg-gray-700 border-gray-300 dark:border-gray-600 text-gray-800 dark:text-gray-100 placeholder:text-gray-500 dark:placeholder:text-gray-400 focus:border-purple-400 dark:focus:border-purple-500"

                      required

                    />

                  </div>

                  <div className="space-y-2">

                    <Label htmlFor="signup-password" className="text-gray-700 dark:text-gray-300">Password</Label>

                    <div className="relative">

                      <Input

                        id="signup-password"

                        type={showPassword ? "text" : "password"}

                        placeholder="Create a password"

                        value={signupData.password}

                        onChange={(e) => setSignupData({ ...signupData, password: e.target.value })}

                        className="bg-gray-50 dark:bg-gray-700 border-gray-300 dark:border-gray-600 text-gray-800 dark:text-gray-100 placeholder:text-gray-500 dark:placeholder:text-gray-400 focus:border-purple-400 dark:focus:border-purple-500 pr-10"

                        required

                      />

                      <button

                        type="button"

                        onClick={() => setShowPassword(!showPassword)}

                        className="absolute right-3 top-1/2 transform -translate-y-1/2 text-gray-500 dark:text-gray-400 hover:text-gray-700 dark:hover:text-gray-300"

                      >

                        {showPassword ? <EyeOff className="w-4 h-4" /> : <Eye className="w-4 h-4" />}

                      </button>

                    </div>

                  </div>

                  <div className="space-y-2">

                    <Label htmlFor="confirm-password" className="text-gray-700 dark:text-gray-300">Confirm Password</Label>

                    <Input

                      id="confirm-password"

                      type="password"

                      placeholder="Confirm your password"

                      value={signupData.confirmPassword}

                      onChange={(e) => setSignupData({ ...signupData, confirmPassword: e.target.value })}

                      className="bg-gray-50 dark:bg-gray-700 border-gray-300 dark:border-gray-600 text-gray-800 dark:text-gray-100 placeholder:text-gray-500 dark:placeholder:text-gray-400 focus:border-purple-400 dark:focus:border-purple-500"

                      required

                    />

                  </div>

                  <Button

                    type="submit"

                    className="w-full bg-gradient-to-r from-purple-500 to-purple-600 hover:from-purple-600 hover:to-purple-700 text-white font-semibold py-3 transition-all duration-300 transform hover:scale-105"

                  >

                    Create Account

                  </Button>

                </form>

              </TabsContent>

            </Tabs>

          </CardContent>

        </Card>

      </div>

    </div>

  );

};

export default UserLogin;

**ELIGIBILITY PREDICTION**

import React, { useState } from 'react';

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

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

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

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

import { ScrollArea } from '@/components/ui/scroll-area';

import { Alert, AlertDescription } from '@/components/ui/alert';

import { Loader2, CheckCircle, AlertCircle, User, Briefcase, Calculator, TrendingUp, Shield } from 'lucide-react';

// All dropdown/select options for every possible field (all others empty for free entry)

const fieldOptions: Record<string, string[]> = {

  "Education Level": ["PhD", "Masters", "Graduate", "Undergraduate", "None"],

  "Employment Type": ["Full-time", "Part-time", "Contracted", "Internship/Seasonal/Casual"],

  "Insurance": ["Comprehensive", "Partial", "Basic", "None"],

  "Emergency Fund": ["Available", "Partial", "None"],

  "Digital Access": ["Full Access", "Limited", "None"],

  "Health Insurance": ["Comprehensive", "Basic", "None"],

  "Device Access": ["Multiple Devices", "Single Device", "No Device"],

  "ID Proof": ["Govt Verified", "Partial", "None"],

  "Institution": ["Top Tier", "Recognized", "Average", "Unknown"],

  "Institution Ranking": ["Top 100", "Top 500", "Other"],

  "Internet Access": ["Stable Broadband", "Mobile Only", "None"],

  "Job Role": ["Senior", "Mid", "Junior", "Intern"],

  "Leasing Company": ["Reputed", "Registered", "Unknown"],

  "Living Costs": ["Low", "Medium", "High"],

  "Loan Co-signer": ["Strong", "Moderate", "Weak", "None"],

  "Loan History": ["Good", "Mixed", "Bad"],

  "Loan Purpose": ["Education", "Business", "Medical", "Consumption", "Other"],

  "Mentor Assigned": ["Experienced", "Average", "None"],

  "Monthly Sales Target": ["Met", "Near Target", "Missed"],

  "Org Type": ["International NGO", "Local NGO", "Informal"],

  "Parent Support": ["Full", "Partial", "None"],

  "Past Employment": ["Govt/Stable", "Corporate", "Contractual", "None"],

  "Platform Used": ["Verified", "Common", "Niche"],

  "Product Sold": ["High Demand", "Niche", "Seasonal"],

  "Project Type": ["Corporate", "SME", "Individual"],

  "Property Owned": ["Multiple", "Single", "None"],

  "References": ["Strong", "Moderate", "None"],

  "Registration Status": ["Registered", "Pending", "Unregistered"],

  "Risk Score": ["Low", "Moderate", "High"],

  "Scholarship": ["Full", "Partial", "None"],

  "Sector": ["Core", "Growth", "Other"],

  "Sector Type": ["Tech", "Finance", "Healthcare", "Edu", "Other"],

  "Skill Category": ["High", "Mid", "Low"],

  "Skills": ["Advanced", "Intermediate", "Basic"],

  "Sponsor (Y/N)": ["Yes", "No"],

  "Spouse Job Type": ["Stable", "Moderate", "Unstable"],

  "Tax Compliance": ["Full", "Partial", "None"],

  "Tax Record": ["Clean", "Minor Issues", "Problematic"],

  "Tax Status": ["Filed", "Irregular", "Not Filed"],

  "Tools Owned": ["Complete Set", "Partial", "Minimal"],

  "Training Plan": ["In Progress", "Planned", "None"],

  "Training Type": ["Technical", "Soft Skills", "General"],

  "Transport Mode": ["Own Vehicle", "Public", "None"],

  "Volunteer Hours": ["Frequent", "Occasional", "Rare"],

  "Volatility Buffer": ["Yes", "No"],

  "Work Category": ["Consulting", "Creative", "Other"],

  "Client Industry": ["Tier 1", "Tier 2", "Unstable"],

  "Digital Adoption": ["High", "Moderate", "Low"],

  "Employer Sector": ["Govt/PSU", "Private", "Gig"],

  "Employer Type": ["MNC", "SME", "Unregistered"],

  "Field": ["STEM", "Business", "Arts", "Skilled", "Technical", "Other"],

  "Benefits Access": ["Comprehensive", "Partial", "Minimal", "None"],

  "Career Gap Reason": ["Health/Family", "Education", "Relocation", "Unknown"],

  "Certificate Status": ["Certified", "Pending", "Not Pursued"],

  "GST Score": ["Excellent", "Good", "Average", "Poor", "Unknown"],

  "Assistance Type": ["Housing & Income", "Basic Income Support", "Food Only"],

  "Affiliation": ["Top-tier University", "Recognized Institution", "Local/Unknown"],

  "Sponsor": ["Yes", "No"]

};

// User types as before

const employedInputs: Record<string, string[]> = {

  "Full-Time Employment": [

    "Age", "Monthly Salary", "Years of Experience", "Job Tenure", "Credit Score",

    "Tax Status", "PF Balance", "Dependents", "Sector Type",

    "Education Level", "Insurance", "Job Stability Score"

  ],

  "Part-Time Employment": [

    "Age", "Monthly Income", "Weekly Hours", "Job Duration", "Secondary Income",

    "Credit Score", "Tax Filing Status", "Employer Type", "Skills",

    "Health Insurance", "Savings Level"

  ],

  "Apprenticeship": [

    "Age", "Stipend", "Program Duration", "Institution", "Grade", "Sponsored (Y/N)",

    "Living Costs", "Credit Score", "Parent Income", "Loan Purpose", "Digital Access",

    "Emergency Fund"

  ],

  "Traineeship": [

    "Age", "Program Length", "Affiliation", "Training Type", "Mentor Assigned",

    "Living Costs", "Transport Mode", "GPA", "Parent Support",

    "Certificate Status"

  ],

  "Internship": [

    "Age", "Paid/Unpaid", "Duration", "Stipend", "Parent Income", "GPA",

    "Credit Score", "Savings", "Loan Purpose", "Institution Ranking", "Career Interest",

    "Device Access"

  ],

  "Contract Employment": [

    "Age", "Income", "Contract Duration", "Employer Sector", "Prior Contracts",

    "Gaps", "Fixed Expenses", "Credit Score", "Emergency Fund",

    "Tax Records", "Project Count"

  ],

  "Seasonal Employment": [

    "Age", "Last Season Income", "Season Type", "Months Worked", "Off-Season Income",

    "Loan Purpose", "Credit Score", "Govt Scheme Eligibility", "Savings",

    "Family Dependents"

  ],

  "Casual Employment": [

    "Age", "Daily Wage", "Work Days/Month", "Role Type", "Income Consistency",

    "Credit Score", "Living Costs", "Health Status", "ID Proof",

    "Employer Type"

  ],

  "Employment on Commission": [

    "Age", "Monthly Earnings", "Product Sold", "Base Pay", "Sales Success Rate",

    "Credit Score", "Commission Volatility", "CRM Use", "Expenses",

    "Monthly Sales Target"

  ],

  "Probationary Employment": [

    "Age", "Salary", "Probation Period", "Employer Type", "Job Role",

    "Credit Score", "Insurance", "Expenses", "Future Job Confirmation",

    "Fixed Deductions"

  ],

  "Leased Employment": [

    "Age", "Salary", "Leasing Company", "Client Industry", "Contract Period",

    "Credit Score", "Hours Billed", "Expense Claims", "Job Changes",

    "Benefits Access"

  ],

  "Contingent Employment": [

    "Age", "Role (Freelancer/Consultant)", "Income", "Project Frequency",

    "Credit Score", "Platform Used", "Tools Owned", "Experience", "Work Category",

    "Tax Compliance"

  ]

};

const notEmployedInputs: Record<string, string[]> = {

  "Self-Employed": [

    "Age", "Business Type", "Monthly Revenue", "Tax Filing", "Credit Score",

    "Employees Count", "Operation Years", "Asset Owned",

    "Registration Status", "Sector"

  ],

  "Freelancer / Gig Worker": [

    "Age", "Skill Category", "Monthly Earnings", "Platform Used", "Clients Count",

    "Credit Score", "Ratings", "Internet Access", "Expenses", "Emergency Funds",

    "Tax Compliance"

  ],

  "Consultant / Independent Contractor": [

    "Age", "Field", "Hourly Rate", "Billing Hours", "Client Count",

    "Project Type", "Certifications", "Credit Score", "Contract Duration",

    "References"

  ],

  "Business Owner / Entrepreneur": [

    "Age", "Business Category", "Monthly Revenue", "Net Profit Margin", "Credit Score",

    "Employees", "Tax Record", "GST Score", "Assets",

    "Digital Adoption"

  ],

  "Retired (Pensioner)": [

    "Age", "Pension Income", "Savings", "Credit Score",

    "Health Costs", "Insurance", "Dependents", "Property Ownership", "Spouse Income",

    "Past Employment"

  ],

  "Homemaker": [

    "Age", "Spouse Job Type", "Household Income", "Loan Co-signer", "Credit Score",

    "Family Expenses", "Dependents", "Property Ownership", "Insurance", "ID Proof"

  ],

  "Unemployed – Actively Looking": [

    "Age", "Last Job Duration", "Gap Length", "Applications/Week", "Credit Score",

    "Emergency Fund", "Expected Salary", "Skills Training (Y/N)"

  ],

  "Unemployed – Not Looking": [

    "Age", "Career Gap Reason", "Past Job", "Credit Score", "Supporter Income",

    "Expenses", "Training Plan", "Loan Co-signer", "Emergency Fund", "Property Ownership"

  ],

  "Student": [

    "Age", "Education Level", "GPA", "Institution", "Scholarship",

    "Family Income", "Part-Time Job", "Living Expenses", "Device Access", "Co-signer",

    "Credit Score"

  ],

  "Receiving Government Assistance": [

    "Age", "Assistance Type", "Monthly Income", "Credit Score",

    "Housing", "Dependents", "Tax Status", "Savings", "Government ID",

    "Loan History"

  ],

  "Volunteer Work (Unpaid)": [

    "Age", "Volunteer Hours", "Org Type", "Duration", "Skills",

    "Living Costs", "Credit Score", "Sponsor (Y/N)", "Family Support", "Loan Purpose",

    "Expenses"

  ],

  "Investor": [

    "Age", "Income Type", "Monthly Income", "Portfolio Value", "Credit Score",

    "Property Owned", "Asset Types", "Risk Score", "Tax Filing", "Loan Purpose",

    "Volatility Buffer"

  ]

};

const userTypesMap = {

  employed: Object.keys(employedInputs),

  notemployed: Object.keys(notEmployedInputs)

};

const LoanPrediction: React.FC<{ userId: number }> = ({ userId }) => {

  const [employmentStatus, setEmploymentStatus] = useState<string>('');

  const [userType, setUserType] = useState<string>('');

  const [formData, setFormData] = useState<Record<string, string>>({});

  const [features, setFeatures] = useState<string[]>([]);

  const [result, setResult] = useState<any>(null);

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

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

  const [touched, setTouched] = useState<Record<string, boolean>>({});

  const [showExplain, setShowExplain] = useState<boolean>(false);

  // Handle employment selection

  const handleEmploymentChange = (status: string) => {

    setEmploymentStatus(status);

    setUserType('');

    setFormData({});

    setFeatures([]);

    setResult(null);

    setError('');

    setTouched({});

    setShowExplain(false);

  };

  // Handle user type (profile type) selection

  const handleUserTypeSelect = (type: string) => {

    setUserType(type);

    const featureList = userTypesMap.employed.includes(type)

      ? employedInputs[type]

      : notEmployedInputs[type];

    setFeatures(featureList || []);

    setFormData({});

    setResult(null);

    setError('');

    setTouched({});

    setShowExplain(false);

  };

  // Handle individual field input

  const handleInputChange = (key: string, value: string) => {

    setFormData(prev => ({ ...prev, [key]: value }));

    setTouched(prev => ({ ...prev, [key]: true }));

  };

  // Form validation (all fields must be filled, numbers must be numbers)

  const validateForm = () => {

    let errors: Record<string, string> = {};

    features.forEach(field => {

      if (!formData[field] || formData[field].trim() === "") {

        errors[field] = "Required";

      }

      if (

        field.toLowerCase().match(/age|year|gpa|score|amount|income|salary|count|duration|balance|expenses|tenure|months|margin/) &&

        formData[field] &&

        isNaN(Number(formData[field]))

      ) {

        errors[field] = "Must be a number";

      }

    });

    return errors;

  };

  // On submit: validate and fetch prediction

  const handleSubmit = async () => {

    setTouched(Object.fromEntries(features.map(f => [f, true])));

    const errors = validateForm();

    if (Object.keys(errors).length > 0) {

      setError('Please fill all fields correctly.');

      setResult(null);

      return;

    }

    setLoading(true);

    setError('');

    try {

      const stringInputs = Object.fromEntries(

        Object.entries(formData).map(([k, v]) => [k, String(v)])

      );

      const response = await fetch('http://localhost:5000/api/predict', {

        method: 'POST',

        headers: { 'Content-Type': 'application/json', "x-user-id": String(userId) },

        body: JSON.stringify({

          user\_type: userType,

          inputs: stringInputs

        })

      });

      if (!response.ok) {

        throw new Error(`HTTP error! status: ${response.status}`);

      }

      const data = await response.json();

      setResult(data);

    } catch (err) {

      setError('Error fetching prediction. Please check your inputs and try again.');

      setResult(null);

    } finally {

      setLoading(false);

      setShowExplain(false);

    }

  };

  // For gradient card blocks

  const getGradientBlock = (color: string, children: React.ReactNode) => (

    <div className={`p-6 rounded-xl border shadow-md bg-gradient-to-br from-slate-50 to-gray-100 border-gray-200 hover:shadow-lg transition-shadow duration-300`}>

      {children}

    </div>

  );

  // Format INR currency

  const formatCurrency = (val: any) =>

    val !== undefined && !isNaN(val) ? `₹${Number(val).toLocaleString()}` : '—';

  return (

    <div className="min-h-screen bg-white">

      <div className="w-full max-w-none mx-0">

        {/\* Header Section \*/}

        <div className="bg-gradient-to-r from-blue-50 via-indigo-50 to-purple-50 border-b border-gray-200">

          <div className="max-w-7xl mx-auto px-6 py-8">

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

              <div className="bg-gradient-to-br from-blue-500 to-indigo-600 p-3 rounded-xl shadow-lg">

                <Calculator className="h-8 w-8 text-white" />

              </div>

              <div>

                <h1 className="text-3xl font-bold text-gray-800 mb-1 flex items-center gap-3">

                  Eligibility Prediction

                  <div className="flex gap-1">

                    <TrendingUp className="h-6 w-6 text-green-600" />

                    <Shield className="h-6 w-6 text-blue-600" />

                  </div>

                </h1>

                <p className="text-gray-600 text-lg">Get instant, accurate loan eligibility assessment based on your comprehensive profile analysis</p>

              </div>

            </div>

            {/\* Feature highlights \*/}

            <div className="flex flex-wrap gap-6 mt-6">

              <div className="flex items-center gap-2 bg-white/60 px-4 py-2 rounded-full shadow-sm">

                <div className="w-2 h-2 bg-green-500 rounded-full"></div>

                <span className="text-sm font-medium text-gray-700">Instant Results</span>

              </div>

              <div className="flex items-center gap-2 bg-white/60 px-4 py-2 rounded-full shadow-sm">

                <div className="w-2 h-2 bg-blue-500 rounded-full"></div>

                <span className="text-sm font-medium text-gray-700">AI-Powered Analysis</span>

              </div>

              <div className="flex items-center gap-2 bg-white/60 px-4 py-2 rounded-full shadow-sm">

                <div className="w-2 h-2 bg-purple-500 rounded-full"></div>

                <span className="text-sm font-medium text-gray-700">Detailed Insights</span>

              </div>

              <div className="flex items-center gap-2 bg-white/60 px-4 py-2 rounded-full shadow-sm">

                <div className="w-2 h-2 bg-orange-500 rounded-full"></div>

                <span className="text-sm font-medium text-gray-700">Secure & Private</span>

              </div>

            </div>

          </div>

        </div>

        {/\* Main Content \*/}

        <div className="max-w-7xl mx-auto px-6 py-8">

          {/\* Employment Status Selection \*/}

          <Card className="mb-8 shadow-lg border-0 bg-white">

            <CardHeader className="bg-gradient-to-r from-slate-50 to-gray-100">

              <CardTitle className="text-xl flex items-center gap-2 text-gray-800">

                <Briefcase className="h-5 w-5 text-blue-600" />

                Employment Status

              </CardTitle>

            </CardHeader>

            <CardContent className="p-6">

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

                <Button

                  variant="outline"

                  onClick={() => handleEmploymentChange('employed')}

                  className={`h-16 text-lg font-medium transition-all duration-300

                    ${employmentStatus === 'employed'

                      ? 'bg-gradient-to-r from-blue-500 to-indigo-600 text-white border-0 shadow-lg hover:from-blue-600 hover:to-indigo-700'

                      : 'bg-white text-gray-700 border-gray-300 hover:bg-gray-50 hover:border-gray-400'

                    }`}

                >

                  <User className="mr-2 h-5 w-5" />

                  Employed

                </Button>

                <Button

                  variant="outline"

                  onClick={() => handleEmploymentChange('notemployed')}

                  className={`h-16 text-lg font-medium transition-all duration-300

                    ${employmentStatus === 'notemployed'

                      ? 'bg-gradient-to-r from-blue-500 to-indigo-600 text-white border-0 shadow-lg hover:from-blue-600 hover:to-indigo-700'

                      : 'bg-white text-gray-700 border-gray-300 hover:bg-gray-50 hover:border-gray-400'

                    }`}

                >

                  <User className="mr-2 h-5 w-5" />

                  Not Employed

                </Button>

              </div>

            </CardContent>

          </Card>

          {/\* Category Selection \*/}

          {employmentStatus && (

            <Card className="mb-8 shadow-lg border-0 bg-white">

              <CardHeader className="bg-gradient-to-r from-slate-50 to-gray-100">

                <CardTitle className="text-xl text-gray-800">Select Your Category</CardTitle>

              </CardHeader>

              <CardContent className="p-6">

                <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4 gap-4">

                  {userTypesMap[employmentStatus].map((type, idx) => (

                    <Button

                      key={idx}

                      variant="outline"

                      onClick={() => handleUserTypeSelect(type)}

                      className={`h-auto p-4 text-sm font-medium text-left whitespace-normal transition-all duration-300

