FRONTEND

Create portfolio

"use client"

import type React from "react"

import { useState, useEffect } from "react"

import { useRouter } from "next/navigation"

import Link from "next/link"

import { motion } from "framer-motion"

import {

  ArrowLeft,

  Loader2,

  Save,

  User,

  Briefcase,

  Mail,

  Phone,

  MapPin,

  Github,

  Linkedin,

  Twitter,

  Globe,

  GraduationCap,

  Building,

  FolderKanban,

  Palette,

  Sparkles,

} from "lucide-react"

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 { CardContent, CardFooter } from "@/components/ui/card"

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

import { RadioGroup, RadioGroupItem } from "@/components/ui/radio-group"

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

import { DashboardHeader } from "@/components/dashboard-header"

import { DashboardShell } from "@/components/dashboard-shell"

import { FileUpload } from "@/components/ui/file-upload"

export default function CreatePortfolioPage() {

  const [formData, setFormData] = useState({

    title: "",

    description: "",

    fullName: "",

    jobTitle: "",

    email: "",

    phone: "",

    location: "",

    bio: "",

    skills: "",

    github: "",

    linkedin: "",

    twitter: "",

    website: "",

    education: "",

    experience: "",

    projects: "",

    theme: "creative",

    profileImage: "",

    projectImages: ["", "", ""],

  })

  const [isLoading, setIsLoading] = useState(false)

  const [user, setUser] = useState<{ name?: string; email: string } | null>(null)

  const [activeTab, setActiveTab] = useState("basic")

  const router = useRouter()

  const { toast } = useToast()

  useEffect(() => {

    // Check if user is logged in

    const userData = localStorage.getItem("user")

    if (!userData) {

      router.push("/login")

      return

    }

    const parsedUser = JSON.parse(userData)

    setUser(parsedUser)

    // Pre-fill some fields if user data exists

    if (parsedUser.name) {

      setFormData((prev) => ({

        ...prev,

        fullName: parsedUser.name,

        email: parsedUser.email,

      }))

    }

  }, [router])

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

    const { name, value } = e.target

    setFormData((prev) => ({

      ...prev,

      [name]: value,

    }))

  }

  const handleThemeChange = (value: string) => {

    setFormData((prev) => ({

      ...prev,

      theme: value,

    }))

  }

  const handleProfileImageChange = (base64: string) => {

    setFormData((prev) => ({

      ...prev,

      profileImage: base64,

    }))

  }

  const handleProjectImageChange = (index: number, base64: string) => {

    setFormData((prev) => {

      const newProjectImages = [...prev.projectImages]

      newProjectImages[index] = base64

      return {

        ...prev,

        projectImages: newProjectImages,

      }

    })

  }

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

    e.preventDefault()

    // Validate form (basic validation)

    if (!formData.title || !formData.fullName || !formData.jobTitle) {

      toast({

        title: "Missing required fields",

        description: "Please fill in all required fields.",

        variant: "destructive",

      })

      return

    }

    setIsLoading(true)

    // Create portfolio object

    const newPortfolio = {

      id: Date.now().toString(),

      ...formData,

      createdAt: new Date().toISOString(),

    }

    // Get existing portfolios or initialize empty array

    const existingPortfolios = JSON.parse(localStorage.getItem("portfolios") || "[]")

    // Add new portfolio

    const updatedPortfolios = [...existingPortfolios, newPortfolio]

    // Save to localStorage

    localStorage.setItem("portfolios", JSON.stringify(updatedPortfolios))

    setTimeout(() => {

      toast({

        title: "Portfolio created",

        description: "Your portfolio has been created successfully.",

      })

      router.push(`/portfolio/${newPortfolio.id}`)

      setIsLoading(false)

    }, 1500)

  }

  const handleLogout = () => {

    localStorage.removeItem("user")

    toast({

      title: "Logged out",

      description: "You have been logged out successfully.",

    })

    router.push("/")

  }

  if (!user) {

    return (

      <div className="flex h-screen items-center justify-center bg-slate-950">

        <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2 border-teal-500"></div>

      </div>

    )

