**User Profile Options**

When someone creates an account or before starting their first assessment, we ask short profile questions. These then **tailor results, insights, and progress tracking**.

**1. Basic Demographics**

* **Role/Position:** (Executive, Mid-level Manager, Team Lead, Individual Contributor, Entrepreneur)
* **Company Size:** (Solo, Micro Startup, SMB, Mid-size, Enterprise)
* **Industry:** (Tech, Healthcare, Finance, Construction, Other)
* **Years in Leadership / Experience Level**

**2. Development Goals**

* Primary reason for taking assessment:
  + Improve leadership agility
  + Build stronger team collaboration
  + Balance short-term vs. long-term
  + Prepare for promotion
  + Personal growth

**3. Contextual Variables**

* **Team Size Managed**
* **Decision-Making Environment:** (Stable, Volatile, Growth, Turnaround)
* **Key Challenges:** (Delegation, Conflict, Risk-taking, Vision execution, etc.)

import { useState, useEffect } from "react";

import { useSimpleAuth } from "@/hooks/useSimpleAuth";

import { useToast } from "@/hooks/use-toast";

import { apiRequest, queryClient } from "@/lib/queryClient";

import { useMutation, useQuery } from "@tanstack/react-query";

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

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

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

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

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

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";

import { Switch } from "@/components/ui/switch";

import { User, Edit3, Save, X } from "lucide-react";

import { type User as UserType } from "@shared/schema";

export default function Profile() {

const { user, isLoading: authLoading } = useSimpleAuth();

const { toast } = useToast();

const [isEditing, setIsEditing] = useState(false);

const [formData, setFormData] = useState({

firstName: "",

lastName: "",

phone: "",

age: "",

gender: "",

address: "",

occupation: "",

workType: "",

industry: "",

incomeRange: "",

relationshipStatus: "",

hasChildren: false,

fitnessLevel: "",

primaryHealthGoal: "",

primaryFinanceGoal: "",

primaryRelationshipGoal: "",

primaryInnerPeaceGoal: "",

preferredLearningStyle: "",

timezone: "",

bio: "",

goals: "",

});

// Get user profile data

const { data: fullUserData, isLoading: profileLoading } = useQuery<UserType>({

queryKey: ['/api/auth/user'],

enabled: !!user,

});

// Initialize form data when user data loads

useEffect(() => {

if (fullUserData) {

setFormData({

firstName: fullUserData.firstName || "",

lastName: fullUserData.lastName || "",

phone: fullUserData.phone || "",

age: fullUserData.age?.toString() || "",

gender: fullUserData.gender || "",

address: fullUserData.address || "",

occupation: fullUserData.occupation || "",

workType: fullUserData.workType || "",

industry: fullUserData.industry || "",

incomeRange: fullUserData.incomeRange || "",

relationshipStatus: fullUserData.relationshipStatus || "",

hasChildren: fullUserData.hasChildren || false,

fitnessLevel: fullUserData.fitnessLevel || "",

primaryHealthGoal: fullUserData.primaryHealthGoal || "",

primaryFinanceGoal: fullUserData.primaryFinanceGoal || "",

primaryRelationshipGoal: fullUserData.primaryRelationshipGoal || "",

primaryInnerPeaceGoal: fullUserData.primaryInnerPeaceGoal || "",

preferredLearningStyle: fullUserData.preferredLearningStyle || "visual",

timezone: fullUserData.timezone || "America/New\_York",

bio: fullUserData.bio || "",

goals: fullUserData.goals || "",

});

}

}, [fullUserData]);

// Profile update mutation

const updateProfileMutation = useMutation({

mutationFn: async (data: any) => {

const updateData = {

...data,

age: data.age ? parseInt(data.age) : undefined,

};

return await apiRequest('PUT', '/api/profile', updateData);

},

onSuccess: () => {

toast({

title: "Profile Updated",

description: "Your profile has been successfully saved.",

});

setIsEditing(false);

queryClient.invalidateQueries({ queryKey: ['/api/auth/user'] });

},

onError: (error) => {

toast({

title: "Error",

description: "Failed to save profile. Please try again.",

variant: "destructive",

});

console.error('Profile update error:', error);

},

});

const handleSaveProfile = () => {

updateProfileMutation.mutate(formData);