                        ${userType === type

                          ? 'bg-gradient-to-r from-blue-500 to-indigo-600 text-white border-0 shadow-lg hover:from-blue-600 hover:to-indigo-700'

                          : 'bg-white text-gray-700 border-gray-300 hover:bg-gray-50 hover:border-gray-400'

                        }`}

                    >

                      {type}

                    </Button>

                  ))}

                </div>

              </CardContent>

            </Card>

          )}

          {/\* Profile Form \*/}

          {userType && features.length > 0 && (

            <Card className="mb-8 shadow-lg border-0 bg-white">

              <CardHeader className="bg-gradient-to-r from-slate-50 to-gray-100">

                <CardTitle className="text-xl text-gray-800">Complete Your Profile</CardTitle>

                <p className="text-gray-600">

                  Fill in the details for: <span className="font-medium text-blue-600">{userType}</span>

                </p>

              </CardHeader>

              <CardContent className="p-6">

                <ScrollArea className="h-96 pr-4">

                  <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-x-8 gap-y-6">

                    {features.map((feature, idx) => (

                      <div key={idx} className="flex flex-col space-y-2">

                        <Label className="text-sm font-medium text-gray-700">{feature}</Label>

                        {fieldOptions[feature] && fieldOptions[feature].length > 0 ? (

                          <select

                            value={formData[feature] || ''}

                            onChange={e => handleInputChange(feature, e.target.value)}

                            className="h-11 px-3 py-2 text-sm rounded-lg border border-gray-300 bg-white text-gray-700 focus:ring-2 focus:ring-blue-500 focus:border-transparent transition-all duration-200"

                          >

                            <option value="">Select {feature}</option>

                            {fieldOptions[feature].map(option => (

                              <option key={option} value={option}>{option}</option>

                            ))}

                          </select>

                        ) : (

                          <Input

                            placeholder={`Enter ${feature}`}

                            value={formData[feature] || ''}

                            onChange={e => handleInputChange(feature, e.target.value)}

                            className="h-11 px-3 py-2 text-sm text-gray-700 placeholder:text-gray-400 border-gray-300 focus:ring-2 focus:ring-blue-500 focus:border-transparent transition-all duration-200"

                          />

                        )}

                        {touched[feature] && (

                          !formData[feature] || formData[feature].trim() === "" ? (

                            <span className="text-xs text-red-600">Required</span>

                          ) : (

                            feature.toLowerCase().match(/age|year|gpa|score|amount|income|salary|count|duration|balance|expenses|tenure|months|margin/) &&

                            isNaN(Number(formData[feature])) && (

                              <span className="text-xs text-red-600">Must be a number</span>

                            )

                          )

                        )}

                      </div>

                    ))}

                  </div>

                </ScrollArea>

                <div className="mt-6 pt-4 border-t border-gray-200">

                  <Button

                    onClick={handleSubmit}

                    disabled={loading}

                    className="w-full h-12 text-lg font-semibold bg-gradient-to-r from-blue-500 to-indigo-600 text-white hover:from-blue-600 hover:to-indigo-700 transition-all duration-300 shadow-lg"

                  >

                    {loading ? (

                      <>

                        <Loader2 className="mr-2 h-5 w-5 animate-spin" />

                        Processing Prediction...

                      </>

                    ) : (

                      'Get Loan Prediction'

                    )}

                  </Button>

                </div>

              </CardContent>

            </Card>

          )}

          {/\* Error Display \*/}

          {error && (

            <Alert className="mb-6 border-red-200 bg-red-50">

              <AlertCircle className="h-4 w-4 text-red-600" />

              <AlertDescription className="text-red-800">{error}</AlertDescription>

            </Alert>

          )}

          {/\* Results Display \*/}

          {result && (

            <Card className="mb-8 shadow-lg border-0 bg-white">

              <CardHeader className="bg-gradient-to-r from-green-50 to-emerald-50">

                <CardTitle className="text-xl flex items-center gap-2 text-gray-800">

                  <CheckCircle className="h-5 w-5 text-green-600" />

                  Loan Prediction Results

                </CardTitle>

              </CardHeader>

              <CardContent className="p-6">

                <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4 gap-6">

                  {getGradientBlock(

                    "blue",

                    <>

                      <div className="text-2xl font-bold text-blue-700 mb-2">{formatCurrency(result.eligible\_amount)}</div>

                      <div className="text-sm font-medium text-gray-600">Eligible Amount</div>

                    </>

                  )}

                  {getGradientBlock(

                    "purple",

                    <>

                      <div className="text-2xl font-bold text-purple-700 mb-2">{result.base\_rate}%</div>

                      <div className="text-sm font-medium text-gray-600">Base Rate</div>

                    </>

                  )}

                  {getGradientBlock(

                    "green",

                    <>

                      <div className="text-2xl font-bold text-green-700 mb-2">{result.discount}%</div>

                      <div className="text-sm font-medium text-gray-600">Discount</div>

                    </>

                  )}

                  {getGradientBlock(

                    "orange",

                    <>

                      <div className="text-2xl font-bold text-orange-700 mb-2">{result.effective\_rate}%</div>

                      <div className="text-sm font-medium text-gray-600">Effective Rate</div>

                    </>

                  )}

                  {getGradientBlock(

                    "indigo",

                    <>

                      <div className="text-2xl font-bold text-indigo-700 mb-2">{result.tenure}</div>

                      <div className="text-sm font-medium text-gray-600">Tenure (months)</div>

                    </>

                  )}

                  {getGradientBlock(

                    "yellow",

                    <>

                      <div className="text-2xl font-bold text-yellow-700 mb-2">{formatCurrency(result.emi)}</div>

                      <div className="text-sm font-medium text-gray-600">EMI</div>

                    </>

                  )}

                  {getGradientBlock(

                    "pink",

                    <>

                      <div className="text-2xl font-bold text-pink-700 mb-2">{result.risk}</div>

                      <div className="text-sm font-medium text-gray-600">Risk</div>

                    </>

                  )}

                </div>

              </CardContent>

            </Card>

          )}

          {/\* Explain Button \*/}

          {result && (

            <div className="mb-8 flex justify-center">

              <Button

                onClick={() => setShowExplain(s => !s)}

                className="bg-gradient-to-r from-gray-600 to-gray-700 text-white hover:from-gray-700 hover:to-gray-800 transition-all duration-300 shadow-lg px-8"

              >

                {showExplain ? 'Hide Explanation' : 'Explain Results'}

              </Button>

            </div>

          )}

          {/\* SHAP Explanation \*/}

          {showExplain && result?.shap\_values && (

            <Card className="mb-8 shadow-lg border-0 bg-white">

              <CardHeader className="bg-gradient-to-r from-slate-50 to-gray-100">

                <CardTitle className="text-xl flex items-center gap-2 text-gray-800">

                  <User className="h-5 w-5 text-gray-600" />

                  Feature Contributions Analysis

                </CardTitle>

              </CardHeader>

              <CardContent className="p-6">

                <div className="space-y-3">

                  {Object.entries(result.shap\_values)

                    .sort((a, b) => Math.abs(b[1]) - Math.abs(a[1]))

                    .map(([feat, val]) => (

                      <div key={feat} className="flex justify-between items-center py-2 px-3 bg-gray-50 rounded-lg">

                        <span className="font-medium text-gray-700">{feat}</span>

                        <span className={`font-bold ${val >= 0 ? "text-green-600" : "text-red-600"}`}>

                          {val >= 0 ? "+" : ""}{val.toFixed(2)}

                        </span>

                      </div>

                    ))}

                </div>

                <div className="mt-6 p-4 bg-blue-50 rounded-lg">

                  <p className="font-semibold text-gray-800 mb-3">Top 3 Impact Factors:</p>

                  <ul className="space-y-2 text-sm">

                    {Object.entries(result.shap\_values)

                      .sort((a, b) => Math.abs(b[1]) - Math.abs(a[1]))

                      .slice(0, 3)

                      .map(([feat, val]) => (

                        <li key={feat} className="flex items-start gap-2">

                          <div className={`w-2 h-2 rounded-full mt-2 ${val >= 0 ? 'bg-green-500' : 'bg-red-500'}`}></div>

                          <span className="text-gray-700">

                            Your <strong>{feat}</strong> {val >= 0 ? "positively contributed" : "negatively impacted"} your score by <strong>{Math.abs(val).toFixed(2)} points</strong>.

                          </span>

                        </li>

                      ))}

                  </ul>

                </div>

              </CardContent>

            </Card>

          )}

        </div>

      </div>

    </div>

  );

};

export default LoanPrediction;

**ROLE BASED PREDICTION**

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

import { User, CreditCard, Calculator, FileText, TrendingUp, Shield, Users, Building } from 'lucide-react';

interface UserData {

  [key: string]: string | number | boolean;

}

interface PredictionResponse {

  base\_interest\_rate?: number;

  discounted\_interest\_rate?: number;

  loan\_tenure?: number;

  risk\_rating?: string;

  confidence\_score?: number;

  timestamp: string;

}

interface ExplanationResponse {

  feature\_importance: { [key: string]: number };

  recommendations: string[];

  timestamp: string;

}

const USER\_TYPE\_FEATURES = {

  salaried: [

    { name: 'Age', label: 'Age', type: 'number' },

    { name: 'Income\_Level', label: 'Income Level (₹)', type: 'number' },

    { name: 'Loan\_Amount', label: 'Loan Amount (₹)', type: 'number' },

    { name: 'Credit\_Score', label: 'Credit Score', type: 'number' },

    { name: 'Existing\_FD\_Amount', label: 'Existing FD Amount (₹)', type: 'number' },

    { name: 'FD\_Tenure\_Months', label: 'FD Tenure (Months)', type: 'number' },

    { name: 'Years\_of\_Experience', label: 'Years of Experience', type: 'number' },

    { name: 'Job\_Sector\_Government', label: 'Government Job', type: 'checkbox' },

    { name: 'Job\_Sector\_PSU', label: 'PSU Job', type: 'checkbox' },

    { name: 'Job\_Sector\_Private', label: 'Private Job', type: 'checkbox' },

    { name: 'Job\_Sector\_Startup', label: 'Startup Job', type: 'checkbox' }

  ],

  business: [

    { name: 'Years\_in\_Operation', label: 'Years in Operation', type: 'number' },

    { name: 'Annual\_Revenue', label: 'Annual Revenue (₹)', type: 'number' },

    { name: 'Loan\_Amount', label: 'Loan Amount (₹)', type: 'number' },

    { name: 'Monthly\_Revenue\_Variance', label: 'Monthly Revenue Variance (%)', type: 'number' },

    { name: 'Credit\_Score', label: 'Credit Score', type: 'number' },

    { name: 'GST\_Return\_Score', label: 'GST Return Score', type: 'number' },

    { name: 'ITR\_Compliance\_Score', label: 'ITR Compliance Score', type: 'number' },

    { name: 'Banking\_History\_Score', label: 'Banking History Score', type: 'number' },

    { name: 'Collateral\_Value', label: 'Collateral Value (₹)', type: 'number' },

    { name: 'Collateral\_Coverage\_Ratio', label: 'Collateral Coverage Ratio', type: 'number' },

    { name: 'Revenue\_to\_Loan\_Ratio', label: 'Revenue to Loan Ratio', type: 'number' },

    { name: 'Num\_of\_Defaults', label: 'Number of Defaults', type: 'number' },

    { name: 'Market\_Trend\_Impact', label: 'Market Trend Impact', type: 'number' }

  ],

  hnwi: [

    { name: 'Age', label: 'Age', type: 'number' },

    { name: 'Net\_Worth\_Cr', label: 'Net Worth (Crores)', type: 'number' },

    { name: 'Annual\_Income', label: 'Annual Income (₹)', type: 'number' },

    { name: 'Business\_Interest', label: 'Business Interest', type: 'number' },

    { name: 'Investment\_Portfolio\_Diversity', label: 'Portfolio Diversity (%)', type: 'number' },

    { name: 'Real\_Estate\_Value', label: 'Real Estate Value (₹)', type: 'number' },

    { name: 'Existing\_FD\_Amount', label: 'Existing FD Amount (₹)', type: 'number' },

    { name: 'Loan\_Amount', label: 'Loan Amount (₹)', type: 'number' },

    { name: 'Credit\_Score', label: 'Credit Score', type: 'number' },

    { name: 'Luxury\_Expenditure\_%', label: 'Luxury Expenditure (%)', type: 'number' },

    { name: 'Bank\_Relationship', label: 'Bank Relationship Score', type: 'number' },

    { name: 'Repayment\_History', label: 'Repayment History Score', type: 'number' },

    { name: 'Market\_Trend\_Impact', label: 'Market Trend Impact', type: 'number' },

    { name: 'Debt\_to\_Asset\_Ratio', label: 'Debt to Asset Ratio', type: 'number' },

    { name: 'Spending\_to\_Income\_Ratio', label: 'Spending to Income Ratio', type: 'number' }

  ],

  women: [

    { name: 'Age', label: 'Age', type: 'number' },

    { name: 'Monthly\_Income', label: 'Monthly Income (₹)', type: 'number' },

    { name: 'Existing\_Loan', label: 'Existing Loan Amount (₹)', type: 'number' },

    { name: 'Existing\_FD\_Amount', label: 'Existing FD Amount (₹)', type: 'number' },

    { name: 'Loan\_Amount', label: 'Loan Amount (₹)', type: 'number' },

    { name: 'Credit\_Score', label: 'Credit Score', type: 'number' },

    { name: 'Financial\_Literacy\_Score', label: 'Financial Literacy Score', type: 'number' },

    { name: 'Has\_Dependent\_Children', label: 'Has Dependent Children', type: 'checkbox' },

    { name: 'Marital\_Status', label: 'Marital Status', type: 'select', options: ['single', 'married', 'divorced', 'widowed'] },

    { name: 'Employment\_Status', label: 'Employment Status', type: 'select', options: ['salaried', 'self-employed', 'business', 'unemployed', 'retired'] },

    { name: 'Bank\_Affiliation', label: 'Bank Affiliation Score', type: 'number' },

    { name: 'Repayment\_Behavior', label: 'Repayment Behavior Score', type: 'number' },

    { name: 'Market\_Trend\_Impact', label: 'Market Trend Impact', type: 'number' }

  ]

};

const RoleBase: React.FC = () => {

  const [userType, setUserType] = useState<string>('salaried');

  const [userData, setUserData] = useState<UserData>({});

  const [prediction, setPrediction] = useState<PredictionResponse | null>(null);

  const [explanation, setExplanation] = useState<ExplanationResponse | null>(null);

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

  const [activeTab, setActiveTab] = useState('assessment');

  const userTypeIcons = {

    salaried: <User className="w-5 h-5" />,

    business: <Building className="w-5 h-5" />,

    hnwi: <TrendingUp className="w-5 h-5" />,

    women: <Users className="w-5 h-5" />

  };

  const userTypeLabels = {

    salaried: 'Structured Earners',

    business: 'Enterprise Operators',

    hnwi: 'Established Individuals',

    women: 'Emerging Participants'

  };

  const handleInputChange = (name: string, value: string | number | boolean) => {

    setUserData(prev => ({

      ...prev,

      [name]: value

    }));

  };

  const handleSubmit = async (type: 'assessment' | 'analysis') => {

    setLoading(true);

    try {

      const endpoint = type === 'assessment' ? 'loan-assessment' : 'feature-analysis';

      const userTypeEndpoint = userType === 'salaried' ? 'salaried-employee' :

                              userType === 'business' ? 'business-owner' :

                              userType === 'hnwi' ? 'hnwi' :

                              'women-entrepreneur';

      const response = await fetch(`/api/v1/role-based/${userTypeEndpoint}/${endpoint}`, {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

        },

        body: JSON.stringify({

          user\_type: userType,

          user\_data: userData

        })

      });

      const data = await response.json();

      if (type === 'assessment') {

        setPrediction(data);

      } else {

        setExplanation(data);

      }

    } catch (error) {

      console.error('Error:', error);

    } finally {

      setLoading(false);

    }

  };

  const generateReport = async () => {

    if (!prediction || !explanation) {

      alert('Please run both assessment and analysis first');

      return;

    }

    setLoading(true);

    try {

      const userTypeEndpoint = userType === 'salaried' ? 'salaried-employee' :

                              userType === 'business' ? 'business-owner' :

                              userType === 'hnwi' ? 'hnwi' :

                              'women-entrepreneur';

      const response = await fetch(`/api/v1/role-based/${userTypeEndpoint}/generate-report`, {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

        },

        body: JSON.stringify({

          user\_profile: userData,

          prediction: prediction,

          explanation: explanation.feature\_importance,

          include\_charts: true

        })

      });

      const blob = await response.blob();

      const url = window.URL.createObjectURL(blob);

      const a = document.createElement('a');

      a.href = url;

      a.download = `apexwolf\_${userType}\_report\_${new Date().toISOString().slice(0, 10)}.pdf`;

      document.body.appendChild(a);

      a.click();

      window.URL.revokeObjectURL(url);

      document.body.removeChild(a);

    } catch (error) {

      console.error('Error generating report:', error);

    } finally {

      setLoading(false);

    }

  };

  const renderFormFields = () => {

    const fields = USER\_TYPE\_FEATURES[userType as keyof typeof USER\_TYPE\_FEATURES] || [];

    return (

      <div className="grid grid-cols-2 gap-4">

        {fields.map((field, index) => (

          <div key={field.name} className="space-y-2">

            <label className="block text-sm font-medium text-gray-700">

              {field.label}

            </label>

            {field.type === 'number' && (

              <input

                type="number"

                className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-transparent transition-all duration-200"

                value={userData[field.name] || ''}

                onChange={(e) => handleInputChange(field.name, parseFloat(e.target.value) || 0)}

                placeholder={`Enter ${field.label.toLowerCase()}`}

              />

            )}

            {field.type === 'checkbox' && (

              <div className="flex items-center space-x-2">

                <input

                  type="checkbox"

                  className="w-4 h-4 text-blue-600 border-gray-300 rounded focus:ring-blue-500"

                  checked={!!userData[field.name]}

                  onChange={(e) => handleInputChange(field.name, e.target.checked)}

                />

                <span className="text-sm text-gray-600">Yes</span>

              </div>

            )}

            {field.type === 'select' && (

              <select

                className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-transparent transition-all duration-200"

                value={userData[field.name] || ''}

                onChange={(e) => handleInputChange(field.name, e.target.value)}

              >

                <option value="">Select {field.label}</option>

                {field.options?.map(option => (

                  <option key={option} value={option}>

                    {option.charAt(0).toUpperCase() + option.slice(1)}

                  </option>

                ))}

              </select>

            )}

          </div>

        ))}

      </div>

    );

  };

  const getRiskColor = (risk: string) => {

    switch (risk?.toLowerCase()) {

      case 'low': return 'text-green-600 bg-green-50';

      case 'medium': return 'text-yellow-600 bg-yellow-50';

      case 'high': return 'text-red-600 bg-red-50';

      default: return 'text-gray-600 bg-gray-50';

    }

  };

  return (

    <div className="min-h-screen bg-white">

      <div className="container mx-auto px-4 py-8">

        {/\* Header \*/}

        <div className="text-center mb-8">

          <p className="text-lg text-gray-600">Role-Based Loan Assessment & Financial Analysis</p>

        </div>

        {/\* User Type Selection \*/}

        <div className="bg-gray-50 rounded-xl p-6 mb-8">

          <h2 className="text-xl font-semibold text-gray-800 mb-4">Select Your Profile</h2>

          <div className="grid grid-cols-2 md:grid-cols-4 gap-4">

            {Object.entries(userTypeLabels).map(([type, label]) => (

              <button

                key={type}

                onClick={() => setUserType(type)}

                className={`p-4 rounded-lg border-2 transition-all duration-200 flex flex-col items-center space-y-2 ${

                  userType === type

                    ? 'border-blue-500 bg-gradient-to-r from-blue-500 to-purple-600 text-white shadow-lg'

                    : 'border-gray-200 bg-white text-gray-700 hover:border-blue-300 hover:shadow-md'