  }

  const glowVariants = {

    initial: { opacity: 0.5, scale: 1 },

    animate: {

      opacity: [0.5, 0.8, 0.5],

      scale: [1, 1.05, 1],

      transition: {

        duration: 3,

        repeat: Number.POSITIVE\_INFINITY,

        repeatType: "reverse",

      },

    },

  }

  const borderVariants = {

    initial: { backgroundPosition: "0% 50%" },

    animate: {

      backgroundPosition: ["0% 50%", "100% 50%", "0% 50%"],

      transition: {

        duration: 10,

        repeat: Number.POSITIVE\_INFINITY,

        ease: "linear",

      },

    },

  }

  return (

    <DashboardShell>

      <motion.div

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

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

        transition={{ duration: 0.5 }}

        className="absolute top-0 right-0 h-96 w-96 rounded-full blur-[100px] opacity-20 bg-purple-500 z-0"

      />

      <motion.div

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

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

        transition={{ duration: 0.5, delay: 0.2 }}

        className="absolute bottom-0 left-0 h-96 w-96 rounded-full blur-[100px] opacity-20 bg-teal-500 z-0"

      />

      <DashboardHeader

        heading="Create Portfolio"

        text="Fill in your details to create a new portfolio."

        user={user}

        onLogout={handleLogout}

      >

        <Button

          variant="outline"

          asChild

          className="relative backdrop-blur-sm border-slate-700/60 bg-slate-800/50 text-slate-300 hover:bg-slate-800/70 hover:text-slate-100"

        >

          <Link href="/dashboard" className="gap-2">

            <ArrowLeft className="h-4 w-4" /> Back to Dashboard

          </Link>

        </Button>

      </DashboardHeader>

      <div className="grid gap-8 relative z-10">

        <form onSubmit={handleSubmit}>

          <motion.div

            initial="initial"

            animate="animate"

            variants={glowVariants}

            className="absolute inset-0 -z-10 opacity-30 blur-[80px] bg-gradient-to-r from-teal-500/40 via-purple-500/40 to-teal-500/40"

          />

          <motion.div

            className="relative rounded-2xl overflow-hidden border border-white/10 shadow-2xl bg-black/20 backdrop-blur-xl"

            initial={{ opacity: 0, y: 20 }}

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

            transition={{ duration: 0.6 }}

          >

            <motion.div

              className="absolute inset-0 -z-10 bg-gradient-to-r from-teal-500/10 via-purple-500/10 to-teal-500/10"

              initial="initial"

              animate="animate"

              variants={borderVariants}

            />

            <motion.div

              className="absolute inset-0 border-2 border-transparent rounded-2xl"

              style={{

                background:

                  "linear-gradient(to right, rgba(56, 189, 248, 0.1), rgba(168, 85, 247, 0.1), rgba(56, 189, 248, 0.1))",

                backgroundClip: "padding-box, border-box",

                WebkitBackgroundClip: "padding-box, border-box",

                padding: "2px",

                mask: "linear-gradient(#fff 0 0) content-box, linear-gradient(#fff 0 0)",

                WebkitMask: "linear-gradient(#fff 0 0) content-box, linear-gradient(#fff 0 0)",

                maskComposite: "exclude",

                WebkitMaskComposite: "xor",

              }}

              initial="initial"

              animate="animate"

              variants={borderVariants}

            />

            {/\* Small Sparkles \*/}

            <motion.div

              className="absolute top-16 right-16 text-purple-500 opacity-70"

              animate={{

                scale: [1, 1.2, 1],

                opacity: [0.7, 1, 0.7],

                transition: { duration: 2, repeat: Number.POSITIVE\_INFINITY },

              }}

            >

              <Sparkles size={24} />

            </motion.div>

            <motion.div

              className="absolute bottom-16 left-16 text-teal-500 opacity-70"

              animate={{

                scale: [1, 1.2, 1],

                opacity: [0.7, 1, 0.7],

                transition: { duration: 2, repeat: Number.POSITIVE\_INFINITY, delay: 0.5 },

              }}

            >

              <Sparkles size={24} />

            </motion.div>

            <motion.div

              className="absolute top-1/2 left-8 text-teal-500 opacity-70"

              animate={{

                scale: [1, 1.2, 1],

                opacity: [0.7, 1, 0.7],

                transition: { duration: 2, repeat: Number.POSITIVE\_INFINITY, delay: 1 },

              }}

            >

              <Sparkles size={20} />

            </motion.div>

            <div className="relative pt-8 px-8 pb-2">

              <motion.div

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

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

                transition={{ duration: 0.5, delay: 0.2 }}

                className="flex flex-col gap-2"