};

const handleCancelEdit = () => {

// Reset to original data

if (fullUserData) {

setFormData({

firstName: fullUserData.firstName || "",

lastName: fullUserData.lastName || "",

phone: fullUserData.phone || "",

age: fullUserData.age?.toString() || "",

gender: fullUserData.gender || "",

address: fullUserData.address || "",

occupation: fullUserData.occupation || "",

workType: fullUserData.workType || "",

industry: fullUserData.industry || "",

incomeRange: fullUserData.incomeRange || "",

relationshipStatus: fullUserData.relationshipStatus || "",

hasChildren: fullUserData.hasChildren || false,

fitnessLevel: fullUserData.fitnessLevel || "",

primaryHealthGoal: fullUserData.primaryHealthGoal || "",

primaryFinanceGoal: fullUserData.primaryFinanceGoal || "",

primaryRelationshipGoal: fullUserData.primaryRelationshipGoal || "",

primaryInnerPeaceGoal: fullUserData.primaryInnerPeaceGoal || "",

preferredLearningStyle: fullUserData.preferredLearningStyle || "visual",

timezone: fullUserData.timezone || "America/New\_York",

bio: fullUserData.bio || "",

goals: fullUserData.goals || "",

});

}

setIsEditing(false);

};

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

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

};

const handleSwitchChange = (field: string, value: boolean) => {

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

};

if (authLoading || profileLoading) {

return (

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

<div className="animate-pulse space-y-8">

<div className="h-8 bg-gray-300 rounded w-1/4"></div>

<div className="h-96 bg-gray-300 rounded"></div>

</div>

</div>

);

}

return (

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

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

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

<User className="h-8 w-8" />

My Profile

</h1>

{!isEditing ? (

<Button

onClick={() => setIsEditing(true)}

className="flex items-center gap-2"

>

<Edit3 className="h-4 w-4" />

Edit Profile

</Button>

) : (

<div className="flex gap-2">

<Button

onClick={handleSaveProfile}

disabled={updateProfileMutation.isPending}

className="flex items-center gap-2"

>

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

{updateProfileMutation.isPending ? "Saving..." : "Save"}

</Button>

<Button

variant="outline"

onClick={handleCancelEdit}

className="flex items-center gap-2"

>

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

Cancel

</Button>

</div>

)}

</div>

<div className="max-w-2xl">

<Card>

<CardHeader>

<CardTitle>Personal Information</CardTitle>

<CardDescription>Update your personal details</CardDescription>

</CardHeader>

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

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

<div>

<Label htmlFor="firstName">First Name</Label>

<Input

id="firstName"

value={formData.firstName}

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

disabled={!isEditing}

placeholder="Enter your first name"

/>

</div>

<div>

<Label htmlFor="lastName">Last Name</Label>

<Input

id="lastName"

value={formData.lastName}

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

disabled={!isEditing}

placeholder="Enter your last name"

/>

</div>

<div>

<Label htmlFor="email">Email</Label>

<Input

id="email"

value={user?.email || ""}

disabled={true}

className="bg-gray-50"

/>

<p className="text-xs text-gray-500 mt-1">Email cannot be changed</p>

</div>

<div>

<Label htmlFor="phone">Phone</Label>

<Input

id="phone"

value={formData.phone}

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

disabled={!isEditing}

placeholder="+1 (555) 123-4567"

/>

</div>

<div>

<Label htmlFor="age">Age Range</Label>

<Select

value={formData.age}

onValueChange={(value) => handleInputChange('age', value)}

disabled={!isEditing}

>

<SelectTrigger>

<SelectValue placeholder="Select age range" />

</SelectTrigger>

<SelectContent>

<SelectItem value="18">18-24</SelectItem>

<SelectItem value="25">25-34</SelectItem>

<SelectItem value="35">35-44</SelectItem>

<SelectItem value="45">45-54</SelectItem>

<SelectItem value="55">55-64</SelectItem>

<SelectItem value="65">65+</SelectItem>

</SelectContent>

</Select>

</div>

<div>

<Label htmlFor="gender">Gender</Label>

<Select

value={formData.gender}

onValueChange={(value) => handleInputChange('gender', value)}

disabled={!isEditing}

>

<SelectTrigger>

<SelectValue placeholder="Select gender" />

</SelectTrigger>

<SelectContent>

<SelectItem value="male">Male</SelectItem>

<SelectItem value="female">Female</SelectItem>

<SelectItem value="non-binary">Non-binary</SelectItem>

<SelectItem value="prefer-not-to-say">Prefer not to say</SelectItem>

</SelectContent>

</Select>

</div>

</div>

<div>

<Label htmlFor="address">Address</Label>

<Textarea

id="address"

value={formData.address}

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

disabled={!isEditing}

placeholder="Street address, city, state, country"

rows={2}

/>

</div>

<div>

<Label htmlFor="occupation">Occupation</Label>

<Input

id="occupation"

value={formData.occupation}

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

disabled={!isEditing}

placeholder="Your occupation"