                }`}

              >

                {userTypeIcons[type as keyof typeof userTypeIcons]}

                <span className="text-sm font-medium">{label}</span>

              </button>

            ))}

          </div>

        </div>

        {/\* Main Content \*/}

        <div className="bg-gray-50 rounded-xl p-6">

          {/\* Navigation Tabs \*/}

          <div className="flex space-x-4 mb-6">

            <button

              onClick={() => setActiveTab('assessment')}

              className={`px-6 py-3 rounded-lg font-medium transition-all duration-200 ${

                activeTab === 'assessment'

                  ? 'bg-gradient-to-r from-blue-500 to-purple-600 text-white shadow-lg'

                  : 'bg-white text-gray-700 hover:shadow-md'

              }`}

            >

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

              Loan Assessment

            </button>

            <button

              onClick={() => setActiveTab('analysis')}

              className={`px-6 py-3 rounded-lg font-medium transition-all duration-200 ${

                activeTab === 'analysis'

                  ? 'bg-gradient-to-r from-blue-500 to-purple-600 text-white shadow-lg'

                  : 'bg-white text-gray-700 hover:shadow-md'

              }`}

            >

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

              Feature Analysis

            </button>

          </div>

          {/\* Form \*/}

          <div className="bg-white rounded-lg p-6 mb-6">

            <h3 className="text-lg font-semibold text-gray-800 mb-4">

              {userTypeLabels[userType as keyof typeof userTypeLabels]} Information

            </h3>

            {renderFormFields()}

          </div>

          {/\* Action Buttons \*/}

          <div className="flex flex-wrap gap-4 mb-6">

            <button

              onClick={() => handleSubmit('assessment')}

              disabled={loading}

              className="px-6 py-3 bg-gradient-to-r from-blue-500 to-purple-600 text-white rounded-lg font-medium hover:from-blue-600 hover:to-purple-700 transition-all duration-200 shadow-lg hover:shadow-xl disabled:opacity-50"

            >

              {loading ? 'Processing...' : 'Run Assessment'}

            </button>

            <button

              onClick={() => handleSubmit('analysis')}

              disabled={loading}

              className="px-6 py-3 bg-gradient-to-r from-purple-500 to-indigo-600 text-white rounded-lg font-medium hover:from-purple-600 hover:to-indigo-700 transition-all duration-200 shadow-lg hover:shadow-xl disabled:opacity-50"

            >

              {loading ? 'Analyzing...' : 'Feature Analysis'}

            </button>

            <button

              onClick={generateReport}

              disabled={loading || !prediction || !explanation}

              className="px-6 py-3 bg-gradient-to-r from-indigo-500 to-blue-600 text-white rounded-lg font-medium hover:from-indigo-600 hover:to-blue-700 transition-all duration-200 shadow-lg hover:shadow-xl disabled:opacity-50"

            >

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

              Generate Report

            </button>

          </div>

          {/\* Results \*/}

          {prediction && (

            <div className="bg-white rounded-lg p-6 mb-6">

              <h3 className="text-lg font-semibold text-gray-800 mb-4 flex items-center">

                <CreditCard className="w-5 h-5 mr-2" />

                Loan Assessment Results

              </h3>

              <div className="grid grid-cols-2 md:grid-cols-4 gap-4">

                <div className="bg-blue-50 p-4 rounded-lg">

                  <p className="text-sm text-blue-600 font-medium">Base Interest Rate</p>

                  <p className="text-2xl font-bold text-blue-800">{prediction.base\_interest\_rate}%</p>

                </div>

                <div className="bg-purple-50 p-4 rounded-lg">

                  <p className="text-sm text-purple-600 font-medium">Discounted Rate</p>

                  <p className="text-2xl font-bold text-purple-800">{prediction.discounted\_interest\_rate}%</p>

                </div>

                <div className="bg-indigo-50 p-4 rounded-lg">

                  <p className="text-sm text-indigo-600 font-medium">Loan Tenure</p>

                  <p className="text-2xl font-bold text-indigo-800">{prediction.loan\_tenure} months</p>

                </div>

                <div className={`p-4 rounded-lg ${getRiskColor(prediction.risk\_rating || '')}`}>

                  <p className="text-sm font-medium opacity-80">Risk Rating</p>

                  <p className="text-2xl font-bold flex items-center">

                    <Shield className="w-5 h-5 mr-1" />

                    {prediction.risk\_rating}

                  </p>

                </div>

              </div>

              <div className="mt-4 bg-gray-50 p-4 rounded-lg">

                <p className="text-sm text-gray-600">Confidence Score: <span className="font-bold">{prediction.confidence\_score}</span></p>

              </div>

            </div>

          )}

          {explanation && (

            <div className="bg-white rounded-lg p-6">

              <h3 className="text-lg font-semibold text-gray-800 mb-4 flex items-center">

                <TrendingUp className="w-5 h-5 mr-2" />

                Feature Analysis Results

              </h3>

              {/\* Feature Importance \*/}

              <div className="mb-6">

                <h4 className="font-medium text-gray-800 mb-3">Feature Importance</h4>

                <div className="space-y-2">

                  {Object.entries(explanation.feature\_importance)

                    .sort(([,a], [,b]) => Math.abs(b) - Math.abs(a))

                    .slice(0, 8)

                    .map(([feature, score]) => (

                      <div key={feature} className="flex items-center justify-between bg-gray-50 p-3 rounded-lg">

                        <span className="text-sm font-medium text-gray-700">

                          {feature.replace(/\_/g, ' ').replace(/\b\w/g, l => l.toUpperCase())}

                        </span>

                        <div className="flex items-center">

                          <div className={`w-20 h-2 rounded-full mr-2 ${score > 0 ? 'bg-green-200' : 'bg-red-200'}`}>

                            <div

                              className={`h-full rounded-full ${score > 0 ? 'bg-green-500' : 'bg-red-500'}`}

                              style={{ width: `${Math.abs(score) \* 100}%` }}

                            />

                          </div>

                          <span className={`text-sm font-bold ${score > 0 ? 'text-green-600' : 'text-red-600'}`}>

                            {score.toFixed(3)}

                          </span>

                        </div>

                      </div>

                    ))}

                </div>

              </div>

              {/\* Recommendations \*/}

              {explanation.recommendations && explanation.recommendations.length > 0 && (

                <div>

                  <h4 className="font-medium text-gray-800 mb-3">Personalized Recommendations</h4>

                  <div className="bg-blue-50 p-4 rounded-lg">

                    <ul className="space-y-2">

                      {explanation.recommendations.map((rec, index) => (

                        <li key={index} className="text-sm text-blue-800 flex items-start">

                          <span className="mr-2">•</span>

                          <span>{rec}</span>

                        </li>

                      ))}

                    </ul>

                  </div>

                </div>

              )}

            </div>

          )}

        </div>

      </div>

    </div>

  );

};

export default RoleBase;

**LOAN SIMULATOR**

import { useEffect, useState } from 'react'

import { Pie } from 'react-chartjs-2'

import { Chart as ChartJS, ArcElement, Tooltip, Legend } from 'chart.js'

import ChartDataLabels from 'chartjs-plugin-datalabels'

import jsPDF from 'jspdf'

import autoTable from 'jspdf-autotable'

import { Calculator, Download, TrendingUp, DollarSign, Calendar, CreditCard } from 'lucide-react'

ChartJS.register(ArcElement, Tooltip, Legend, ChartDataLabels)

export default function LoanSimulator() {

  // State variables for loan simulator fields

  const [amount, setAmount] = useState(1000000)

  const [tenure, setTenure] = useState(36)

  const [creditScore, setCreditScore] = useState(750)

  const [rate, setRate] = useState(8.0)

  const [emi, setEmi] = useState(0)

  const [totalPayment, setTotalPayment] = useState(0)

  const [totalInterest, setTotalInterest] = useState(0)

  const [schedule, setSchedule] = useState([])

  const [error, setError] = useState('')

  const [isLoading, setIsLoading] = useState(false)

  // On change of input fields, fetch EMI calculation if inputs are valid

  useEffect(() => {

    if (validateInputs()) {

      fetchEMI()

    } else {

      setEmi(0)

      setTotalInterest(0)

      setTotalPayment(0)

      setSchedule([])

    }

    // eslint-disable-next-line

  }, [amount, tenure, creditScore])

  // Validate all user inputs before calculation

  const validateInputs = () => {

    // Loan amount must be between 100,000 and 5,000,000

    if (isNaN(amount) || amount < 100000 || amount > 5000000) {

      setError("Loan Amount must be between 1,00,000 and 50,00,000")

      return false

    }

    // Tenure must be between 6 and 360 months

    if (isNaN(tenure) || tenure < 6 || tenure > 360) {

      setError("Loan Tenure must be between 6 and 360 months")

      return false

    }

    // Credit score must be between 300 and 900

    if (isNaN(creditScore) || creditScore < 300 || creditScore > 900) {

      setError("Credit Score must be between 300 and 900")

      return false

    }

    // Clear any previous error message if all inputs are valid

    setError("")

    return true

  }

  // Fetch EMI calculation from backend if validation passes

  const fetchEMI = async () => {

    setIsLoading(true)

    try {

      const res = await fetch("http://localhost:5000/api/loan/simulate", {

        method: "POST",

        headers: { "Content-Type": "application/json" },

        body: JSON.stringify({ amount, tenure, credit\_score: creditScore })

      })

      const data = await res.json()

      if (res.ok) {

        setRate(data.rate\_used)

        setEmi(data.emi)

        setTotalPayment(data.total\_payment)

        setTotalInterest(data.total\_interest)

        setSchedule(data.schedule)

      }

    } catch (err) {

      setError("Error fetching EMI schedule. Please try again later.")

    } finally {

      setIsLoading(false)

    }

  }

  // Generate PDF for EMI schedule table

  const downloadPDF = () => {

    const doc = new jsPDF()

    doc.text("Loan EMI Schedule", 14, 16)

    autoTable(doc, {

      startY: 20,

      head: [["Month", "EMI", "Principal", "Interest", "Balance"]],

      body: schedule.map((row) => [

        row.month,

        `₹${row.emi.toFixed(2)}`,

        `₹${row.principal.toFixed(2)}`,

        `₹${row.interest.toFixed(2)}`,

        `₹${row.balance.toFixed(2)}`

      ]),

      styles: { fontSize: 10 }

    })

    doc.save("emi-schedule.pdf")

  }

  const formatCurrency = (value) => {

    return new Intl.NumberFormat('en-IN', {

      style: 'currency',

      currency: 'INR',

      maximumFractionDigits: 0

    }).format(value)

  }

  const getCreditScoreColor = (score) => {

    if (score >= 750) return 'text-green-600 bg-green-50'

    if (score >= 650) return 'text-yellow-600 bg-yellow-50'

    return 'text-red-600 bg-red-50'

  }

  const getCreditScoreLabel = (score) => {

    if (score >= 750) return 'Excellent'

    if (score >= 700) return 'Good'

    if (score >= 650) return 'Fair'

    return 'Poor'

  }

  return (

    <div className="min-h-screen bg-gradient-to-br from-blue-50 via-indigo-50 to-purple-50 py-8 px-4">

      <div className="max-w-7xl mx-auto">

        {/\* Header \*/}

        <div className="text-center mb-10">

          <div className="flex items-center justify-center mb-4">

            <div className="bg-blue-600 p-3 rounded-full mr-3">

              <Calculator className="h-8 w-8 text-white" />

            </div>

            <h1 className="text-4xl font-bold text-gray-800">EMI Calculator</h1>

          </div>

          <p className="text-gray-600 text-lg">Calculate your loan EMI and plan your finances with ease</p>

        </div>

        <div className="bg-white rounded-2xl shadow-2xl overflow-hidden">

          {/\* Main Content \*/}

          <div className="p-8">

            <div className="grid lg:grid-cols-3 gap-8">

              {/\* Input Controls \*/}

              <div className="lg:col-span-2 space-y-8">

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

                  {/\* Loan Amount \*/}

                  <div className="bg-gradient-to-r from-blue-50 to-indigo-50 p-6 rounded-xl border border-blue-100">

                    <div className="flex items-center mb-3">

                      <DollarSign className="h-5 w-5 text-blue-600 mr-2" />

                      <label className="block font-semibold text-gray-700">Loan Amount</label>

                    </div>

                    <div className="text-2xl font-bold text-blue-600 mb-3">

                      {formatCurrency(amount)}

                    </div>

                    <input

                      type="number"

                      value={amount}

                      onChange={(e) => setAmount(+e.target.value)}

                      className="w-full border-2 border-blue-200 px-4 py-3 rounded-lg mb-3 focus:border-blue-500 focus:outline-none transition-colors"

                      placeholder="Enter loan amount"

                    />

                    <input

                      type="range"

                      min={100000}

                      max={5000000}

                      step={10000}

                      value={amount}

                      onChange={(e) => setAmount(+e.target.value)}

                      className="w-full h-2 bg-blue-200 rounded-lg appearance-none cursor-pointer slider"

                    />

                    <div className="flex justify-between text-sm text-gray-500 mt-1">

                      <span>₹1L</span>

                      <span>₹50L</span>

                    </div>

                  </div>

                  {/\* Credit Score \*/}

                  <div className="bg-gradient-to-r from-green-50 to-emerald-50 p-6 rounded-xl border border-green-100">

                    <div className="flex items-center mb-3">

                      <CreditCard className="h-5 w-5 text-green-600 mr-2" />

                      <label className="block font-semibold text-gray-700">Credit Score</label>

                    </div>

                    <div className="flex items-center mb-3">

                      <span className="text-2xl font-bold text-green-600 mr-3">{creditScore}</span>

                      <span className={`px-3 py-1 rounded-full text-sm font-medium ${getCreditScoreColor(creditScore)}`}>

                        {getCreditScoreLabel(creditScore)}

                      </span>

                    </div>

                    <input

                      type="number"

                      min={300}

                      max={900}

                      value={creditScore}

                      onChange={(e) => setCreditScore(+e.target.value)}

                      className="w-full border-2 border-green-200 px-4 py-3 rounded-lg mb-3 focus:border-green-500 focus:outline-none transition-colors"

                    />

                    <input

                      type="range"

                      min={300}

                      max={900}

                      step={1}

                      value={creditScore}

                      onChange={(e) => setCreditScore(+e.target.value)}

                      className="w-full h-2 bg-green-200 rounded-lg appearance-none cursor-pointer"

                    />

                    <div className="flex justify-between text-sm text-gray-500 mt-1">

                      <span>300</span>

                      <span>900</span>

                    </div>

                  </div>

                  {/\* Loan Tenure \*/}

                  <div className="bg-gradient-to-r from-purple-50 to-pink-50 p-6 rounded-xl border border-purple-100">

                    <div className="flex items-center mb-3">

                      <Calendar className="h-5 w-5 text-purple-600 mr-2" />

                      <label className="block font-semibold text-gray-700">Loan Tenure</label>

                    </div>

                    <div className="text-2xl font-bold text-purple-600 mb-3">

                      {tenure} months ({(tenure/12).toFixed(1)} years)

                    </div>

                    <input

                      type="number"

                      value={tenure}

                      onChange={(e) => setTenure(+e.target.value)}

                      className="w-full border-2 border-purple-200 px-4 py-3 rounded-lg mb-3 focus:border-purple-500 focus:outline-none transition-colors"

                    />

                    <input

                      type="range"

                      min={6}

                      max={360}

                      step={1}

                      value={tenure}

                      onChange={(e) => setTenure(+e.target.value)}

                      className="w-full h-2 bg-purple-200 rounded-lg appearance-none cursor-pointer"

                    />

                    <div className="flex justify-between text-sm text-gray-500 mt-1">

                      <span>6 months</span>

                      <span>30 years</span>

                    </div>

                  </div>

                  {/\* Interest Rate \*/}

                  <div className="bg-gradient-to-r from-orange-50 to-red-50 p-6 rounded-xl border border-orange-100">

                    <div className="flex items-center mb-3">

                      <TrendingUp className="h-5 w-5 text-orange-600 mr-2" />

                      <label className="block font-semibold text-gray-700">Interest Rate</label>

                      <span className="ml-2 text-xs bg-gray-200 px-2 py-1 rounded">Auto-calculated</span>

                    </div>

                    <div className="text-2xl font-bold text-orange-600 mb-3">

                      {rate.toFixed(2)}% p.a.

                    </div>

                    <input

                      type="text"

                      value={`${rate.toFixed(2)}% per annum`}

                      readOnly

                      className="w-full bg-gradient-to-r from-gray-50 to-gray-100 border-2 border-gray-200 px-4 py-3 rounded-lg text-gray-600 cursor-not-allowed"

                    />

                    <p className="text-sm text-gray-500 mt-2">Rate Set By Admin</p>

                  </div>

                </div>

              </div>

              {/\* Chart \*/}

              <div className="flex flex-col items-center justify-center bg-gradient-to-br from-indigo-50 to-blue-50 p-6 rounded-xl border border-indigo-100">

                <h3 className="text-lg font-semibold text-gray-700 mb-6 text-center">Loan Breakdown</h3>

                {isLoading ? (

                  <div className="flex items-center justify-center h-64">

                    <div className="animate-spin rounded-full h-16 w-16 border-b-2 border-blue-600"></div>

                  </div>

                ) : (

                  <div className="w-full max-w-sm">

                    <Pie

                      data={{

                        labels: ["Principal", "Interest"],

                        datasets: [

                          {

                            data: [amount, totalInterest],

                            backgroundColor: ["#87ceeb", "#1e3a8a"],

                            borderWidth: 2,

                            borderColor: "#ffffff",

                            hoverBackgroundColor: ["#87ceeb", "#1e40af"],

                            hoverBorderColor: "#ffffff",

                            hoverBorderWidth: 3

                          }

                        ]

                      }}

                      options={{

                        responsive: true,

                        maintainAspectRatio: false,

                        plugins: {

                          legend: {

                            position: "bottom",

                            labels: {

                              padding: 20,

                              font: { size: 14, weight: 'bold' },

                              color: '#374151'

                            }

                          },

                          datalabels: {

                            color: "#fff",

                            formatter: (value, ctx) => {

                              const total = ctx.chart.data.datasets[0].data.reduce((a, b) => a + b, 0)

                              return total > 0 ? `${((value / total) \* 100).toFixed(1)}%` : '0%'

                            },

                            font: { weight: "bold", size: 16 }

                          }

                        }

                      }}

                      plugins={[ChartDataLabels]}

                      height={280}

                    />

                  </div>

                )}

              </div>

            </div>

            {/\* Error Message \*/}

            {error && (

              <div className="mt-6 bg-red-50 border-l-4 border-red-400 p-4 rounded-r-lg animate-pulse">

                <div className="flex">

                  <div className="ml-3">

                    <p className="text-red-700 font-medium">{error}</p>

                  </div>

                </div>

              </div>

            )}

            {/\* Results Cards \*/}

            {!error && emi > 0 && (

              <div className="mt-8 grid md:grid-cols-3 gap-6">

                <div className="bg-gradient-to-r from-blue-500 to-blue-600 p-6 rounded-xl text-white shadow-lg transform hover:scale-105 transition-transform">

                  <div className="flex items-center justify-between mb-2">

                    <h3 className="text-lg font-semibold">Monthly EMI</h3>

                    <Calculator className="h-6 w-6 opacity-80" />

                  </div>

                  <p className="text-3xl font-bold">{formatCurrency(emi)}</p>

                </div>

                <div className="bg-gradient-to-r from-green-500 to-green-600 p-6 rounded-xl text-white shadow-lg transform hover:scale-105 transition-transform">

                  <div className="flex items-center justify-between mb-2">

                    <h3 className="text-lg font-semibold">Total Interest</h3>

                    <TrendingUp className="h-6 w-6 opacity-80" />

                  </div>

                  <p className="text-3xl font-bold">{formatCurrency(totalInterest)}</p>

                </div>

                <div className="bg-gradient-to-r from-purple-500 to-purple-600 p-6 rounded-xl text-white shadow-lg transform hover:scale-105 transition-transform">

                  <div className="flex items-center justify-between mb-2">

                    <h3 className="text-lg font-semibold">Total Payment</h3>

                    <DollarSign className="h-6 w-6 opacity-80" />

                  </div>

                  <p className="text-3xl font-bold">{formatCurrency(totalPayment)}</p>

                </div>

              </div>

            )}

            {/\* EMI Schedule Table \*/}

            {schedule.length > 0 && (

              <div className="mt-10">

                <div className="flex items-center justify-between mb-6">

                  <h3 className="text-2xl font-bold text-gray-800">EMI Payment Schedule</h3>

                  <button

                    onClick={downloadPDF}

                    className="bg-gradient-to-r from-blue-600 to-blue-700 text-white px-6 py-3 rounded-lg hover:from-blue-700 hover:to-blue-800 transition-all duration-300 flex items-center shadow-lg hover:shadow-xl transform hover:-translate-y-1"