              >

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

                  <span className="h-8 w-8 rounded-full bg-gradient-to-r from-teal-600 to-purple-600 flex items-center justify-center shadow-lg shadow-teal-500/20">

                    <Briefcase className="h-4 w-4 text-white" />

                  </span>

                  <h2 className="text-2xl font-bold bg-gradient-to-r from-white to-slate-300 bg-clip-text text-transparent">

                    Portfolio Information

                  </h2>

                </div>

                <p className="text-slate-400 ml-10">Fill in the details for your new portfolio</p>

              </motion.div>

            </div>

            <CardContent className="p-8">

              <Tabs value={activeTab} onValueChange={setActiveTab} className="w-full">

                <TabsList className="relative grid grid-cols-5 mb-8 bg-slate-800/40 backdrop-blur-md border border-white/10 rounded-xl p-1">

                  <motion.div

                    className="absolute inset-0 rounded-xl"

                    style={{

                      background: "radial-gradient(circle at center, rgba(45, 212, 191, 0.15), transparent 70%)",

                    }}

                    animate={{

                      scale: [1, 1.02, 1],

                      opacity: [0.5, 0.7, 0.5],

                    }}

                    transition={{ duration: 3, repeat: Number.POSITIVE\_INFINITY }}

                  />

                  <TabsTrigger

                    value="basic"

                    className="relative data-[state=active]:bg-gradient-to-r data-[state=active]:from-teal-500/20 data-[state=active]:to-purple-500/20 data-[state=active]:text-white data-[state=active]:backdrop-blur-md data-[state=active]:shadow-md data-[state=active]:shadow-teal-500/10 rounded-lg transition-all duration-300"

                  >

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

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

                      <span className="hidden sm:inline">Basic Info</span>

                    </div>

                  </TabsTrigger>

                  <TabsTrigger

                    value="contact"

                    className="relative data-[state=active]:bg-gradient-to-r data-[state=active]:from-teal-500/20 data-[state=active]:to-purple-500/20 data-[state=active]:text-white data-[state=active]:backdrop-blur-md data-[state=active]:shadow-md data-[state=active]:shadow-teal-500/10 rounded-lg transition-all duration-300"

                  >

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

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

                      <span className="hidden sm:inline">Contact</span>

                    </div>

                  </TabsTrigger>

                  <TabsTrigger

                    value="content"

                    className="relative data-[state=active]:bg-gradient-to-r data-[state=active]:from-teal-500/20 data-[state=active]:to-purple-500/20 data-[state=active]:text-white data-[state=active]:backdrop-blur-md data-[state=active]:shadow-md data-[state=active]:shadow-teal-500/10 rounded-lg transition-all duration-300"

                  >

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

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

                      <span className="hidden sm:inline">Content</span>

                    </div>

                  </TabsTrigger>

                  <TabsTrigger

                    value="images"

                    className="relative data-[state=active]:bg-gradient-to-r data-[state=active]:from-teal-500/20 data-[state=active]:to-purple-500/20 data-[state=active]:text-white data-[state=active]:backdrop-blur-md data-[state=active]:shadow-md data-[state=active]:shadow-teal-500/10 rounded-lg transition-all duration-300"

                  >

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

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

                      <span className="hidden sm:inline">Images</span>

                    </div>

                  </TabsTrigger>

                  <TabsTrigger

                    value="theme"

                    className="relative data-[state=active]:bg-gradient-to-r data-[state=active]:from-teal-500/20 data-[state=active]:to-purple-500/20 data-[state=active]:text-white data-[state=active]:backdrop-blur-md data-[state=active]:shadow-md data-[state=active]:shadow-teal-500/10 rounded-lg transition-all duration-300"

                  >

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

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

                      <span className="hidden sm:inline">Theme</span>

                    </div>

                  </TabsTrigger>

                </TabsList>

                <TabsContent value="basic" className="space-y-4">

                  <motion.div

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

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

                    transition={{ duration: 0.3 }}

                    className="space-y-6"

                  >

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

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

                        <Label htmlFor="title" className="text-slate-300 flex items-center gap-2">

                          <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                            1

                          </span>

                          Portfolio Title <span className="text-teal-400">\*</span>

                        </Label>

                        <div className="relative">

                          <Input

                            id="title"

                            name="title"

                            placeholder="My Professional Portfolio"

                            value={formData.title}

                            onChange={handleChange}

                            required

                            className="glassmorphism-input"