/>

</div>

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

<div>

<Label htmlFor="workType">Work Type</Label>

<Select

value={formData.workType}

onValueChange={(value) => handleInputChange('workType', value)}

disabled={!isEditing}

>

<SelectTrigger>

<SelectValue placeholder="Select work type" />

</SelectTrigger>

<SelectContent>

<SelectItem value="full-time">Full-time</SelectItem>

<SelectItem value="part-time">Part-time</SelectItem>

<SelectItem value="contract">Contract</SelectItem>

<SelectItem value="freelance">Freelance</SelectItem>

<SelectItem value="self-employed">Self-employed</SelectItem>

<SelectItem value="unemployed">Unemployed</SelectItem>

<SelectItem value="retired">Retired</SelectItem>

<SelectItem value="student">Student</SelectItem>

</SelectContent>

</Select>

</div>

<div>

<Label htmlFor="industry">Industry</Label>

<Input

id="industry"

value={formData.industry}

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

disabled={!isEditing}

placeholder="e.g., Technology, Healthcare, Finance"

/>

</div>

</div>

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

<div>

<Label htmlFor="incomeRange">Income Range</Label>

<Select

value={formData.incomeRange}

onValueChange={(value) => handleInputChange('incomeRange', value)}

disabled={!isEditing}

>

<SelectTrigger>

<SelectValue placeholder="Select income range" />

</SelectTrigger>

<SelectContent>

<SelectItem value="under-25k">Under $25,000</SelectItem>

<SelectItem value="25k-50k">$25,000 - $50,000</SelectItem>

<SelectItem value="50k-75k">$50,000 - $75,000</SelectItem>

<SelectItem value="75k-100k">$75,000 - $100,000</SelectItem>

<SelectItem value="100k-150k">$100,000 - $150,000</SelectItem>

<SelectItem value="150k-250k">$150,000 - $250,000</SelectItem>

<SelectItem value="over-250k">Over $250,000</SelectItem>

<SelectItem value="prefer-not-to-say">Prefer not to say</SelectItem>

</SelectContent>

</Select>

</div>

<div>

<Label htmlFor="fitnessLevel">Fitness Level</Label>

<Select

value={formData.fitnessLevel}

onValueChange={(value) => handleInputChange('fitnessLevel', value)}

disabled={!isEditing}

>

<SelectTrigger>

<SelectValue placeholder="Select fitness level" />

</SelectTrigger>

<SelectContent>

<SelectItem value="sedentary">Sedentary</SelectItem>

<SelectItem value="lightly-active">Lightly Active</SelectItem>

<SelectItem value="moderately-active">Moderately Active</SelectItem>

<SelectItem value="very-active">Very Active</SelectItem>

<SelectItem value="extremely-active">Extremely Active</SelectItem>

</SelectContent>

</Select>

</div>

</div>

<div>

<Label htmlFor="relationshipStatus">Relationship Status</Label>

<Select

value={formData.relationshipStatus}

onValueChange={(value) => handleInputChange('relationshipStatus', value)}

disabled={!isEditing}

>

<SelectTrigger>

<SelectValue placeholder="Select relationship status" />

</SelectTrigger>

<SelectContent>

<SelectItem value="single">Single</SelectItem>

<SelectItem value="in-relationship">In a Relationship</SelectItem>

<SelectItem value="married">Married</SelectItem>

<SelectItem value="divorced">Divorced</SelectItem>

<SelectItem value="widowed">Widowed</SelectItem>

<SelectItem value="prefer-not-to-say">Prefer not to say</SelectItem>

</SelectContent>

</Select>

</div>

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

<Switch

id="hasChildren"