                  >

                    <Download className="h-5 w-5 mr-2" />

                    Download PDF

                  </button>

                </div>

                <div className="bg-white rounded-xl overflow-hidden shadow-lg border border-gray-200">

                  <div className="overflow-x-auto max-h-96">

                    <table className="min-w-full divide-y divide-gray-200">

                      <thead className="bg-gradient-to-r from-gray-50 to-gray-100 sticky top-0">

                        <tr>

                          <th className="px-6 py-4 text-left text-xs font-bold text-gray-700 uppercase tracking-wider">Month</th>

                          <th className="px-6 py-4 text-right text-xs font-bold text-gray-700 uppercase tracking-wider">EMI</th>

                          <th className="px-6 py-4 text-right text-xs font-bold text-gray-700 uppercase tracking-wider">Principal</th>

                          <th className="px-6 py-4 text-right text-xs font-bold text-gray-700 uppercase tracking-wider">Interest</th>

                          <th className="px-6 py-4 text-right text-xs font-bold text-gray-700 uppercase tracking-wider">Balance</th>

                        </tr>

                      </thead>

                      <tbody className="bg-white divide-y divide-gray-200">

                        {schedule.map((row, index) => (

                          <tr key={row.month} className={`hover:bg-gray-50 transition-colors ${index % 2 === 0 ? 'bg-white' : 'bg-gray-25'}`}>

                            <td className="px-6 py-4 whitespace-nowrap text-center font-medium text-gray-900">{row.month}</td>

                            <td className="px-6 py-4 whitespace-nowrap text-right font-semibold text-gray-900">₹{row.emi.toFixed(2)}</td>

                            <td className="px-6 py-4 whitespace-nowrap text-right text-blue-600 font-medium">₹{row.principal.toFixed(2)}</td>

                            <td className="px-6 py-4 whitespace-nowrap text-right text-red-600 font-medium">₹{row.interest.toFixed(2)}</td>

                            <td className="px-6 py-4 whitespace-nowrap text-right font-semibold text-gray-700">₹{row.balance.toFixed(2)}</td>

                          </tr>

                        ))}

                      </tbody>

                    </table>

                  </div>

                </div>

              </div>

            )}

          </div>

        </div>

      </div>

      <style jsx>{`

        .slider::-webkit-slider-thumb {

          appearance: none;

          height: 20px;

          width: 20px;

          border-radius: 50%;

          background: #3b82f6;

          cursor: pointer;

          box-shadow: 0 2px 6px rgba(59, 130, 246, 0.3);

        }

        .slider::-moz-range-thumb {

          height: 20px;

          width: 20px;

          border-radius: 50%;

          background: #3b82f6;

          cursor: pointer;

          border: none;

          box-shadow: 0 2px 6px rgba(59, 130, 246, 0.3);

        }

      `}</style>

    </div>

  )

}

**INTEREST RATE OPTIMIZATION**

import React, { useState, useEffect } from "react";

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

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

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

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

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

import { FileText, DollarSign, Percent, MessageSquare, Upload, CheckCircle, Clock, XCircle, Download, ExternalLink } from "lucide-react";

const Negotiation = () => {

  // State for user input, file upload, and submission status

  const [formData, setFormData] = useState({

    loan\_type: "",

    loan\_amount: "",

    current\_rate: "",

    expected\_rate: "",

    comment: "",

    other\_loan\_type: ""

  });

  const [file, setFile] = useState<File | null>(null);

  const [status, setStatus] = useState("");

  const [recentStatus, setRecentStatus] = useState<any>(null);

  const [verify, setVerify] = useState(false);

  // Validation error state

  const [error, setError] = useState("");

  // Fetch recent negotiation status for display

  useEffect(() => {

    fetch("http://localhost:5000/api/negotiation/status")

      .then(res => res.json())

      .then(data => {

        if (data.status !== "none") {

          setRecentStatus(data);

        }

      });

  }, []);

  // Handle all input and textarea changes

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

    setFormData({ ...formData, [e.target.name]: e.target.value });

  };

  // Validate all input fields before submit

  const validate = () => {

    if (!formData.loan\_type) {

      setError("Please select a loan type.");

      return false;

    }

    if (formData.loan\_type === "other" && !formData.other\_loan\_type.trim()) {

      setError("Specify your loan type.");

      return false;

    }

    if (!formData.loan\_amount || isNaN(Number(formData.loan\_amount)) || Number(formData.loan\_amount) <= 0) {

      setError("Enter a valid loan amount greater than zero.");

      return false;

    }

    if (!formData.current\_rate || isNaN(Number(formData.current\_rate)) || Number(formData.current\_rate) <= 0) {

      setError("Enter a valid current interest rate.");

      return false;

    }

    if (!formData.expected\_rate || isNaN(Number(formData.expected\_rate)) || Number(formData.expected\_rate) <= 0) {

      setError("Enter a valid expected interest rate.");

      return false;

    }

    if (!formData.comment.trim() || formData.comment.length < 10) {

      setError("Provide a detailed reason/comment (minimum 10 characters).");

      return false;

    }

    setError("");

    return true;

  };

  // Handle form submission with validation

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

    e.preventDefault();

    if (!verify) {

      setVerify(true);

      return;

    }

    if (!validate()) return;

    setStatus("Submitting...");

    const body = new FormData();

    const actualLoanType = formData.loan\_type === "other" ? formData.other\_loan\_type : formData.loan\_type;

    body.append("loan\_type", actualLoanType);

    body.append("loan\_amount", formData.loan\_amount);

    body.append("current\_rate", formData.current\_rate);

    body.append("expected\_rate", formData.expected\_rate);

    body.append("comment", formData.comment);

    if (file) {

      body.append("file", file);

    }

    try {

      const res = await fetch("http://localhost:5000/api/negotiation/request", {

        method: "POST",

        body

      });

      const result = await res.json();

      if (!res.ok) throw new Error(result.msg || "Submission failed");

      setStatus(result.msg);

      setVerify(false);

      setFormData({

        loan\_type: "",

        loan\_amount: "",

        current\_rate: "",

        expected\_rate: "",

        comment: "",

        other\_loan\_type: ""

      });

      setFile(null);

      const updated = await fetch("http://localhost:5000/api/negotiation/status");

      const data = await updated.json();

      setRecentStatus(data);

    } catch (err: any) {

      setStatus(err.message);

    }

  };

  const getStatusIcon = (status: string) => {

    switch (status) {

      case "approved":

        return <CheckCircle className="w-4 h-4 text-green-600" />;

      case "rejected":

        return <XCircle className="w-4 h-4 text-red-600" />;

      case "pending":

        return <Clock className="w-4 h-4 text-yellow-600" />;

      default:

        return <Clock className="w-4 h-4 text-gray-600" />;

    }

  };

  const getStatusBadge = (status: string) => {

    const baseClasses = "inline-flex items-center gap-2 px-3 py-2 rounded-lg text-sm font-medium";

    switch (status) {

      case "approved":

        return `${baseClasses} bg-green-100 text-green-800 border border-green-200`;

      case "rejected":

        return `${baseClasses} bg-red-100 text-red-800 border border-red-200`;

      case "pending":

        return `${baseClasses} bg-yellow-100 text-yellow-800 border border-yellow-200`;

      default:

        return `${baseClasses} bg-gray-100 text-gray-800 border border-gray-200`;

    }

  };

  return (

    <div className="min-h-screen bg-white">

      <div className="w-full">

        {/\* Header \*/}

        <div className="w-full bg-gradient-to-r from-indigo-400 to-purple-400 text-white py-6 px-6">

          <div className="max-w-7xl mx-auto flex flex-col items-center text-center">

            <div className="flex items-center gap-4 mb-2">

              <div className="flex items-center justify-center w-12 h-12 bg-white/20 rounded-xl">

                <FileText className="w-6 h-6" />

              </div>

              <h1 className="text-2xl font-bold">Loan Flexibility Request</h1>

            </div>

            <p className="text-indigo-100 text-base opacity-90">Submit your request for better loan terms and conditions</p>

          </div>

        </div>

        {/\* Main Content \*/}

        <div className="max-w-7xl mx-auto p-6">

          {/\* Main Form - Full Width \*/}

          <div className="mb-6">

            <Card className="shadow-lg rounded-xl overflow-hidden border border-gray-200">

              <CardHeader className="bg-gray-50 border-b border-gray-200 p-4">

                <CardTitle className="text-lg font-bold flex items-center gap-2 text-gray-900">

                  <DollarSign className="w-5 h-5 text-indigo-600" />

                  Request Form

                </CardTitle>

                <p className="text-gray-600 mt-1 text-sm">Fill out the details below to submit your request</p>

              </CardHeader>

              <CardContent className="p-6 bg-white">

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

                  {/\* Loan Information \*/}

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

                    <div className="flex items-center gap-2 text-base font-semibold text-gray-900 border-b border-gray-200 pb-2">

                      <FileText className="w-4 h-4 text-indigo-600" />

                      Loan Information

                    </div>

                    <div className="grid grid-cols-1 lg:grid-cols-2 gap-4">

                      <div className="space-y-2">

                        <Label className="text-sm font-medium text-gray-700">Loan Type</Label>

                        <select

                          name="loan\_type"

                          value={formData.loan\_type}

                          onChange={handleChange}

                          className="w-full h-10 px-3 text-sm border border-gray-300 rounded-lg focus:ring-2 focus:ring-indigo-500 focus:border-indigo-500 transition-all duration-200 bg-white hover:border-gray-400"

                          required

                        >

                          <option value="">Select Loan Type</option>

                          <option value="home">🏠 Home Loan</option>

                          <option value="personal">👤 Personal Loan</option>

                          <option value="education">🎓 Education Loan</option>

                          <option value="other">📋 Other</option>

                        </select>

                        {formData.loan\_type === "other" && (

                          <div className="mt-2 animate-in slide-in-from-top duration-200">

                            <Input

                              name="other\_loan\_type"

                              value={formData.other\_loan\_type}

                              onChange={handleChange}

                              className="h-10 px-3 text-sm border border-gray-300 focus:ring-2 focus:ring-indigo-500 focus:border-indigo-500 transition-all duration-200 hover:border-gray-400"

                              placeholder="Enter your loan type"

                              required

                            />

                          </div>

                        )}

                      </div>

                      <div className="space-y-2">

                        <Label className="text-sm font-medium text-gray-700 flex items-center gap-2">

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

                          Loan Amount

                        </Label>

                        <Input

                          name="loan\_amount"

                          type="number"

                          value={formData.loan\_amount}

                          onChange={handleChange}

                          className="h-10 px-3 text-sm border border-gray-300 focus:ring-2 focus:ring-indigo-500 focus:border-indigo-500 transition-all duration-200 hover:border-gray-400"

                          placeholder="Enter amount"

                          required

                        />

                      </div>

                    </div>

                  </div>

                  {/\* Interest Rates \*/}

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

                    <div className="flex items-center gap-2 text-base font-semibold text-gray-900 border-b border-gray-200 pb-2">

                      <Percent className="w-4 h-4 text-indigo-600" />

                      Interest Rates

                    </div>

                    <div className="grid grid-cols-1 lg:grid-cols-2 gap-4">

                      <div className="space-y-2">

                        <Label className="text-sm font-medium text-gray-700">Current Rate (%)</Label>

                        <Input

                          name="current\_rate"

                          type="number"

                          step="0.01"

                          value={formData.current\_rate}

                          onChange={handleChange}

                          className="h-10 px-3 text-sm border border-gray-300 focus:ring-2 focus:ring-indigo-500 focus:border-indigo-500 transition-all duration-200 hover:border-gray-400"

                          placeholder="e.g., 8.5"

                          required

                        />

                      </div>

                      <div className="space-y-2">

                        <Label className="text-sm font-medium text-gray-700">Expected Rate (%)</Label>

                        <Input

                          name="expected\_rate"

                          type="number"

                          step="0.01"

                          value={formData.expected\_rate}

                          onChange={handleChange}

                          className="h-10 px-3 text-sm border border-gray-300 focus:ring-2 focus:ring-indigo-500 focus:border-indigo-500 transition-all duration-200 hover:border-gray-400"

                          placeholder="e.g., 7.5"

                          required

                        />

                      </div>

                    </div>

                    {formData.current\_rate && formData.expected\_rate && (

                      <div className="p-4 bg-blue-50 rounded-lg border border-blue-200">

                        <div className="flex items-center gap-2 text-sm font-medium text-blue-800 mb-2">

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

                          Rate Comparison

                        </div>

                        <p className="text-sm text-blue-700">

                          Requesting <span className="font-semibold text-lg">{(Number(formData.current\_rate) - Number(formData.expected\_rate)).toFixed(2)}%</span> reduction

                          <span className="ml-2 text-xs bg-blue-100 px-2 py-1 rounded-full">

                            {((Number(formData.current\_rate) - Number(formData.expected\_rate)) / Number(formData.current\_rate) \* 100).toFixed(1)}% decrease

                          </span>

                        </p>

                      </div>

                    )}

                  </div>

                  {/\* Justification \*/}

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

                    <div className="flex items-center gap-2 text-base font-semibold text-gray-900 border-b border-gray-200 pb-2">

                      <MessageSquare className="w-4 h-4 text-indigo-600" />

                      Justification

                    </div>

                    <div className="space-y-2">

                      <Label className="text-sm font-medium text-gray-700">Reason / Comment</Label>

                      <Textarea

                        name="comment"

                        rows={4}

                        value={formData.comment}

                        onChange={handleChange}

                        className="resize-none px-3 py-3 text-sm border border-gray-300 focus:ring-2 focus:ring-indigo-500 focus:border-indigo-500 transition-all duration-200 hover:border-gray-400"

                        placeholder="Please provide a detailed explanation for your request (minimum 10 characters)..."

                        required

                      />

                      <div className="text-right">

                        <span className={`text-xs ${formData.comment.length >= 10 ? 'text-green-600' : 'text-gray-500'}`}>

                          {formData.comment.length}/10 minimum characters

                        </span>

                      </div>

                    </div>

                  </div>

                  {/\* File Upload and Submit \*/}

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

                    <div className="space-y-3">

                      <div className="space-y-2">

                        <Label className="text-sm font-medium text-gray-700 flex items-center gap-2">

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

                          Supporting Document (Optional)

                        </Label>

                        <Input

                          type="file"

                          onChange={(e) => setFile(e.target.files?.[0] || null)}

                          className="h-11 p-2 text-sm file:mr-3 file:py-2 file:px-4 file:rounded-md file:border-0 file:text-xs file:font-medium file:bg-indigo-50 file:text-indigo-700 hover:file:bg-indigo-100 file:cursor-pointer cursor-pointer border border-gray-300 hover:border-gray-400 transition-colors"

                        />

                        {file && (

                          <div className="text-sm text-green-600 flex items-center gap-2 font-medium bg-green-50 p-2 rounded-lg border border-green-200">

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

                            {file.name} selected

                          </div>

                        )}

                      </div>

                      <Button

                        type="submit"

                        onClick={handleSubmit}

                        className={`w-full h-12 text-sm font-medium transition-all duration-300 transform hover:scale-[1.02] active:scale-[0.98] ${

                          verify

                            ? "bg-green-600 hover:bg-green-500 shadow-lg hover:shadow-green-200"

                            : "bg-gradient-to-r from-indigo-400 to-purple-400 hover:from-indigo-700 hover:to-purple-700 shadow-lg hover:shadow-indigo-200"

                        }`}

                      >

                        {verify ? (

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

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

                            Confirm & Submit Request

                          </span>

                        ) : (

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

                            <FileText className="w-5 h-5" />

                            Review & Submit Request

                          </span>

                        )}

                      </Button>

                    </div>

                  </div>

                </div>

                {/\* Error Display \*/}

                {error && (

                  <div className="mt-6 p-4 bg-red-50 border border-red-200 rounded-lg animate-in slide-in-from-top duration-200">

                    <div className="flex items-center gap-3 text-red-800 text-sm">

                      <XCircle className="w-5 h-5 flex-shrink-0" />

                      <span className="font-medium">{error}</span>

                    </div>

                  </div>

                )}

                {/\* Status Display \*/}

                {status && !error && (

                  <div className={`mt-6 p-4 border rounded-lg animate-in slide-in-from-top duration-200 ${

                    status.includes("success") || status.includes("submitted")

                      ? "bg-green-50 border-green-200 text-green-800"

                      : "bg-blue-50 border-blue-200 text-blue-800"

                  }`}>

                    <div className="flex items-center gap-3 text-sm">

                      {status.includes("Submitting") ? (

                        <div className="w-5 h-5 border-2 border-blue-600 border-t-transparent rounded-full animate-spin flex-shrink-0"></div>

                      ) : (

                        <CheckCircle className="w-5 h-5 flex-shrink-0" />

                      )}

                      <span className="font-medium">{status}</span>

                    </div>

                  </div>

                )}

              </CardContent>

            </Card>

          </div>

          {/\* Latest Submission - Now Below the Form \*/}

          {recentStatus && (

            <div className="w-full">

              <Card className="shadow-lg rounded-xl overflow-hidden border border-gray-200">

                <CardHeader className="bg-gray-50 border-b border-gray-200 p-4">

                  <CardTitle className="text-lg font-bold flex items-center gap-2 text-gray-900">

                    <Clock className="w-5 h-5 text-indigo-600" />

                    Latest Submission

                  </CardTitle>

                </CardHeader>

                <CardContent className="p-6 space-y-4 bg-white">

                  <div className="text-center">

                    <div className="flex justify-center mb-4">

                      <div className="p-3 rounded-full bg-gray-50">

                        {getStatusIcon(recentStatus.status)}

                      </div>

                    </div>

                    <div className={getStatusBadge(recentStatus.status)}>

                      {getStatusIcon(recentStatus.status)}

                      {recentStatus.status.charAt(0).toUpperCase() + recentStatus.status.slice(1)}

                    </div>

                  </div>

                  <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

                    <div className="p-4 bg-gray-50 rounded-lg border border-gray-100">

                      <div className="text-xs text-gray-600 mb-2 font-medium uppercase tracking-wide">Admin Comment</div>

                      <div className="font-medium text-sm text-gray-900">

                        {recentStatus.admin\_note || "No comment yet"}

                      </div>

                    </div>

                    <div className="p-4 bg-indigo-50 rounded-lg border border-indigo-100">

                      <div className="text-xs text-gray-600 mb-2 font-medium uppercase tracking-wide">Final Rate</div>

                      <div className="font-bold text-2xl text-indigo-600">

                        {recentStatus.final\_rate ? `${recentStatus.final\_rate}%` : "—"}

                      </div>

                    </div>

                  </div>

                  <div className="p-4 bg-blue-50 rounded-lg border border-blue-100">

                    <div className="text-xs text-gray-600 mb-3 font-medium uppercase tracking-wide">Document</div>

                    {recentStatus.file\_path ? (

                      <Button

                        size="sm"

                        className="w-full bg-blue-600 hover:bg-blue-700 text-white text-xs font-medium h-9 transition-colors"

                        onClick={() => window.open(`http://localhost:5000/uploads/${recentStatus.file\_path}`, '\_blank')}

                      >

                        <ExternalLink className="w-3 h-3 mr-2" />

                        View Document

                      </Button>

                    ) : (

                      <span className="text-gray-500 text-sm">No file uploaded</span>

                    )}

                  </div>

                  {recentStatus.status !== "pending" && (

                    <div className="pt-4 border-t border-gray-200">

                      <Button

                        className="w-full bg-green-600 hover:bg-green-700 text-white text-sm font-medium shadow-lg h-10 transition-all duration-200 transform hover:scale-[1.02]"

                        onClick={() => window.open(`http://localhost:5000/api/negotiation/receipt/${recentStatus.id}`, '\_blank')}

                      >

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

                        Download Receipt

                      </Button>

                    </div>

                  )}

                </CardContent>

              </Card>

            </div>

          )}

        </div>

      </div>

    </div>

  );

};

export default Negotiation;