                          />

                          <Briefcase className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                        </div>

                      </div>

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

                        <Label htmlFor="fullName" className="text-slate-300 flex items-center gap-2">

                          <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                            2

                          </span>

                          Full Name <span className="text-teal-400">\*</span>

                        </Label>

                        <div className="relative">

                          <Input

                            id="fullName"

                            name="fullName"

                            placeholder="John Doe"

                            value={formData.fullName}

                            onChange={handleChange}

                            required

                            className="glassmorphism-input"

                          />

                          <User className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                        </div>

                      </div>

                    </div>

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

                      <Label htmlFor="description" className="text-slate-300 flex items-center gap-2">

                        <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                          3

                        </span>

                        Short Description

                      </Label>

                      <Textarea

                        id="description"

                        name="description"

                        placeholder="A brief description of your portfolio"

                        value={formData.description}

                        onChange={handleChange}

                        rows={2}

                        className="glassmorphism-textarea resize-none"

                      />

                    </div>

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

                      <Label htmlFor="jobTitle" className="text-slate-300 flex items-center gap-2">

                        <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                          4

                        </span>

                        Job Title <span className="text-teal-400">\*</span>

                      </Label>

                      <div className="relative">

                        <Input

                          id="jobTitle"

                          name="jobTitle"

                          placeholder="Frontend Developer"

                          value={formData.jobTitle}

                          onChange={handleChange}

                          required

                          className="glassmorphism-input"

                        />

                        <Briefcase className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                      </div>

                    </div>

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

                      <Label htmlFor="bio" className="text-slate-300 flex items-center gap-2">

                        <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                          5

                        </span>

                        Bio

                      </Label>

                      <Textarea

                        id="bio"

                        name="bio"

                        placeholder="Tell us about yourself"

                        value={formData.bio}

                        onChange={handleChange}

                        rows={5}

                        className="glassmorphism-textarea"

                      />

                    </div>

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

                      <Label htmlFor="skills" className="text-slate-300 flex items-center gap-2">

                        <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                          6

                        </span>

                        Skills (comma separated)

                      </Label>

                      <Input

                        id="skills"

                        name="skills"

                        placeholder="React, JavaScript, CSS, HTML"

                        value={formData.skills}

                        onChange={handleChange}

                        className="glassmorphism-input"

                      />

                    </div>

                  </motion.div>

                </TabsContent>

                <TabsContent value="contact" className="space-y-4">

                  <motion.div

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

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

                    transition={{ duration: 0.3 }}

                    className="space-y-6"

                  >

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

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

                        <Label htmlFor="email" className="text-slate-300 flex items-center gap-2">

                          <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                            @

                          </span>

                          Email

                        </Label>

                        <div className="relative">

                          <Input

                            id="email"

                            name="email"

                            type="email"

                            placeholder="you@example.com"

                            value={formData.email}

                            onChange={handleChange}

                            className="glassmorphism-input"

                          />

                          <Mail className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                        </div>

                      </div>

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

                        <Label htmlFor="phone" className="text-slate-300 flex items-center gap-2">

                          <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                            #

                          </span>

                          Phone

                        </Label>

                        <div className="relative">

                          <Input

                            id="phone"

                            name="phone"

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

                            value={formData.phone}

                            onChange={handleChange}

                            className="glassmorphism-input"

                          />

                          <Phone className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                        </div>

                      </div>

                    </div>

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

                      <Label htmlFor="location" className="text-slate-300 flex items-center gap-2">

                        <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                          <MapPin className="h-3 w-3" />

                        </span>

                        Location

                      </Label>

                      <div className="relative">

                        <Input

                          id="location"

                          name="location"

                          placeholder="New York, NY"

                          value={formData.location}

                          onChange={handleChange}

                          className="glassmorphism-input"

                        />

                        <MapPin className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                      </div>

                    </div>

                    <div className="relative p-6 border border-white/10 rounded-xl backdrop-blur-md bg-white/5">

                      <motion.div

                        className="absolute inset-0 rounded-xl opacity-20 bg-gradient-to-r from-teal-500/10 via-purple-500/10 to-teal-500/10"