checked={formData.hasChildren}

onCheckedChange={(checked) => handleSwitchChange('hasChildren', checked)}

disabled={!isEditing}

/>

<Label htmlFor="hasChildren">I have children</Label>

</div>

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

<h3 className="text-lg font-semibold">Primary Goals</h3>

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

<div>

<Label htmlFor="primaryHealthGoal">Health Goal</Label>

<Select

value={formData.primaryHealthGoal}

onValueChange={(value) => handleInputChange('primaryHealthGoal', value)}

disabled={!isEditing}

>

<SelectTrigger>

<SelectValue placeholder="Select health goal" />

</SelectTrigger>

<SelectContent>

<SelectItem value="weight-loss">Weight Loss</SelectItem>

<SelectItem value="muscle-gain">Muscle Gain</SelectItem>

<SelectItem value="general-fitness">General Fitness</SelectItem>

<SelectItem value="stress-management">Stress Management</SelectItem>

<SelectItem value="energy-improvement">Energy Improvement</SelectItem>

<SelectItem value="better-sleep">Better Sleep</SelectItem>

<SelectItem value="chronic-condition">Managing Chronic Condition</SelectItem>

</SelectContent>

</Select>

</div>

<div>

<Label htmlFor="primaryFinanceGoal">Finance Goal</Label>

<Select

value={formData.primaryFinanceGoal}

onValueChange={(value) => handleInputChange('primaryFinanceGoal', value)}

disabled={!isEditing}

>

<SelectTrigger>

<SelectValue placeholder="Select finance goal" />

</SelectTrigger>

<SelectContent>

<SelectItem value="debt-reduction">Debt Reduction</SelectItem>

<SelectItem value="emergency-fund">Emergency Fund</SelectItem>

<SelectItem value="investment-growth">Investment Growth</SelectItem>

<SelectItem value="retirement-planning">Retirement Planning</SelectItem>

<SelectItem value="home-purchase">Home Purchase</SelectItem>

<SelectItem value="increase-income">Increase Income</SelectItem>

<SelectItem value="financial-education">Financial Education</SelectItem>

</SelectContent>

</Select>

</div>

<div>

<Label htmlFor="primaryRelationshipGoal">Relationship Goal</Label>

<Select

value={formData.primaryRelationshipGoal}

onValueChange={(value) => handleInputChange('primaryRelationshipGoal', value)}

disabled={!isEditing}

>

<SelectTrigger>

<SelectValue placeholder="Select relationship goal" />

</SelectTrigger>

<SelectContent>

<SelectItem value="find-partner">Find Life Partner</SelectItem>

<SelectItem value="improve-communication">Improve Communication</SelectItem>

<SelectItem value="resolve-conflicts">Resolve Conflicts</SelectItem>

<SelectItem value="build-trust">Build Trust</SelectItem>

<SelectItem value="strengthen-family">Strengthen Family Bonds</SelectItem>

<SelectItem value="social-connections">Expand Social Connections</SelectItem>

<SelectItem value="set-boundaries">Set Healthy Boundaries</SelectItem>

</SelectContent>

</Select>

</div>

<div>

<Label htmlFor="primaryInnerPeaceGoal">Inner Peace Goal</Label>

<Select

value={formData.primaryInnerPeaceGoal}

onValueChange={(value) => handleInputChange('primaryInnerPeaceGoal', value)}

disabled={!isEditing}

>

<SelectTrigger>

<SelectValue placeholder="Select inner peace goal" />

</SelectTrigger>

<SelectContent>

<SelectItem value="stress-reduction">Stress Reduction</SelectItem>

<SelectItem value="meditation-practice">Meditation Practice</SelectItem>

<SelectItem value="emotional-regulation">Emotional Regulation</SelectItem>

<SelectItem value="mindfulness">Mindfulness</SelectItem>

<SelectItem value="self-acceptance">Self-Acceptance</SelectItem>

<SelectItem value="purpose-clarity">Find Life Purpose</SelectItem>

<SelectItem value="spiritual-growth">Spiritual Growth</SelectItem>

</SelectContent>

</Select>

</div>

</div>

</div>

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

<div>

<Label htmlFor="preferredLearningStyle">Learning Style</Label>

<Select

value={formData.preferredLearningStyle}

onValueChange={(value) => handleInputChange('preferredLearningStyle', value)}

disabled={!isEditing}

>

<SelectTrigger>

<SelectValue placeholder="Select learning style" />

</SelectTrigger>

<SelectContent>

<SelectItem value="visual">Visual (Reading, Charts)</SelectItem>

<SelectItem value="auditory">Auditory (Listening, Podcasts)</SelectItem>

<SelectItem value="kinesthetic">Hands-on (Practice, Exercises)</SelectItem>

<SelectItem value="mixed">Mixed Approach</SelectItem>

</SelectContent>

</Select>

</div>

<div>

<Label htmlFor="timezone">Timezone</Label>

<Select

value={formData.timezone}

onValueChange={(value) => handleInputChange('timezone', value)}

disabled={!isEditing}

>

<SelectTrigger>

<SelectValue placeholder="Select timezone" />

</SelectTrigger>

<SelectContent>

<SelectItem value="America/New\_York">Eastern Time (EST/EDT)</SelectItem>

<SelectItem value="America/Chicago">Central Time (CST/CDT)</SelectItem>

<SelectItem value="America/Denver">Mountain Time (MST/MDT)</SelectItem>

<SelectItem value="America/Los\_Angeles">Pacific Time (PST/PDT)</SelectItem>

<SelectItem value="Europe/London">GMT/BST</SelectItem>

<SelectItem value="Europe/Paris">Central European Time</SelectItem>

<SelectItem value="Asia/Tokyo">Japan Time</SelectItem>

<SelectItem value="Australia/Sydney">Australian Eastern Time</SelectItem>

</SelectContent>

</Select>

</div>

</div>

<div>

<Label htmlFor="goals">Personal Goals & Aspirations</Label>

<Textarea

id="goals"

value={formData.goals}

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

disabled={!isEditing}

placeholder="What are your main goals and aspirations for the next year?"

rows={3}

/>

</div>

<div>

<Label htmlFor="bio">Bio</Label>

<Textarea

id="bio"

value={formData.bio}

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

disabled={!isEditing}

placeholder="Tell us about yourself..."

rows={3}

/>

</div>

</CardContent>

</Card>

</div>

</div>

);

}