**LOAN PROCESSING**

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

import {

  Upload,

  FileText,

  DollarSign,

  Calendar,

  Percent,

  Download,

  CheckCircle,

  AlertCircle,

  ArrowRight,

  Calculator,

  TrendingUp,

  Building,

  User,

  Settings

} from 'lucide-react';

const LoanProcessing = () => {

  const [currentStep, setCurrentStep] = useState(1);

  const [appId, setAppId] = useState('');

  const [loanData, setLoanData] = useState({

    loan\_amount: '',

    interest\_rate: '',

    tenure: ''

  });

  const [investmentDecision, setInvestmentDecision] = useState({

    wants\_to\_invest: false,

    investment\_amount: ''

  });

  const [uploadedFile, setUploadedFile] = useState(null);

  const [signedFile, setSignedFile] = useState(null);

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

  const [error, setError] = useState('');

  const [success, setSuccess] = useState('');

  const [reportId, setReportId] = useState('');

  const [calculations, setCalculations] = useState(null);

  const API\_BASE = 'http://localhost:5000';

  const gradientButton = "bg-gradient-to-r from-blue-400 to-purple-400 hover:from-blue-500 hover:to-purple-500 text-white font-semibold py-4 px-8 rounded-xl transition-all duration-300 transform hover:scale-105 shadow-lg hover:shadow-xl";

  const gradientIcon = "text-transparent bg-clip-text bg-gradient-to-r from-blue-400 to-purple-400";

  const handleManualEntry = async () => {

    if (!loanData.loan\_amount || !loanData.interest\_rate || !loanData.tenure) {

      setError('Please fill all loan details');

      return;

    }

    setLoading(true);

    setError('');

    try {

      const response = await fetch(`${API\_BASE}/loan/step1/manual-entry`, {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

        },

        body: JSON.stringify({

          loan\_amount: parseFloat(loanData.loan\_amount),

          interest\_rate: parseFloat(loanData.interest\_rate),

          tenure: parseInt(loanData.tenure)

        }),

      });

      const data = await response.json();

      if (response.ok) {

        setAppId(data.app\_id);

        setSuccess('Loan details saved successfully!');

        setCurrentStep(2);

      } else {

        setError(data.detail || 'Failed to save loan details');

      }

    } catch (err) {

      setError('Network error. Please try again.');

    } finally {

      setLoading(false);

    }

  };

  const handlePdfUpload = async () => {

    if (!uploadedFile) {

      setError('Please select a PDF file');

      return;

    }

    setLoading(true);

    setError('');

    const formData = new FormData();

    formData.append('file', uploadedFile);

    try {

      const response = await fetch(`${API\_BASE}/loan/step1/upload-pdf`, {

        method: 'POST',

        body: formData,

      });

      const data = await response.json();

      if (response.ok) {

        setAppId(data.app\_id);

        setLoanData({

          loan\_amount: data.extracted\_data.loan\_amount.toString(),

          interest\_rate: data.extracted\_data.interest\_rate.toString(),

          tenure: data.extracted\_data.tenure.toString()

        });

        setSuccess('PDF processed successfully!');

        setCurrentStep(2);

      } else {

        setError(data.detail || 'Failed to process PDF');

      }

    } catch (err) {

      setError('Network error. Please try again.');

    } finally {

      setLoading(false);

    }

  };

  const generateAgreement = async () => {

    setLoading(true);

    setError('');

    try {

      const response = await fetch(`${API\_BASE}/loan/step2/generate-agreement/${appId}`, {

        method: 'POST',

      });

      const data = await response.json();

      if (response.ok) {

        setCalculations(data.loan\_calculations);

        setSuccess('Agreement generated successfully!');

        setCurrentStep(3);

      } else {

        setError(data.detail || 'Failed to generate agreement');

      }

    } catch (err) {

      setError('Network error. Please try again.');

    } finally {

      setLoading(false);

    }

  };

  const downloadAgreement = () => {

    window.open(`${API\_BASE}/loan/step2/download-agreement/${appId}`, '\_blank');

  };

  const uploadSignedAgreement = async () => {

    if (!signedFile) {

      setError('Please select the signed agreement file');

      return;

    }

    setLoading(true);

    setError('');

    const formData = new FormData();

    formData.append('file', signedFile);

    try {

      const response = await fetch(`${API\_BASE}/loan/step3/upload-signed/${appId}`, {

        method: 'POST',

        body: formData,

      });

      const data = await response.json();

      if (response.ok) {

        setSuccess('Signed agreement uploaded successfully!');

        setCurrentStep(4);

      } else {

        setError(data.detail || 'Failed to upload signed agreement');

      }

    } catch (err) {

      setError('Network error. Please try again.');

    } finally {

      setLoading(false);

    }

  };

  const submitInvestmentDecision = async () => {

    if (investmentDecision.wants\_to\_invest && !investmentDecision.investment\_amount) {

      setError('Please enter investment amount');

      return;

    }

    setLoading(true);

    setError('');

    try {

      const response = await fetch(`${API\_BASE}/loan/step4/investment-decision/${appId}`, {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

        },

        body: JSON.stringify({

          wants\_to\_invest: investmentDecision.wants\_to\_invest,

          investment\_amount: investmentDecision.wants\_to\_invest ? parseFloat(investmentDecision.investment\_amount) : null

        }),

      });

      const data = await response.json();

      if (response.ok) {

        setReportId(data.report\_id);

        setSuccess('Process completed successfully!');

        setCurrentStep(5);

      } else {

        setError(data.detail || 'Failed to process investment decision');

      }

    } catch (err) {

      setError('Network error. Please try again.');

    } finally {

      setLoading(false);

    }

  };

  const downloadFinalReport = () => {

    window.open(`${API\_BASE}/loan/step4/download-final-report/${reportId}`, '\_blank');

  };

  const resetProcess = () => {

    setCurrentStep(1);

    setAppId('');

    setLoanData({ loan\_amount: '', interest\_rate: '', tenure: '' });

    setInvestmentDecision({ wants\_to\_invest: false, investment\_amount: '' });

    setUploadedFile(null);

    setSignedFile(null);

    setError('');

    setSuccess('');

    setReportId('');

    setCalculations(null);

  };

  const StepIndicator = ({ step, title, completed, active }) => (

    <div className={`flex items-center space-x-4 p-6 rounded-xl border-2 transition-all duration-300 transform hover:scale-105 ${

      active ? 'border-blue-400 bg-blue-50 shadow-lg' : completed ? 'border-green-400 bg-green-50 shadow-md' : 'border-gray-200 bg-gray-50'

    }`}>

      <div className={`w-12 h-12 rounded-full flex items-center justify-center transition-all duration-300 ${

        completed ? 'bg-green-400 text-white shadow-lg' : active ? 'bg-gradient-to-r from-blue-400 to-purple-400 text-white shadow-lg' : 'bg-gray-300 text-gray-600'

      }`}>

        {completed ? <CheckCircle size={24} /> : <span className="text-lg font-bold">{step}</span>}

      </div>

      <span className={`font-semibold text-lg ${active ? gradientIcon : completed ? 'text-green-600' : 'text-gray-600'}`}>

        {title}

      </span>

    </div>

  );

  return (

    <div className="min-h-screen w-full bg-gradient-to-br from-gray-50 via-white to-gray-50 flex flex-col">

      {/\* Main Content Area \*/}

      <div className="flex-1 flex flex-col items-center justify-start py-6 px-4">

        <div className="w-full max-w-6xl">

          {/\* Header with Icon and Tagline \*/}

          <div className="text-center mb-8">

            <Settings className="w-12 h-12 mx-auto mb-3 text-blue-500" />

            <p className="text-gray-600 text-lg font-medium">Complete your loan application in 4 simple steps</p>

          </div>

          {/\* Step Indicators \*/}

          <div className="grid grid-cols-1 lg:grid-cols-4 gap-6 mb-10">

            <StepIndicator step={1} title="Loan Details" completed={currentStep > 1} active={currentStep === 1} />

            <StepIndicator step={2} title="Agreement" completed={currentStep > 2} active={currentStep === 2} />

            <StepIndicator step={3} title="Upload Signed" completed={currentStep > 3} active={currentStep === 3} />

            <StepIndicator step={4} title="Investment" completed={currentStep > 4} active={currentStep === 4} />

          </div>

          {/\* Error/Success Messages \*/}

          {error && (

            <div className="bg-red-50 border-2 border-red-200 text-red-700 px-6 py-4 rounded-xl mb-8 flex items-center space-x-3 shadow-lg max-w-4xl mx-auto">

              <AlertCircle size={24} />

              <span className="text-lg">{error}</span>

            </div>

          )}

          {success && (

            <div className="bg-green-50 border-2 border-green-200 text-green-700 px-6 py-4 rounded-xl mb-8 flex items-center space-x-3 shadow-lg max-w-4xl mx-auto">

              <CheckCircle size={24} />

              <span className="text-lg">{success}</span>

            </div>

          )}

          {/\* Step Content \*/}

          <div className="bg-white rounded-2xl shadow-xl border border-gray-200 max-w-5xl mx-auto">

            <div className="p-10 lg:p-12">

              {currentStep === 1 && (

                <div>

                  <h2 className={`text-3xl font-bold mb-8 text-center ${gradientIcon}`}>

                    Step 1: Enter Loan Details

                  </h2>

                  {/\* PDF Upload Option \*/}

                  <div className="mb-10 p-8 border-2 border-dashed border-gray-300 rounded-xl hover:border-blue-300 transition-colors duration-300">

                    <h3 className="text-xl font-semibold mb-6 flex items-center justify-center space-x-3">

                      <Upload className="text-blue-400" size={24} />

                      <span>Option 1: Upload PDF</span>

                    </h3>

                    <div className="max-w-2xl mx-auto">

                      <div className="flex flex-col lg:flex-row items-center space-y-4 lg:space-y-0 lg:space-x-4">

                        <input

                          type="file"

                          accept=".pdf"

                          onChange={(e) => setUploadedFile(e.target.files[0])}

                          className="flex-1 w-full border-2 border-gray-300 rounded-lg p-4 text-lg focus:border-blue-400 transition-colors"

                        />

                        <button

                          onClick={handlePdfUpload}

                          disabled={loading || !uploadedFile}

                          className={`${gradientButton} disabled:opacity-50 disabled:cursor-not-allowed whitespace-nowrap`}

                        >

                          {loading ? 'Processing...' : 'Upload PDF'}

                        </button>

                      </div>

                    </div>

                  </div>

                  <div className="text-center text-gray-500 text-xl mb-10 font-medium">OR</div>

                  {/\* Manual Entry Option \*/}

                  <div>

                    <h3 className="text-xl font-semibold mb-6 flex items-center justify-center space-x-3">

                      <User className="text-purple-400" size={24} />

                      <span>Option 2: Manual Entry</span>

                    </h3>

                    <div className="max-w-4xl mx-auto">

                      <div className="grid grid-cols-1 lg:grid-cols-3 gap-8 mb-8">

                        <div>

                          <label className="block text-lg font-semibold text-gray-700 mb-3 flex items-center space-x-2">

                            <DollarSign className="text-blue-400" size={20} />

                            <span>Loan Amount (₹)</span>

                          </label>

                          <input

                            type="number"

                            value={loanData.loan\_amount}

                            onChange={(e) => setLoanData({...loanData, loan\_amount: e.target.value})}

                            className="w-full border-2 border-gray-300 rounded-lg p-4 text-lg focus:ring-2 focus:ring-blue-400 focus:border-transparent transition-all"

                            placeholder="Enter loan amount"

                          />

                        </div>

                        <div>

                          <label className="block text-lg font-semibold text-gray-700 mb-3 flex items-center space-x-2">

                            <Percent className="text-purple-400" size={20} />

                            <span>Interest Rate (%)</span>

                          </label>

                          <input

                            type="number"

                            step="0.01"

                            value={loanData.interest\_rate}

                            onChange={(e) => setLoanData({...loanData, interest\_rate: e.target.value})}

                            className="w-full border-2 border-gray-300 rounded-lg p-4 text-lg focus:ring-2 focus:ring-purple-400 focus:border-transparent transition-all"

                            placeholder="Enter interest rate"

                          />

                        </div>

                        <div>

                          <label className="block text-lg font-semibold text-gray-700 mb-3 flex items-center space-x-2">

                            <Calendar className="text-blue-400" size={20} />

                            <span>Tenure (months)</span>

                          </label>

                          <input

                            type="number"

                            value={loanData.tenure}

                            onChange={(e) => setLoanData({...loanData, tenure: e.target.value})}

                            className="w-full border-2 border-gray-300 rounded-lg p-4 text-lg focus:ring-2 focus:ring-blue-400 focus:border-transparent transition-all"

                            placeholder="Enter tenure"

                          />

                        </div>

                      </div>

                      <div className="text-center">

                        <button

                          onClick={handleManualEntry}

                          disabled={loading}

                          className={`${gradientButton} disabled:opacity-50 disabled:cursor-not-allowed flex items-center justify-center space-x-3 mx-auto`}

                        >

                          <span className="text-lg">{loading ? 'Saving...' : 'Save Loan Details'}</span>

                          <ArrowRight size={24} />

                        </button>

                      </div>

                    </div>

                  </div>

                </div>

              )}

              {currentStep === 2 && (

                <div>

                  <h2 className={`text-3xl font-bold mb-8 text-center ${gradientIcon}`}>

                    Step 2: Generate Agreement Document

                  </h2>

                  {/\* Loan Summary \*/}

                  <div className="bg-gradient-to-r from-blue-50 to-purple-50 p-8 rounded-xl mb-8 max-w-4xl mx-auto">

                    <h3 className="text-xl font-semibold mb-6 text-center">Loan Summary</h3>

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

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

                        <DollarSign className="w-12 h-12 text-blue-400 mx-auto mb-3" />

                        <p className="text-3xl font-bold text-gray-800 mb-2">₹{parseFloat(loanData.loan\_amount).toLocaleString()}</p>

                        <p className="text-gray-600 text-lg">Loan Amount</p>

                      </div>

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

                        <Percent className="w-12 h-12 text-purple-400 mx-auto mb-3" />

                        <p className="text-3xl font-bold text-gray-800 mb-2">{loanData.interest\_rate}%</p>

                        <p className="text-gray-600 text-lg">Interest Rate</p>

                      </div>

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

                        <Calendar className="w-12 h-12 text-blue-400 mx-auto mb-3" />

                        <p className="text-3xl font-bold text-gray-800 mb-2">{loanData.tenure}</p>

                        <p className="text-gray-600 text-lg">Months</p>

                      </div>

                    </div>

                  </div>

                  {calculations && (

                    <div className="bg-green-50 p-8 rounded-xl mb-8 max-w-4xl mx-auto">

                      <h3 className="text-xl font-semibold mb-6 flex items-center justify-center space-x-3">

                        <Calculator className="text-green-600" size={24} />

                        <span>Loan Calculations</span>

                      </h3>

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

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

                          <p className="text-2xl font-bold text-gray-800 mb-2">₹{calculations.emi.toLocaleString()}</p>

                          <p className="text-gray-600 text-lg">Monthly EMI</p>

                        </div>

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

                          <p className="text-2xl font-bold text-gray-800 mb-2">₹{calculations.total\_interest.toLocaleString()}</p>

                          <p className="text-gray-600 text-lg">Total Interest</p>

                        </div>

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

                          <p className="text-2xl font-bold text-gray-800 mb-2">₹{calculations.total\_amount.toLocaleString()}</p>

                          <p className="text-gray-600 text-lg">Total Amount</p>

                        </div>

                      </div>

                    </div>

                  )}

                  <div className="flex flex-col lg:flex-row justify-center items-center space-y-4 lg:space-y-0 lg:space-x-6 max-w-2xl mx-auto">

                    <button

                      onClick={generateAgreement}

                      disabled={loading}

                      className={`${gradientButton} disabled:opacity-50 disabled:cursor-not-allowed flex items-center justify-center space-x-3 w-full lg:w-auto`}

                    >

                      <FileText size={24} />

                      <span className="text-lg">{loading ? 'Generating...' : 'Generate Agreement'}</span>

                    </button>

                    {calculations && (

                      <button

                        onClick={downloadAgreement}

                        className="bg-gray-600 hover:bg-gray-700 text-white font-semibold py-4 px-8 rounded-xl transition-all duration-300 flex items-center space-x-3 w-full lg:w-auto shadow-lg hover:shadow-xl transform hover:scale-105"

                      >

                        <Download size={24} />

                        <span className="text-lg">Download</span>

                      </button>

                    )}

                  </div>

                </div>

              )}

              {currentStep === 3 && (

                <div>

                  <h2 className={`text-3xl font-bold mb-8 text-center ${gradientIcon}`}>

                    Step 3: Upload Signed Agreement

                  </h2>

                  <div className="text-center mb-10 max-w-2xl mx-auto">

                    <FileText className="w-20 h-20 text-gray-400 mx-auto mb-6" />

                    <p className="text-gray-600 mb-6 text-lg">

                      Please download, sign, and upload the loan agreement document.

                    </p>

                    <button

                      onClick={downloadAgreement}

                      className="bg-gray-600 hover:bg-gray-700 text-white font-semibold py-3 px-6 rounded-xl transition-all duration-300 inline-flex items-center space-x-3 mb-8 shadow-lg hover:shadow-xl transform hover:scale-105"

                    >

                      <Download size={24} />

                      <span className="text-lg">Download Agreement</span>

                    </button>

                  </div>

                  <div className="border-2 border-dashed border-gray-300 rounded-xl p-8 max-w-3xl mx-auto hover:border-blue-300 transition-colors duration-300">

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

                      <input

                        type="file"

                        accept=".pdf"

                        onChange={(e) => setSignedFile(e.target.files[0])}

                        className="w-full border-2 border-gray-300 rounded-lg p-4 text-lg focus:border-blue-400 transition-colors"

                      />

                      <div className="text-center">

                        <button

                          onClick={uploadSignedAgreement}

                          disabled={loading || !signedFile}

                          className={`${gradientButton} disabled:opacity-50 disabled:cursor-not-allowed flex items-center justify-center space-x-3 mx-auto`}

                        >

                          <Upload size={24} />

                          <span className="text-lg">{loading ? 'Uploading...' : 'Upload Signed Agreement'}</span>

                        </button>

                      </div>

                    </div>

                  </div>

                </div>

              )}

              {currentStep === 4 && (

                <div>

                  <h2 className={`text-3xl font-bold mb-8 text-center ${gradientIcon}`}>

                    Step 4: Investment Decision

                  </h2>

                  <div className="bg-gradient-to-r from-blue-50 to-purple-50 p-8 rounded-xl mb-8 max-w-3xl mx-auto text-center">

                    <TrendingUp className="w-16 h-16 text-blue-400 mx-auto mb-6" />

                    <h3 className="text-xl font-semibold mb-4">Investment Opportunity</h3>

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

                      Would you like to invest a portion of your loan amount to potentially earn returns?

                    </p>

                  </div>

                  <div className="space-y-8 max-w-2xl mx-auto">

                    <div className="flex items-center justify-center space-x-4 p-6 bg-gray-50 rounded-xl">

                      <input

                        type="checkbox"

                        id="invest"

                        checked={investmentDecision.wants\_to\_invest}

                        onChange={(e) => setInvestmentDecision({

                          ...investmentDecision,

                          wants\_to\_invest: e.target.checked,

                          investment\_amount: e.target.checked ? investmentDecision.investment\_amount : ''

                        })}

                        className="w-6 h-6 text-blue-400 focus:ring-blue-400 border-gray-300 rounded transition-all"

                      />

                      <label htmlFor="invest" className="text-xl font-semibold text-gray-700 cursor-pointer">

                        Yes, I want to invest part of my loan amount

                      </label>

                    </div>

                    {investmentDecision.wants\_to\_invest && (

                      <div className="p-6 bg-green-50 rounded-xl">

                        <label className="block text-lg font-semibold text-gray-700 mb-4 flex items-center justify-center space-x-3">

                          <DollarSign className="text-green-400" size={20} />

                          <span>Investment Amount (₹)</span>

                        </label>

                        <input

                          type="number"

                          value={investmentDecision.investment\_amount}

                          onChange={(e) => setInvestmentDecision({...investmentDecision, investment\_amount: e.target.value})}

                          className="w-full border-2 border-gray-300 rounded-lg p-4 text-lg focus:ring-2 focus:ring-green-400 focus:border-transparent transition-all"

                          placeholder="Enter investment amount"

                          max={loanData.loan\_amount}

                        />

                        <p className="text-gray-500 mt-3 text-center text-lg">

                          Maximum: ₹{parseFloat(loanData.loan\_amount).toLocaleString()}

                        </p>

                      </div>

                    )}

                    <div className="text-center">

                      <button

                        onClick={submitInvestmentDecision}

                        disabled={loading}

                        className={`${gradientButton} disabled:opacity-50 disabled:cursor-not-allowed flex items-center justify-center space-x-3 mx-auto`}

                      >

                        <span className="text-lg">{loading ? 'Processing...' : 'Complete Application'}</span>

                        <ArrowRight size={24} />

                      </button>

                    </div>

                  </div>

                </div>

              )}

              {currentStep === 5 && (

                <div className="text-center max-w-3xl mx-auto">

                  <CheckCircle className="w-24 h-24 text-green-400 mx-auto mb-8" />

                  <h2 className={`text-4xl font-bold mb-6 ${gradientIcon}`}>

                    Application Complete!

                  </h2>

                  <p className="text-gray-600 mb-10 text-xl">

                    Your loan application has been processed successfully.

                    Download your final report below.

                  </p>

                  <div className="flex flex-col lg:flex-row justify-center items-center space-y-4 lg:space-y-0 lg:space-x-6 mb-10">

                    <button

                      onClick={downloadFinalReport}

                      className={`${gradientButton} flex items-center space-x-3`}

                    >

                      <Download size={24} />

                      <span className="text-lg">Download Final Report</span>

                    </button>

                    <button

                      onClick={resetProcess}

                      className="bg-gray-600 hover:bg-gray-700 text-white font-semibold py-4 px-8 rounded-xl transition-all duration-300 flex items-center space-x-3 shadow-lg hover:shadow-xl transform hover:scale-105"

                    >

                      <span className="text-lg">Start New Application</span>

                    </button>

                  </div>

                  <div className="bg-green-50 p-8 rounded-xl border-2 border-green-200">

                    <h3 className="text-xl font-semibold mb-4">Application Details</h3>

                    <div className="space-y-2 text-lg">

                      <p className="text-gray-600">Application ID: <span className="font-semibold text-gray-800">{appId}</span></p>

                      <p className="text-gray-600">Report ID: <span className="font-semibold text-gray-800">{reportId}</span></p>

                    </div>

                  </div>

                </div>

              )}

            </div>

          </div>

        </div>

      </div>

    </div>

  );

};

export default LoanProcessing;

**EMERGENCY LOAN**

import React, { useState, useRef } from 'react';

import { Upload, FileText, CheckCircle, XCircle, Clock, AlertCircle, DollarSign, Shield, Zap } from 'lucide-react';

const EmergencyLoan = () => {

  const [formData, setFormData] = useState({

    amount: '',

    reason: '',

    userId: 'user123'

  });

  const [selectedFile, setSelectedFile] = useState(null);

  const [requestId, setRequestId] = useState('');

  const [status, setStatus] = useState('idle');

  const [loanStatus, setLoanStatus] = useState(null);

  const [error, setError] = useState('');

  const fileInputRef = useRef(null);

  const handleInputChange = (e) => {

    const { name, value } = e.target;

    setFormData(prev => ({

      ...prev,

      [name]: value

    }));

  };

  const handleFileSelect = (e) => {

    const file = e.target.files[0];

    if (file) {

      const allowedTypes = ['application/pdf', 'image/jpeg', 'image/png', 'image/jpg'];

      if (!allowedTypes.includes(file.type)) {

        setError('Please upload PDF, JPEG, or PNG files only');

        return;

      }

      if (file.size > 5 \* 1024 \* 1024) {

        setError('File size must be less than 5MB');

        return;

      }

      setSelectedFile(file);

      setError('');

    }

  };

  const submitLoanRequest = async () => {

    if (!formData.amount || !formData.reason || !selectedFile) {

      setError('Please fill all fields and upload a document');

      return;

    }

    if (parseFloat(formData.amount) <= 0) {

      setError('Please enter a valid amount');

      return;

    }

    setStatus('submitting');

    setError('');

    const formDataToSend = new FormData();

    formDataToSend.append('amount', formData.amount);

    formDataToSend.append('reason', formData.reason);

    formDataToSend.append('user\_id', formData.userId);

    formDataToSend.append('document', selectedFile);

    try {

      const response = await fetch('http://localhost:5000/emergency-loan/submit', {

        method: 'POST',

        body: formDataToSend

      });

      const result = await response.json();

      if (response.ok) {

        setRequestId(result.request\_id);

        setStatus('submitted');

      } else {

        setError(result.detail || 'Failed to submit request');

        setStatus('idle');

      }

    } catch (err) {

      setError('Network error. Please try again.');

      setStatus('idle');

    }

  };

  const checkStatus = async () => {

    if (!requestId) return;

    setStatus('checking');

    try {

      const response = await fetch(`http://localhost:5000/emergency-loan/status/${requestId}`);

      const result = await response.json();

      if (response.ok) {

        setLoanStatus(result);

      } else {

        setError(result.detail || 'Failed to fetch status');

      }

    } catch (err) {

      setError('Network error. Please try again.');

    }

    setStatus('submitted');

  };

  const resetForm = () => {

    setFormData({ amount: '', reason: '', userId: 'user123' });

    setSelectedFile(null);

    setRequestId('');

    setStatus('idle');

    setLoanStatus(null);

    setError('');

    if (fileInputRef.current) {

      fileInputRef.current.value = '';

    }

  };

  const renderStatusIcon = () => {

    if (!loanStatus) return <Clock className="w-6 h-6 text-amber-500" />;

    switch (loanStatus.status) {

      case 'pending':

        return <Clock className="w-6 h-6 text-amber-500" />;

      case 'approved':

        return <CheckCircle className="w-6 h-6 text-emerald-500" />;

      case 'rejected':

        return <XCircle className="w-6 h-6 text-rose-500" />;

      default:

        return <Clock className="w-6 h-6 text-amber-500" />;

    }

  };

  return (

    <div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-indigo-100 p-4">

      <div className="w-full max-w-7xl mx-auto">

        {/\* Header \*/}

        <div className="text-center mb-8 pt-6">

          <div className="inline-flex items-center justify-center w-16 h-16 bg-gradient-to-r from-blue-600 to-indigo-600 rounded-full mb-4 shadow-lg">

            <DollarSign className="w-8 h-8 text-white" />

          </div>

          <p className="text-lg text-gray-600 max-w-3xl mx-auto">

            Quick financial assistance when you need it most. Fast approval, secure process.

          </p>

          {/\* Feature highlights \*/}

          <div className="flex justify-center flex-wrap gap-6 mt-6">

            <div className="flex items-center space-x-2 text-gray-600">

              <Zap className="w-5 h-5 text-blue-500" />

              <span className="font-medium">Fast Processing</span>

            </div>

            <div className="flex items-center space-x-2 text-gray-600">

              <Shield className="w-5 h-5 text-green-500" />

              <span className="font-medium">Secure & Safe</span>

            </div>

            <div className="flex items-center space-x-2 text-gray-600">

              <CheckCircle className="w-5 h-5 text-purple-500" />

              <span className="font-medium">24/7 Support</span>

            </div>

          </div>

        </div>

        {status === 'idle' && (

          <div className="bg-white/80 backdrop-blur-sm border border-white/20 rounded-3xl p-10 shadow-2xl">

            <div className="space-y-8">

              {/\* Loan Amount \*/}

              <div className="space-y-3">

                <label className="flex items-center space-x-2 text-gray-800 text-lg font-semibold">

                  <DollarSign className="w-5 h-5 text-blue-600" />

                  <span>Loan Amount (₹)</span>

                </label>

                <input

                  type="number"

                  name="amount"

                  value={formData.amount}

                  onChange={handleInputChange}

                  placeholder="Enter amount needed"

                  className="w-full px-6 py-4 text-lg rounded-2xl border-2 border-gray-200 focus:border-blue-500 focus:ring-4 focus:ring-blue-100 text-gray-900 transition-all duration-200 bg-white/50"

                />

              </div>

              {/\* Reason \*/}

              <div className="space-y-3">

                <label className="flex items-center space-x-2 text-gray-800 text-lg font-semibold">

                  <FileText className="w-5 h-5 text-blue-600" />

                  <span>Reason for Emergency Loan</span>

                </label>

                <textarea

                  name="reason"

                  value={formData.reason}

                  onChange={handleInputChange}

                  placeholder="Please explain your emergency situation in detail..."

                  rows={5}

                  className="w-full px-6 py-4 text-lg rounded-2xl border-2 border-gray-200 text-gray-900 focus:border-blue-500 focus:ring-4 focus:ring-blue-100 resize-none transition-all duration-200 bg-white/50"

                />

              </div>

              {/\* File Upload \*/}

              <div className="space-y-3">

                <label className="flex items-center space-x-2 text-gray-800 text-lg font-semibold">

                  <Upload className="w-5 h-5 text-blue-600" />

                  <span>Supporting Documents</span>

                </label>

                <div className="relative">

                  <input

                    type="file"

                    ref={fileInputRef}

                    onChange={handleFileSelect}

                    accept=".pdf,.jpg,.jpeg,.png"

                    className="hidden"

                  />

                  <div

                    onClick={() => fileInputRef.current?.click()}

                    className="w-full px-6 py-8 rounded-2xl border-2 border-dashed border-gray-300 hover:border-blue-400 text-gray-600 cursor-pointer transition-all duration-200 text-center bg-gradient-to-br from-gray-50 to-blue-50/30 hover:from-blue-50 hover:to-indigo-50"

                  >

                    {selectedFile ? (

                      <div className="space-y-3">

                        <div className="inline-flex items-center justify-center w-16 h-16 bg-green-100 rounded-full">

                          <FileText className="w-8 h-8 text-green-600" />

                        </div>

                        <div>

                          <p className="text-lg font-semibold text-gray-800">{selectedFile.name}</p>

                          <p className="text-sm text-gray-500">File uploaded successfully</p>

                        </div>

                      </div>

                    ) : (

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

                        <div className="inline-flex items-center justify-center w-16 h-16 bg-blue-100 rounded-full">

                          <Upload className="w-8 h-8 text-blue-600" />

                        </div>

                        <div>

                          <p className="text-lg font-semibold text-gray-800">Click to upload documents</p>

                          <p className="text-sm text-gray-500 mt-1">PDF, JPEG, or PNG files • Max size: 5MB</p>

                        </div>

                      </div>

                    )}

                  </div>

                </div>

              </div>

              {error && (

                <div className="flex items-center space-x-3 text-rose-700 bg-rose-50 border border-rose-200 p-4 rounded-2xl">

                  <AlertCircle className="w-6 h-6 flex-shrink-0" />

                  <span className="font-medium">{error}</span>

                </div>

              )}

              <button

                type="button"

                onClick={submitLoanRequest}

                disabled={status === 'submitting'}

                className="w-full bg-gradient-to-r from-blue-600 to-indigo-600 hover:from-blue-700 hover:to-indigo-700 text-white py-5 px-8 rounded-2xl text-lg font-semibold disabled:opacity-50 disabled:cursor-not-allowed transition-all duration-200 transform hover:scale-[1.02] shadow-xl"

              >

                {status === 'submitting' ? (

                  <div className="flex items-center justify-center space-x-3">

                    <div className="animate-spin w-5 h-5 border-2 border-white/30 border-t-white rounded-full"></div>

                    <span>Processing Request...</span>

                  </div>

                ) : (

                  <div className="flex items-center justify-center space-x-3">

                    <Zap className="w-5 h-5" />

                    <span>Submit Emergency Loan Request</span>

                  </div>

                )}

              </button>

            </div>

          </div>

        )}

        {status === 'submitting' && (

          <div className="bg-white/90 backdrop-blur-sm border border-white/20 rounded-3xl p-12 text-center shadow-2xl">

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

              <div className="relative">

                <div className="animate-spin w-20 h-20 border-4 border-blue-200 border-t-blue-600 rounded-full mx-auto"></div>

                <div className="absolute inset-0 flex items-center justify-center">

                  <div className="w-10 h-10 bg-blue-600 rounded-full flex items-center justify-center">

                    <Zap className="w-5 h-5 text-white" />

                  </div>

                </div>

              </div>

              <h2 className="text-3xl font-bold text-gray-900">Processing Your Request</h2>

              <p className="text-lg text-gray-600 max-w-md mx-auto">

                Please wait while we securely process your emergency loan application...

              </p>

            </div>

          </div>

        )}

        {status === 'submitted' && (

          <div className="bg-white/90 backdrop-blur-sm border border-white/20 rounded-3xl p-10 shadow-2xl">

            <div className="text-center mb-8">

              <div className="inline-flex items-center justify-center w-20 h-20 bg-emerald-100 rounded-full mb-6">

                <CheckCircle className="w-10 h-10 text-emerald-600" />

              </div>

              <h2 className="text-3xl font-bold text-gray-900 mb-3">Request Submitted Successfully!</h2>

              <div className="inline-flex items-center space-x-2 bg-gray-100 px-4 py-2 rounded-full">

                <span className="text-sm text-gray-600">Request ID:</span>

                <span className="font-mono text-sm font-semibold text-gray-900">{requestId}</span>

              </div>

            </div>

            {loanStatus && (

              <div className="bg-gradient-to-br from-gray-50 to-blue-50/50 rounded-2xl p-8 mb-8 border border-gray-200">

                <div className="flex items-center space-x-4 mb-6">

                  <div className="flex-shrink-0">

                    {renderStatusIcon()}

                  </div>

                  <div>

                    <h3 className="text-2xl font-bold text-gray-800 capitalize">

                      Status: {loanStatus.status}

                    </h3>

                  </div>

                </div>

                {loanStatus.status === 'approved' && (

                  <div className="bg-emerald-50 border border-emerald-200 rounded-xl p-6 space-y-4">

                    <h4 className="text-lg font-semibold text-emerald-800 mb-3">🎉 Congratulations! Your loan has been approved</h4>

                    <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

                      <div className="flex items-center justify-between bg-white p-4 rounded-lg">

                        <span className="text-gray-600 font-medium">Approved Amount:</span>

                        <span className="text-xl font-bold text-emerald-600">₹{loanStatus.approved\_amount?.toLocaleString()}</span>

                      </div>

                      <div className="flex items-center justify-between bg-white p-4 rounded-lg">

                        <span className="text-gray-600 font-medium">Interest Rate:</span>

                        <span className="text-xl font-bold text-emerald-600">{loanStatus.interest\_rate}% p.a.</span>

                      </div>

                    </div>

                  </div>

                )}

                {loanStatus.status === 'rejected' && (

                  <div className="bg-rose-50 border border-rose-200 rounded-xl p-6">

                    <h4 className="text-lg font-semibold text-rose-800 mb-3">Application Status</h4>

                    <div className="bg-white p-4 rounded-lg">

                      <span className="text-gray-600 font-medium">Reason: </span>

                      <span className="text-rose-600 font-semibold">{loanStatus.rejection\_reason}</span>

                    </div>

                  </div>

                )}

                {loanStatus.status === 'pending' && (

                  <div className="bg-amber-50 border border-amber-200 rounded-xl p-6">

                    <h4 className="text-lg font-semibold text-amber-800 mb-3">Under Review</h4>

                    <p className="text-amber-700 bg-white p-4 rounded-lg">

                      Your request is currently being reviewed by our team. Please check back later for updates.

                    </p>

                  </div>

                )}

              </div>

            )}

            <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

              <button

                onClick={checkStatus}

                disabled={status === 'checking'}

                className="bg-gradient-to-r from-blue-600 to-indigo-600 hover:from-blue-700 hover:to-indigo-700 text-white py-4 px-6 rounded-2xl font-semibold disabled:opacity-50 transition-all duration-200 transform hover:scale-[1.02] shadow-lg"

              >

                {status === 'checking' ? (

                  <div className="flex items-center justify-center space-x-2">

                    <div className="animate-spin w-4 h-4 border-2 border-white/30 border-t-white rounded-full"></div>

                    <span>Checking Status...</span>

                  </div>

                ) : (

                  <div className="flex items-center justify-center space-x-2">

                    <Clock className="w-5 h-5" />

                    <span>Check Status</span>

                  </div>

                )}

              </button>

              <button

                onClick={resetForm}

                className="bg-white hover:bg-gray-50 text-gray-700 border-2 border-gray-300 py-4 px-6 rounded-2xl font-semibold transition-all duration-200 transform hover:scale-[1.02] shadow-lg"

              >

                <div className="flex items-center justify-center space-x-2">

                  <Upload className="w-5 h-5" />

                  <span>New Request</span>

                </div>

              </button>

            </div>

          </div>

        )}

      </div>

    </div>

  );

};

export default EmergencyLoan;

**FEEDBACK**

import React, { useState } from 'react';

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

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

import { MessageCircle, Star, ThumbsUp, ThumbsDown, Send, Lightbulb, Bug, Heart, Zap, Target } from 'lucide-react';

import axios from 'axios';

const Feedback = () => {

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

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

  const [rating, setRating] = useState(0);

  const [category, setCategory] = useState('');

  const [priority, setPriority] = useState('medium');

  const [contactBack, setContactBack] = useState(false);

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

  const [activeTab, setActiveTab] = useState('feedback');

  const suggestions = [

    "User interface is intuitive",

    "Need more investment options",

    "App is slow on mobile",

    "Notifications are too frequent",

    "Excellent customer support",

    "Love the new dashboard design",

    "Would like dark mode option",

    "Charts could be more interactive",

    "Great mobile responsiveness",

    "Need better filtering options"

  ];

  const categories = [

    { id: 'ui', label: 'User Interface', icon: Target, color: 'bg-blue-500' },

    { id: 'performance', label: 'Performance', icon: Zap, color: 'bg-yellow-500' },

    { id: 'feature', label: 'Feature Request', icon: Lightbulb, color: 'bg-green-500' },

    { id: 'bug', label: 'Bug Report', icon: Bug, color: 'bg-red-500' },

    { id: 'general', label: 'General', icon: MessageCircle, color: 'bg-purple-500' }

  ];

  const validateFeedback = (text: string) => {

    if (!text.trim()) {

      setMessage('Please enter your feedback or select a suggestion.');

      return false;

    }

    if (text.length > 500) {

      setMessage('Feedback cannot exceed 500 characters.');

      return false;

    }

    if (contactBack && !email.trim()) {

      setMessage('Please provide your email address for follow-up.');

      return false;

    }

    if (contactBack && !email.includes('@')) {

      setMessage('Please provide a valid email address.');

      return false;

    }

    setMessage('');

    return true;

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    console.log("Submitting feedback...");

    if (!validateFeedback(content)) return;

    try {

      await axios.post('http://localhost:5000/feedback/submit', { feedback: content });

      setMessage('✅ Feedback submitted successfully! Thank you for helping us improve.');

      setContent('');

    } catch (err) {

      console.error("Submission failed", err);

      setMessage('❌ Failed to submit feedback. Please try again.');

    }

  };

  const handleQuickFeedback = (type: 'positive' | 'negative') => {

    const feedbackText = type === 'positive'

      ? "I'm satisfied with the current experience."

      : "I think there's room for improvement.";

    setContent(feedbackText);

    setRating(type === 'positive' ? 5 : 2);

    setActiveTab('feedback');

  };

  return (

    <div className="min-h-screen bg-gradient-to-br from-sky-50 via-purple-50 to-pink-50 p-6">

      <div className="max-w-5xl mx-auto">

        {/\* Header \*/}

        <div className="text-center mb-10">

          <div className="inline-flex items-center justify-center w-16 h-16 bg-gradient-to-r from-sky-400 to-purple-500 rounded-full mb-4">

            <MessageCircle size={32} className="text-white" />

          </div>

          <h1 className="text-4xl font-bold bg-gradient-to-r from-sky-600 to-purple-600 bg-clip-text text-transparent mb-3">

            We Value Your Feedback

          </h1>

          <p className="text-xl text-gray-600">Help us improve your experience by sharing your thoughts</p>

        </div>

        {/\* Quick Feedback Buttons \*/}

        <div className="flex justify-center gap-6 mb-10">

          <Card

            className="cursor-pointer hover:shadow-xl transition-all duration-300 transform hover:-translate-y-2 bg-gradient-to-br from-green-50 to-emerald-50 border-green-200"

            onClick={() => handleQuickFeedback('positive')}

          >

            <CardContent className="p-8 text-center">

              <div className="inline-flex items-center justify-center w-16 h-16 bg-green-500 rounded-full mb-4">

                <ThumbsUp size={32} className="text-white" />

              </div>

              <h3 className="text-xl font-semibold text-green-700 mb-2">Great Experience</h3>

              <p className="text-green-600">Everything is working well!</p>

            </CardContent>

          </Card>

          <Card

            className="cursor-pointer hover:shadow-xl transition-all duration-300 transform hover:-translate-y-2 bg-gradient-to-br from-orange-50 to-red-50 border-orange-200"

            onClick={() => handleQuickFeedback('negative')}

          >

            <CardContent className="p-8 text-center">

              <div className="inline-flex items-center justify-center w-16 h-16 bg-orange-500 rounded-full mb-4">

                <ThumbsDown size={32} className="text-white" />

              </div>

              <h3 className="text-xl font-semibold text-orange-700 mb-2">Needs Improvement</h3>

              <p className="text-orange-600">Something could be better</p>

            </CardContent>

          </Card>

        </div>

        {/\* Tab Navigation \*/}

        <div className="flex justify-center mb-8">

          <div className="bg-white rounded-xl p-2 shadow-lg">

            {[

              { id: 'feedback', label: 'Detailed Feedback', icon: MessageCircle },

              { id: 'rating', label: 'Rate & Review', icon: Star }

            ].map(tab => {

              const IconComponent = tab.icon;

              return (

                <button

                  key={tab.id}

                  onClick={() => setActiveTab(tab.id)}

                  className={`flex items-center space-x-2 px-6 py-3 rounded-lg font-medium transition-all duration-200 ${

                    activeTab === tab.id

                      ? 'bg-gradient-to-r from-sky-400 to-purple-400 text-white shadow-md'

                      : 'text-gray-600 hover:bg-gray-50'

                  }`}

                >

                  <IconComponent size={18} />

                  <span>{tab.label}</span>

                </button>

              );

            })}

          </div>

        </div>

        {/\* Main Content \*/}

        {activeTab === 'feedback' && (

          <Card className="shadow-2xl border-0 bg-white/80 backdrop-blur-sm">

            <CardHeader className="pb-8">

              <CardTitle className="text-2xl font-bold text-center bg-gradient-to-r from-sky-600 to-purple-600 bg-clip-text text-transparent">

                Share Your Detailed Feedback

              </CardTitle>

            </CardHeader>

            <CardContent className="px-8 pb-8">

              <form onSubmit={handleSubmit} className="space-y-8"> {/\* ✅ Added form here \*/}

                {/\* Category Selection \*/}

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

                  <label className="block text-lg font-semibold text-gray-700">What type of feedback is this?</label>

                  <div className="grid grid-cols-2 md:grid-cols-5 gap-4">

                    {categories.map(cat => {

                      const IconComponent = cat.icon;

                      return (

                        <div

                          key={cat.id}

                          onClick={() => setCategory(cat.id)}

                          className={`cursor-pointer p-4 rounded-xl border-2 transition-all duration-200 text-center ${

                            category === cat.id

                              ? 'border-purple-400 bg-purple-50 transform scale-105'

                              : 'border-gray-200 hover:border-gray-300 hover:bg-gray-50'

                          }`}

                        >

                          <div className={`inline-flex items-center justify-center w-10 h-10 rounded-full mb-2 ${cat.color}`}>

                            <IconComponent size={20} className="text-white" />

                          </div>

                          <p className="text-sm font-medium text-gray-700">{cat.label}</p>

                        </div>

                      );

                    })}

                  </div>

                </div>

                {/\* Suggestions Dropdown \*/}

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

                  <label className="block text-lg font-semibold text-gray-700">Quick suggestions:</label>

                  <select

                    value={content}

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

                    className="w-full p-4 text-lg border-2 border-gray-200 rounded-xl focus:border-purple-400 focus:ring-4 focus:ring-purple-100 transition-all duration-200 bg-white shadow-sm"

                  >

                    <option value="">-- Select a suggestion or write your own below --</option>

                    {suggestions.map((s, i) => (

                      <option key={i} value={s}>{s}</option>

                    ))}

                  </select>

                </div>

                {/\* Custom Feedback Textarea \*/}

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

                  <label className="block text-lg font-semibold text-gray-700">Or write your own feedback:</label>

                  <textarea

                    className="w-full p-4 text-lg border-2 border-gray-200 rounded-xl focus:border-purple-400 focus:ring-4 focus:ring-purple-100 transition-all duration-200 bg-white shadow-sm resize-none"

                    rows={6}

                    placeholder="Tell us what you think! Your feedback helps us make the product better..."

                    value={content}

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

                    maxLength={500}

                  />

                  <div className="flex justify-between text-sm text-gray-500">

                    <span>Be specific and constructive</span>

                    <span>{content.length}/500 characters</span>

                  </div>

                </div>

                {/\* Priority Selection \*/}

                <div className="grid grid-cols-1 md:grid-cols-2 gap-8">

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

                    <label className="block text-lg font-semibold text-gray-700">Priority Level:</label>

                    <div className="space-y-3">

                      {[

                        { value: 'low', label: 'Low Priority', color: 'bg-green-100 border-green-300 text-green-700' },

                        { value: 'medium', label: 'Medium Priority', color: 'bg-yellow-100 border-yellow-300 text-yellow-700' },

                        { value: 'high', label: 'High Priority', color: 'bg-red-100 border-red-300 text-red-700' }

                      ].map(option => (

                        <label key={option.value} className="flex items-center cursor-pointer">

                          <input

                            type="radio"

                            name="priority"

                            value={option.value}

                            checked={priority === option.value}

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

                            className="w-5 h-5 mr-3 text-purple-600"

                          />

                          <span className={`px-4 py-2 rounded-lg border-2 transition-all ${

                            priority === option.value ? option.color : 'bg-gray-50 border-gray-200 text-gray-600'

                          }`}>

                            {option.label}

                          </span>

                        </label>

                      ))}

                    </div>

                  </div>

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

                    <div className="bg-gradient-to-r from-blue-50 to-purple-50 p-6 rounded-xl border border-blue-200">

                      <label className="flex items-start space-x-4 cursor-pointer">

                        <input

                          type="checkbox"

                          checked={contactBack}

                          onChange={(e) => setContactBack(e.target.checked)}

                          className="w-5 h-5 mt-1 text-purple-600 border-2 border-gray-300 rounded focus:ring-purple-500"

                        />

                        <div>

                          <span className="text-lg font-medium text-gray-800">Contact me for follow-up</span>

                          <p className="text-sm text-gray-600 mt-1">We'll reach out if we need more information</p>

                        </div>

                      </label>

                      {contactBack && (

                        <div className="mt-4">

                          <input

                            type="email"

                            placeholder="Enter your email address"

                            value={email}

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

                            className="w-full p-3 border-2 border-gray-200 rounded-lg focus:border-purple-400 focus:ring-2 focus:ring-purple-100 transition-all duration-200"

                          />

                        </div>

                      )}

                    </div>

                  </div>

                </div>

                {/\* Message Display \*/}

                {message && (

                  <div className={`p-4 rounded-xl border-l-4 ${

                    message.includes('successfully')

                      ? 'bg-green-50 border-green-400 text-green-700'

                      : 'bg-red-50 border-red-400 text-red-700'

                  }`}>

                    <p className="font-medium">{message}</p>

                  </div>

                )}

                {/\* Submit Button \*/}

                <div className="flex justify-center pt-4">

                  <Button

                    type="submit"

                    className="bg-gradient-to-r from-sky-400 to-purple-500 hover:from-sky-500 hover:to-purple-600 text-white px-12 py-4 text-lg font-semibold rounded-xl shadow-lg hover:shadow-xl transition-all duration-200 transform hover:-translate-y-1 flex items-center space-x-3"

                  >

                    <Send size={20} />

                    <span>Submit Feedback</span>

                  </Button>

                </div>

              </form>

            </CardContent>

          </Card>

        )}

        {/\* Rating Tab \*/}

        {activeTab === 'rating' && (

          <Card className="shadow-2xl border-0 bg-white/80 backdrop-blur-sm">

            <CardHeader className="pb-8">

              <CardTitle className="text-2xl font-bold text-center bg-gradient-to-r from-sky-600 to-purple-600 bg-clip-text text-transparent">

                Rate Your Experience

              </CardTitle>

            </CardHeader>

            <CardContent className="px-8 pb-8">

              <div className="text-center space-y-8">

                {/\* Star Rating \*/}

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

                  <p className="text-xl text-gray-700">How would you rate your overall experience?</p>

                  <div className="flex justify-center space-x-2">

                    {[1, 2, 3, 4, 5].map((star) => (

                      <button

                        key={star}

                        onClick={() => setRating(star)}

                        className="transition-all duration-200 hover:scale-110"

                      >

                        <Star

                          size={48}

                          className={`${

                            star <= rating

                              ? 'text-yellow-400 fill-yellow-400'

                              : 'text-gray-300 hover:text-yellow-300'

                          }`}

                        />

                      </button>

                    ))}

                  </div>

                  {rating > 0 && (

                    <p className="text-lg font-medium text-gray-700">

                      {rating === 5 ? 'Excellent!' : rating === 4 ? 'Very Good!' : rating === 3 ? 'Good' : rating === 2 ? 'Fair' : 'Poor'}

                    </p>

                  )}

                </div>

                {/\* Quick Review \*/}

                {rating > 0 && (

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

                    <label className="block text-lg font-semibold text-gray-700">Quick review (optional):</label>

                    <textarea

                      className="w-full p-4 text-lg border-2 border-gray-200 rounded-xl focus:border-purple-400 focus:ring-4 focus:ring-purple-100 transition-all duration-200 bg-white shadow-sm resize-none"

                      rows={4}

                      placeholder="Tell others about your experience..."

                      value={content}

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

                      maxLength={500}

                    />

                  </div>

                )}

                {/\* Submit Rating \*/}

                {rating > 0 && (

                  <Button

                    onClick={handleSubmit}

                    className="bg-gradient-to-r from-yellow-400 to-orange-500 hover:from-yellow-500 hover:to-orange-600 text-white px-12 py-4 text-lg font-semibold rounded-xl shadow-lg hover:shadow-xl transition-all duration-200 transform hover:-translate-y-1 flex items-center space-x-3 mx-auto"

                  >

                    <Heart size={20} />

                    <span>Submit Rating</span>

                  </Button>

                )}

              </div>

            </CardContent>

          </Card>

        )}

        {/\* Footer \*/}

        <div className="mt-16 text-center">

          <div className="bg-white/60 backdrop-blur-sm rounded-2xl p-8 shadow-lg">

            <h3 className="text-xl font-semibold text-gray-800 mb-4">Thank You for Your Feedback!</h3>

            <p className="text-gray-600 mb-6">Your input helps us create a better experience for everyone.</p>

            <div className="flex flex-wrap justify-center gap-4">

              <Button className="bg-gray-600 hover:bg-gray-700 text-white px-6 py-3 rounded-lg">

                View Previous Feedback

              </Button>

              <Button className="bg-blue-600 hover:bg-blue-700 text-white px-6 py-3 rounded-lg">

                Contact Support

              </Button>

            </div>

          </div>

        </div>

      </div>

    </div>

  );

};

export default Feedback;

**REWARDS HUB**

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

import { useNavigate } from 'react-router-dom';

import axios from 'axios';

import {

  Card,

  CardHeader,

  CardTitle,

  CardContent,

  CardFooter,

} from '@/components/ui/card';

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

import { Loader2, Trophy, Target, Gift, Star, Clock } from 'lucide-react';

interface Task {

  id: string;

  label: string;

  xp: number;

  start\_time: string;

  end\_time: string;

  expire\_date: string;

}

interface Reward {

  id: number;

  name: string;

  cost: number;

  type: string;

  expire\_date: string;

}

const USER\_ID = 'user1';

const RewardsHub: React.FC = () => {

  const navigate = useNavigate();

  const [xp, setXp] = useState(0);

  const [tasks, setTasks] = useState<Task[]>([]);

  const [completedTasks, setCompletedTasks] = useState<Set<string>>(new Set());

  const [rewards, setRewards] = useState<Reward[]>([]);

  const [redeemed, setRedeemed] = useState<Set<number>>(new Set());

  const [loadingData, setLoadingData] = useState(false);

  const [errorMsg, setErrorMsg] = useState('');

  useEffect(() => {

    fetchData();

  }, []);

  const fetchData = async () => {

    setLoadingData(true);

    setErrorMsg('');

    try {

      const { data } = await axios.get(`/api/rewards/user/${USER\_ID}`);

      setXp(data.xp);

      setTasks(data.tasks);

      setCompletedTasks(new Set(data.completed\_tasks));

      setRewards(data.rewards);

      setRedeemed(new Set(data.redeemed));

    } catch (err) {

      console.error(err);

      setErrorMsg('Failed to load rewards data.');

    } finally {

      setLoadingData(false);

    }

  };

  const completeTask = async (t: Task) => {

    try {

      await axios.post('/api/rewards/task', { user\_id: USER\_ID, task\_id: t.id });

      await fetchData();

      alert(`🎉 You earned ${t.xp} XP for "${t.label}"`);

    } catch {

      alert('Failed to complete task.');

    }

  };

  const redeemReward = async (r: Reward) => {

    try {

      await axios.post('/api/rewards/redeem', { user\_id: USER\_ID, reward\_id: r.id });

      await fetchData();

      alert(`✅ Redeemed ${r.name}`);

      if (r.type === 'App benefit') {

        navigate(r.name.toLowerCase().replace(/\s+/g, ''));

      }

    } catch {

      alert('Failed to redeem reward.');

    }

  };

  if (loadingData) {

    return (

      <div className="flex items-center justify-center min-h-screen bg-gradient-to-br from-slate-50 to-blue-50">

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

          <Loader2 className="w-12 h-12 animate-spin text-blue-500 mx-auto" />

          <p className="text-gray-600 font-medium">Loading rewards...</p>

        </div>

      </div>

    );

  }

  const now = new Date();

  // filter out expired and completed

  const availableTasks = tasks.filter(

    (t) =>

      !completedTasks.has(t.id) &&

      new Date(t.expire\_date) >= now

  );

  const availableRewards = rewards.filter(

    (r) =>

      !redeemed.has(r.id) &&

      new Date(r.expire\_date) >= now

  );

  // slider ticks up to current XP

  const xpThresholds = Array.from(new Set(tasks.map((t) => t.xp)))

    .filter((v) => v > 0 && v <= xp)

    .sort((a, b) => a - b);

  const stops = xpThresholds.filter((v) => v !== xp);

  return (

    <div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-purple-50">

      <div className="container mx-auto px-4 py-6 space-y-8 max-w-7xl">

        {errorMsg && (

          <div className="bg-red-50 border border-red-200 text-red-700 px-4 py-3 rounded-lg text-center font-medium">

            {errorMsg}

          </div>

        )}

        {/\* Header Section \*/}

        <div className="text-center space-y-2 mb-8">

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

            <Trophy className="w-6 h-6 text-yellow-500" />

            <h1 className="text-xl font-bold bg-gradient-to-r from-blue-600 to-purple-600 bg-clip-text text-transparent">

              Rewards Hub

            </h1>

          </div>

          <p className="text-gray-600 text-sm">Complete tasks, earn XP, and unlock amazing rewards!</p>

        </div>

        {/\* XP Status Card \*/}

        <Card className="shadow-lg border-0 bg-gradient-to-r from-blue-400 to-purple-400 text-white overflow-hidden relative">

          <div className="absolute inset-0 bg-white opacity-10"></div>

          <CardHeader className="relative pb-2">

            <CardTitle className="flex items-center gap-2 text-lg font-semibold">

              <Star className="w-5 h-5 text-yellow-300" />

              Your XP: {xp}

            </CardTitle>

          </CardHeader>

          <CardContent className="relative space-y-4">

            <div className="space-y-2">

              <div className="w-full bg-white bg-opacity-20 rounded-full h-4 overflow-hidden">

                <div

                  className="h-4 rounded-full transition-all duration-500 bg-gradient-to-r from-yellow-300 to-orange-300"

                  style={{ width: `${(xp / (xp || 1)) \* 100}%` }}

                />

              </div>

              <div className="flex justify-between text-xs opacity-90">

                <span>0</span>

                {stops.map((value) => (

                  <span key={value}>{value}</span>

                ))}

                <span>{xp}</span>

              </div>

            </div>

            <p className="text-center text-sm opacity-90">

              {completedTasks.size

                ? `Last earned: +${

                    tasks

                      .filter((t) => completedTasks.has(t.id))

                      .sort((a, b) => a.xp - b.xp)

                      .pop()!.xp

                  } XP`

                : 'Complete tasks to earn XP'}

            </p>

          </CardContent>

        </Card>

        {/\* Stats Overview \*/}

        <div className="grid grid-cols-1 md:grid-cols-3 gap-4">

          <Card className="text-center border-0 shadow-md bg-white/80 backdrop-blur">

            <CardContent className="pt-6">

              <div className="flex items-center justify-center mb-2">

                <Target className="w-8 h-8 text-blue-500" />

              </div>

              <h3 className="text-2xl font-bold text-gray-800">{availableTasks.length}</h3>

              <p className="text-sm text-gray-600">Available Tasks</p>

            </CardContent>

          </Card>

          <Card className="text-center border-0 shadow-md bg-white/80 backdrop-blur">

            <CardContent className="pt-6">

              <div className="flex items-center justify-center mb-2">

                <Gift className="w-8 h-8 text-purple-500" />

              </div>

              <h3 className="text-2xl font-bold text-gray-800">{availableRewards.length}</h3>

              <p className="text-sm text-gray-600">Rewards Available</p>

            </CardContent>

          </Card>

          <Card className="text-center border-0 shadow-md bg-white/80 backdrop-blur">

            <CardContent className="pt-6">

              <div className="flex items-center justify-center mb-2">

                <Trophy className="w-8 h-8 text-yellow-500" />

              </div>

              <h3 className="text-2xl font-bold text-gray-800">{completedTasks.size}</h3>

              <p className="text-sm text-gray-600">Tasks Completed</p>

            </CardContent>

          </Card>

        </div>

        {/\* Available Tasks Section \*/}

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

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

            <Target className="w-5 h-5 text-blue-500" />

            <h2 className="text-lg font-semibold text-gray-800">🎯 Available Tasks</h2>

          </div>

          <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4 2xl:grid-cols-5 gap-4">

            {availableTasks.map((t) => (

              <Card key={t.id} className="group hover:shadow-xl transition-all duration-300 border-0 shadow-md bg-white/90 backdrop-blur overflow-hidden flex flex-col justify-between">

                <div className="bg-gradient-to-r from-blue-500 to-purple-600 h-1"></div>

                <CardContent className="p-4 flex-grow">

                  <h3 className="text-base font-semibold mb-2 text-gray-800 group-hover:text-blue-600 transition-colors">

                    {t.label}

                  </h3>

                  <div className="space-y-2 text-sm">

                    <div className="flex items-center gap-1 text-blue-600 font-medium">

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

                      <span>Earn {t.xp} XP</span>

                    </div>

                    <div className="flex items-center gap-1 text-gray-500 text-xs">

                      <Clock className="w-3 h-3" />

                      <span className="truncate">

                        {new Date(t.start\_time).toLocaleString()} – {new Date(t.end\_time).toLocaleString()}

                      </span>

                    </div>

                    <div className="text-red-500 text-xs font-medium">

                      Expires: {new Date(t.expire\_date).toLocaleString()}

                    </div>

                  </div>

                </CardContent>

                <CardFooter className="p-4 pt-0">

                  <Button

                    className="w-full bg-gradient-to-r from-sky-300 to-purple-300 hover:from-sky-400 hover:to-purple-400 text-white shadow-lg transition-all duration-200 group-hover:scale-105"

                    onClick={() => completeTask(t)}

                  >

                    Complete

                  </Button>

                </CardFooter>

              </Card>

            ))}

            {availableTasks.length === 0 && (

              <div className="col-span-full text-center py-12">

                <Target className="w-12 h-12 text-gray-300 mx-auto mb-3" />

                <p className="text-gray-500 font-medium">No tasks available.</p>

              </div>

            )}

          </div>

        </section>

        {/\* Claimable Rewards Section \*/}

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

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

            <Gift className="w-5 h-5 text-purple-500" />

            <h2 className="text-lg font-semibold text-gray-800">🎁 Claimable Rewards</h2>

          </div>

          <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4 2xl:grid-cols-5 gap-4">

            {availableRewards.map((r) => (

              <Card key={r.id} className="group hover:shadow-xl transition-all duration-300 border-0 shadow-md bg-white/90 backdrop-blur overflow-hidden flex flex-col justify-between">

                <div className="bg-gradient-to-r from-purple-500 to-pink-500 h-1"></div>

                <CardContent className="p-4 flex-grow">

                  <h3 className="text-base font-semibold mb-2 text-gray-800 group-hover:text-purple-600 transition-colors">

                    {r.name}

                  </h3>

                  <div className="space-y-2 text-sm">

                    <div className="flex items-center gap-1 text-purple-600 font-medium">

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

                      <span>Cost: {r.cost} XP</span>

                    </div>

                    <div className="text-gray-600 text-xs bg-gray-100 px-2 py-1 rounded-full inline-block">

                      Type: {r.type}

                    </div>

                    <div className="text-red-500 text-xs font-medium">

                      Expires: {new Date(r.expire\_date).toLocaleString()}

                    </div>

                  </div>

                </CardContent>

                <CardFooter className="p-4 pt-0">

                  <Button

                    className="w-full bg-gradient-to-r from-sky-300 to-purple-300 hover:from-sky-400 hover:to-purple-400 text-white shadow-lg transition-all duration-200 group-hover:scale-105"

                    onClick={() => redeemReward(r)}

                  >

                    Redeem

                  </Button>

                </CardFooter>

              </Card>

            ))}

            {availableRewards.length === 0 && (

              <div className="col-span-full text-center py-12">

                <Gift className="w-12 h-12 text-gray-300 mx-auto mb-3" />

                <p className="text-gray-500 font-medium">No rewards to claim.</p>

              </div>

            )}

          </div>

        </section>

      </div>

    </div>

  );

};

export default RewardsHub;

**SMART INVESTMENT**

import React, { useState, useEffect } from "react";

import {

  Card,

  CardHeader,

  CardTitle,

  CardContent

} from "@/components/ui/card";

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

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

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

import {

  LineChart,

  Line,

  XAxis,

  YAxis,

  CartesianGrid,

  Tooltip,

  ResponsiveContainer

} from "recharts";

import { TrendingUp, Calculator, Calendar, DollarSign, Target, AlertCircle, CheckCircle, Loader2 } from "lucide-react";

const SmartInvestment = () => {

  const [mode, setMode] = useState("predict");

  const [form, setForm] = useState({

    amount: "",

    duration: "",

    sip: "",

    start\_month: new Date().toISOString().slice(0, 7),

  });

  const [adminInterestRate, setAdminInterestRate] = useState(null);

  const [calculatedDuration, setCalculatedDuration] = useState(1);

  const [data, setData] = useState({ months: [], values: [], navs: [] });

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

  const [error, setError] = useState("");

  useEffect(() => {

    fetch("http://localhost:5000/api/investment/interest\_rate")

      .then(res => res.json())

      .then(json => setAdminInterestRate(json.interest\_rate))

      .catch(() => setAdminInterestRate(null));

  }, []);

  useEffect(() => {

    if (form.start\_month) {

      const start = new Date(form.start\_month + "-01");

      const now = new Date();

      let months =

        now.getFullYear() \* 12 +

        now.getMonth() -

        (start.getFullYear() \* 12 + start.getMonth()) +

        1;

      if (months < 1) months = 1;

      setCalculatedDuration(months);

    }

  }, [form.start\_month]);

  const handleChange = (e) => {

    setForm({ ...form, [e.target.name]: e.target.value });

  };

  const validate = () => {

    if (!form.amount || isNaN(Number(form.amount)) || Number(form.amount) <= 0) {

      setError("Enter a valid amount greater than zero.");

      return false;

    }

    if (form.duration && (isNaN(Number(form.duration)) || Number(form.duration) < 1)) {

      setError("Duration, if given, must be a positive number.");

      return false;

    }

    if (!form.start\_month || !/^\d{4}-\d{2}$/.test(form.start\_month)) {

      setError("Select a valid start month (YYYY-MM).");

      return false;

    }

    if (mode === "predict") {

      if (adminInterestRate === null || isNaN(Number(adminInterestRate)) || Number(adminInterestRate) <= 0) {

        setError("Interest rate set by admin is invalid.");

        return false;

      }

      if (form.sip && (isNaN(Number(form.sip)) || Number(form.sip) < 0)) {

        setError("Monthly SIP must be zero or a positive number.");

        return false;

      }

    }

    setError("");

    return true;

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    if (!validate()) return;

    setMessage("Calculating...");

    try {

      const durationToSend =

        form.duration && Number(form.duration) > 0

          ? Number(form.duration)

          : calculatedDuration;

      const body = {

        ...form,

        mode,

        duration: durationToSend,

      };

      if (mode === "predict") {

        body.interest\_rate = adminInterestRate;

      }

      const res = await fetch("http://localhost:5000/api/investment/track", {

        method: "POST",

        headers: { "Content-Type": "application/json" },

        body: JSON.stringify(body),

      });

      const result = await res.json();

      if (result.months && result.values) {

        setData(result);

        setMessage("Success");

      } else if (result.error) {

        setMessage(result.error);

        setData({ months: [], values: [], navs: [] });

      } else {

        setMessage("No result returned");

        setData({ months: [], values: [], navs: [] });

      }

    } catch {

      setMessage("Failed to fetch");

    }

  };

  const CustomTooltip = ({ active, payload, label }) => {

    if (active && payload && payload.length) {

      const nav =

        data.navs && data.navs.length > payload[0].payload.index

          ? data.navs[payload[0].payload.index]

          : null;

      return (

        <div className="bg-gradient-to-br from-white to-gray-50 p-4 rounded-xl shadow-xl border border-gray-200 backdrop-blur-sm">

          <div className="font-semibold text-gray-800 mb-2 text-base">

            {label}

          </div>

          <div className="text-emerald-600 font-semibold text-base">

            Amount: ₹{payload[0].value?.toLocaleString('en-IN')}

          </div>

          {nav !== null && nav !== undefined && (

            <div className="text-blue-600 font-semibold text-sm mt-1">

              NAV: {typeof nav === "number" ? nav.toFixed(2) : nav}

            </div>

          )}

        </div>

      );

    }

    return null;

  };

  return (

    <div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-indigo-100 p-6">

      <div className="max-w-6xl mx-auto">

        {/\* Enhanced Header \*/}

        <div className="text-center mb-10">

          <div className="flex justify-center items-center gap-3 mb-4">

            <div className="p-3 bg-gradient-to-r from-indigo-500 to-purple-600 rounded-xl shadow-lg">

              <TrendingUp className="w-8 h-8 text-white" />

            </div>

          </div>

          <p className="text-gray-600 text-lg max-w-2xl mx-auto">

            Elevate your investment journey with intelligent tracking and predictive analytics

          </p>

        </div>

        {/\* Main Content \*/}

        <Card className="bg-white/90 backdrop-blur-lg rounded-2xl shadow-xl border-0 hover:shadow-2xl transition-all duration-300">

          <CardHeader className="pb-6">

            <div className="flex items-center gap-3">

              <div className={`p-2 rounded-xl shadow-md ${mode === 'predict' ? 'bg-gradient-to-r from-purple-500 to-indigo-500' : 'bg-gradient-to-r from-emerald-500 to-teal-500'} transition-all duration-300`}>

                {mode === 'predict' ?

                  <Calculator className="w-6 h-6 text-white" /> :

                  <Target className="w-6 h-6 text-white" />

                }

              </div>

              <CardTitle className="text-2xl font-bold bg-gradient-to-r from-gray-800 to-gray-600 bg-clip-text text-transparent">

                {mode === 'predict' ? 'Investment Prediction Engine' : 'Portfolio Progress Tracker'}

              </CardTitle>

            </div>

          </CardHeader>

          <CardContent className="space-y-6">

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

              {/\* Enhanced Mode Selection \*/}

              <div className="p-5 bg-gradient-to-r from-gray-50 to-blue-50 rounded-xl shadow-inner">

                <Label className="text-base font-semibold text-gray-800 mb-4 block flex items-center gap-2">

                  <div className="w-2 h-2 bg-indigo-500 rounded-full animate-pulse"></div>

                  Select Investment Mode

                </Label>

                <div className="flex gap-3">

                  <button

                    type="button"

                    onClick={() => setMode('predict')}

                    className={`flex-1 p-4 rounded-xl font-semibold text-base transition-all duration-300 transform ${

                      mode === 'predict'

                        ? 'bg-gradient-to-r from-purple-400 to-indigo-400 text-white shadow-lg scale-105'

                        : 'bg-white text-gray-700 hover:bg-gray-50 border-2 border-gray-200 hover:border-purple-300'

                    }`}

                  >

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

                    Future Prediction

                  </button>

                  <button

                    type="button"

                    onClick={() => setMode('track')}

                    className={`flex-1 p-4 rounded-xl font-semibold text-base transition-all duration-300 transform ${

                      mode === 'track'

                        ? 'bg-gradient-to-r from-emerald-400 to-teal-400 text-white shadow-lg scale-105'

                        : 'bg-white text-gray-700 hover:bg-gray-50 border-2 border-gray-200 hover:border-emerald-300'

                    }`}

                  >

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

                    Track Performance

                  </button>

                </div>

              </div>

              {/\* Enhanced Form Grid \*/}

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

                {/\* Amount Field \*/}

                <div className="space-y-2">

                  <Label className="text-sm font-semibold text-gray-800 flex items-center gap-2">

                    <div className="p-1.5 bg-green-100 rounded-lg">

                      <DollarSign className="w-4 h-4 text-green-600" />

                    </div>

                    Investment Amount

                  </Label>

                  <div className="relative group">

                    <span className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-600 font-semibold">₹</span>

                    <Input

                      name="amount"

                      type="number"

                      value={form.amount}

                      onChange={handleChange}

                      required

                      className="pl-8 h-12 border-2 border-gray-300 rounded-xl focus:border-indigo-500 focus:ring-4 focus:ring-indigo-100 transition-all duration-300 text-base font-medium group-hover:border-indigo-400"

                      placeholder="Enter investment amount"

                    />

                  </div>

                  {form.amount && (

                    <div className="p-2 bg-indigo-50 rounded-lg border border-indigo-200">

                      <p className="text-xs text-indigo-700 font-medium">

                        💰 ₹{Number(form.amount).toLocaleString('en-IN')}

                      </p>

                    </div>

                  )}

                </div>

                {/\* Duration Field \*/}

                <div className="space-y-2">

                  <Label className="text-sm font-semibold text-gray-800 flex items-center gap-2">

                    <div className="p-1.5 bg-blue-100 rounded-lg">

                      <Calendar className="w-4 h-4 text-blue-600" />

                    </div>

                    Investment Duration

                  </Label>

                  <Input

                    name="duration"

                    type="number"

                    placeholder={`Auto: ${calculatedDuration} months`}

                    value={form.duration}

                    onChange={handleChange}

                    min={1}

                    className="h-12 border-2 border-gray-300 rounded-xl focus:border-indigo-500 focus:ring-4 focus:ring-indigo-100 transition-all duration-300 text-base font-medium"

                  />

                  <div className="p-2 bg-amber-50 rounded-lg border border-amber-200">

                    <div className="text-xs text-amber-700 font-medium">

                      🕒 Auto-calculated: {calculatedDuration} months from start date

                    </div>

                  </div>

                </div>

                {/\* Start Month Field \*/}

                <div className="space-y-2">

                  <Label className="text-sm font-semibold text-gray-800 flex items-center gap-2">

                    <div className="p-1.5 bg-purple-100 rounded-lg">

                      <Calendar className="w-4 h-4 text-purple-600" />

                    </div>

                    Start Month

                  </Label>

                  <Input

                    name="start\_month"

                    type="month"

                    value={form.start\_month}

                    onChange={handleChange}

                    required

                    className="h-12 border-2 border-gray-300 rounded-xl focus:border-indigo-500 focus:ring-4 focus:ring-indigo-100 transition-all duration-300 text-base font-medium"

                  />

                </div>

                {mode === "predict" && (

                  <>

                    {/\* Interest Rate Field \*/}

                    <div className="space-y-2">

                      <Label className="text-sm font-semibold text-gray-800 flex items-center gap-2">

                        <div className="p-1.5 bg-orange-100 rounded-lg">

                          <TrendingUp className="w-4 h-4 text-orange-600" />

                        </div>

                        Interest Rate

                      </Label>

                      <div className="relative">

                        <Input

                          name="interest\_rate"

                          type="number"

                          value={adminInterestRate !== null ? adminInterestRate : ""}

                          readOnly

                          className="h-12 bg-gradient-to-r from-gray-100 to-gray-200 cursor-not-allowed border-2 border-gray-300 rounded-xl text-base font-medium"

                        />

                        <div className="absolute right-3 top-1/2 transform -translate-y-1/2 flex items-center gap-2">

                          <div className="w-2 h-2 bg-green-500 rounded-full animate-pulse"></div>

                          <span className="text-xs text-gray-600 font-medium">Admin Set</span>

                        </div>

                      </div>

                      <div className="p-2 bg-red-50 rounded-lg border border-red-200">

                        <div className="text-xs text-red-700 font-medium">

                          🔐 Rate locked by administrator - {adminInterestRate}% per annum

                        </div>

                      </div>

                    </div>

                    {/\* SIP Field \*/}

                    <div className="space-y-2 md:col-span-2">

                      <Label className="text-sm font-semibold text-gray-800 flex items-center gap-2">

                        <div className="p-1.5 bg-teal-100 rounded-lg">

                          <DollarSign className="w-4 h-4 text-teal-600" />

                        </div>

                        Monthly SIP Contribution

                      </Label>

                      <div className="relative group">

                        <span className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-600 font-semibold">₹</span>

                        <Input

                          name="sip"

                          type="number"

                          value={form.sip}

                          onChange={handleChange}

                          className="pl-8 h-12 border-2 border-gray-300 rounded-xl focus:border-indigo-500 focus:ring-4 focus:ring-indigo-100 transition-all duration-300 text-base font-medium group-hover:border-teal-400"

                          placeholder="Optional monthly SIP amount"

                        />

                      </div>

                      {form.sip && (

                        <div className="p-2 bg-teal-50 rounded-lg border border-teal-200">

                          <p className="text-xs text-teal-700 font-medium">

                            📈 Monthly: ₹{Number(form.sip).toLocaleString('en-IN')}

                          </p>

                        </div>

                      )}

                    </div>

                  </>

                )}

              </div>

              {/\* Enhanced Submit Button \*/}

              <Button

                onClick={handleSubmit}

                className={`w-full h-12 rounded-xl font-semibold text-base transition-all duration-300 transform hover:scale-105 shadow-lg ${

                  mode === 'predict'

                    ? 'bg-gradient-to-r from-purple-400 to-indigo-400 hover:from-purple-700 hover:to-indigo-700'

                    : 'bg-gradient-to-r from-emerald-400 to-teal-400 hover:from-emerald-700 hover:to-teal-700'

                } text-white hover:shadow-xl`}

              >

                <div className="flex items-center justify-center gap-3">

                  {message === "Calculating..." ? (

                    <>

                      <Loader2 className="w-5 h-5 animate-spin" />

                      Processing Investment Data...

                    </>

                  ) : (

                    <>

                      {mode === 'predict' ? <Calculator className="w-5 h-5" /> : <Target className="w-5 h-5" />}

                      {mode === 'predict' ? 'Generate Smart Prediction' : 'Analyze Portfolio Performance'}

                    </>

                  )}

                </div>

              </Button>

              {/\* Enhanced Status Messages \*/}

              {(error || message) && (

                <div className={`p-4 rounded-xl border-2 transition-all duration-300 ${

                  error

                    ? 'bg-gradient-to-r from-red-50 to-pink-50 border-red-300 text-red-800'

                    : message === 'Success'

                    ? 'bg-gradient-to-r from-green-50 to-emerald-50 border-green-300 text-green-800'

                    : 'bg-gradient-to-r from-blue-50 to-indigo-50 border-blue-300 text-blue-800'

                } shadow-md`}>

                  <div className="flex items-center gap-3">

                    {error ? <AlertCircle className="w-5 h-5" /> :

                     message === 'Success' ? <CheckCircle className="w-5 h-5" /> :

                     <Loader2 className="w-5 h-5 animate-spin" />}

                    <span className="font-semibold text-base">

                      {error || message}

                    </span>

                  </div>

                </div>

              )}

              {/\* Enhanced Chart \*/}

              {data.values.length > 0 && (

                <div className="mt-8 p-6 bg-gradient-to-br from-white to-gray-50 rounded-2xl shadow-xl border border-gray-200">

                  <div className="mb-6">

                    <div className="flex items-center gap-3 mb-4">

                      <div className="p-2 bg-gradient-to-r from-emerald-500 to-teal-500 rounded-xl shadow-md">

                        <TrendingUp className="w-6 h-6 text-white" />

                      </div>

                      <h3 className="text-2xl font-bold bg-gradient-to-r from-gray-800 to-gray-600 bg-clip-text text-transparent">

                        Investment Growth Visualization

                      </h3>

                    </div>

                   {/\* Enhanced Stats Grid \*/}

                    <div className="grid grid-cols-1 md:grid-cols-3 gap-4 mb-6">

                      <div className="bg-gradient-to-br from-emerald-500 to-teal-500 p-4 rounded-xl text-white shadow-lg">

                        <div className="text-xs opacity-90 font-medium">Initial Investment</div>

                        <div className="text-xl font-bold">₹{data.values[0]?.toLocaleString('en-IN')}</div>

                      </div>

                      <div className="bg-gradient-to-br from-blue-500 to-indigo-500 p-4 rounded-xl text-white shadow-lg">

                        <div className="text-xs opacity-90 font-medium">Final Value</div>

                        <div className="text-xl font-bold">₹{data.values[data.values.length - 1]?.toLocaleString('en-IN')}</div>

                      </div>

                      <div className="bg-gradient-to-br from-purple-500 to-pink-500 p-4 rounded-xl text-white shadow-lg">

                        <div className="text-xs opacity-90 font-medium">Total Growth</div>

                        <div className="text-xl font-bold">

                          {data.values.length > 1

                            ? ((data.values[data.values.length - 1] - data.values[0]) / data.values[0] \* 100).toFixed(1)

                            : 0

                          }%

                        </div>

                      </div>

                    </div>

                  </div>

                  <div className="h-80 w-full">

                    <ResponsiveContainer width="100%" height="100%">

                      <LineChart

                        data={data.months.map((m, i) => ({

                          month: m,

                          value: data.values[i],

                          nav: data.navs && data.navs[i],

                          index: i,

                        }))}

                        margin={{ top: 20, right: 30, left: 20, bottom: 20 }}

                      >

                        <defs>

                          <linearGradient id="colorGradient" x1="0" y1="0" x2="0" y2="1">

                            <stop offset="5%" stopColor="#4f46e5" stopOpacity={0.3}/>

                            <stop offset="95%" stopColor="#4f46e5" stopOpacity={0.05}/>

                          </linearGradient>

                          <linearGradient id="strokeGradient" x1="0%" y1="0%" x2="100%" y2="0%">

                            <stop offset="0%" stopColor="#8b5cf6" />

                            <stop offset="50%" stopColor="#4f46e5" />

                            <stop offset="100%" stopColor="#06b6d4" />

                          </linearGradient>

                        </defs>

                        <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" strokeWidth={1} />

                        <XAxis

                          dataKey="month"

                          tick={{ fontSize: 12, fontWeight: 600 }}

                          stroke="#64748b"

                          tickLine={{ stroke: '#64748b', strokeWidth: 1 }}

                        />

                        <YAxis

                          tick={{ fontSize: 12, fontWeight: 600 }}

                          stroke="#64748b"

                          tickFormatter={(value) => `₹${(value/1000).toFixed(0)}K`}

                          tickLine={{ stroke: '#64748b', strokeWidth: 1 }}

                        />

                        <Tooltip content={CustomTooltip} />

                        <Line

                          type="monotone"

                          dataKey="value"

                          stroke="url(#strokeGradient)"

                          strokeWidth={3}

                          dot={{ fill: '#4f46e5', strokeWidth: 2, r: 4, stroke: '#fff' }}

                          activeDot={{ r: 6, fill: '#4f46e5', strokeWidth: 3, stroke: '#fff', filter: 'drop-shadow(0 0 4px rgba(79, 70, 229, 0.3))' }}

                          fill="url(#colorGradient)"

                          filter="drop-shadow(0 1px 2px rgba(0,0,0,0.1))"

                        />

                      </LineChart>

                    </ResponsiveContainer>

                  </div>

                </div>

              )}

            </div>

          </CardContent>

        </Card>

      </div>

    </div>

  );

};

export default SmartInvestment;