                        animate={{

                          backgroundPosition: ["0% 0%", "100% 100%", "0% 0%"],

                        }}

                        transition={{ duration: 10, repeat: Number.POSITIVE\_INFINITY, ease: "linear" }}

                      />

                      <h3 className="text-lg font-medium text-white mb-4 flex items-center gap-2">

                        <Globe className="h-5 w-5 text-teal-400" />

                        Social Links

                      </h3>

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

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

                          <Label htmlFor="github" className="text-slate-300">

                            GitHub URL

                          </Label>

                          <div className="relative">

                            <Input

                              id="github"

                              name="github"

                              placeholder="https://github.com/yourusername"

                              value={formData.github}

                              onChange={handleChange}

                              className="glassmorphism-input"

                            />

                            <Github className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                          </div>

                        </div>

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

                          <Label htmlFor="linkedin" className="text-slate-300">

                            LinkedIn URL

                          </Label>

                          <div className="relative">

                            <Input

                              id="linkedin"

                              name="linkedin"

                              placeholder="https://linkedin.com/in/yourusername"

                              value={formData.linkedin}

                              onChange={handleChange}

                              className="glassmorphism-input"

                            />

                            <Linkedin className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                          </div>

                        </div>

                      </div>

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

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

                          <Label htmlFor="twitter" className="text-slate-300">

                            Twitter URL

                          </Label>

                          <div className="relative">

                            <Input

                              id="twitter"

                              name="twitter"

                              placeholder="https://twitter.com/yourusername"

                              value={formData.twitter}

                              onChange={handleChange}

                              className="glassmorphism-input"

                            />

                            <Twitter className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                          </div>

                        </div>

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

                          <Label htmlFor="website" className="text-slate-300">

                            Personal Website

                          </Label>

                          <div className="relative">

                            <Input

                              id="website"

                              name="website"

                              placeholder="https://yourwebsite.com"

                              value={formData.website}

                              onChange={handleChange}

                              className="glassmorphism-input"

                            />

                            <Globe className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                          </div>

                        </div>

                      </div>

                    </div>

                  </motion.div>

                </TabsContent>

                <TabsContent value="content" className="space-y-4">

                  <motion.div

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

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

                    transition={{ duration: 0.3 }}

                    className="space-y-6"

                  >

                    <div className="glassmorphism-panel">

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

                        <div className="h-8 w-8 rounded-full bg-gradient-to-r from-teal-500/40 to-teal-600/40 flex items-center justify-center backdrop-blur-md">

                          <GraduationCap className="h-4 w-4 text-white" />

                        </div>

                        <Label htmlFor="education" className="text-slate-100 text-lg">

                          Education

                        </Label>

                      </div>

                      <Textarea

                        id="education"

                        name="education"

                        placeholder="Your educational background"

                        value={formData.education}

                        onChange={handleChange}

                        rows={4}

                        className="glassmorphism-textarea"

                      />

                      <p className="text-sm text-slate-400 mt-2">Format: School Name, Degree, Year (one per line)</p>

                    </div>

                    <div className="glassmorphism-panel">

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

                        <div className="h-8 w-8 rounded-full bg-gradient-to-r from-purple-500/40 to-purple-600/40 flex items-center justify-center backdrop-blur-md">

                          <Building className="h-4 w-4 text-white" />

                        </div>

                        <Label htmlFor="experience" className="text-slate-100 text-lg">

                          Work Experience

                        </Label>

                      </div>

                      <Textarea

                        id="experience"

                        name="experience"

                        placeholder="Your work experience"

                        value={formData.experience}

                        onChange={handleChange}

                        rows={6}

                        className="glassmorphism-textarea"

                      />

                      <p className="text-sm text-slate-400 mt-2">

                        Format: Company, Position, Duration, Description (one entry per paragraph)

                      </p>

                    </div>

                    <div className="glassmorphism-panel">

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

                        <div className="h-8 w-8 rounded-full bg-gradient-to-r from-teal-500/40 to-purple-600/40 flex items-center justify-center backdrop-blur-md">

                          <FolderKanban className="h-4 w-4 text-white" />

                        </div>

                        <Label htmlFor="projects" className="text-slate-100 text-lg">

                          Projects

                        </Label>

                      </div>

                      <Textarea

                        id="projects"

                        name="projects"

                        placeholder="Your projects"

                        value={formData.projects}

                        onChange={handleChange}

                        rows={6}

                        className="glassmorphism-textarea"

                      />

                      <p className="text-sm text-slate-400 mt-2">

                        Format: Project Name, Technologies, Description (one project per paragraph)

                      </p>

                    </div>

                  </motion.div>

                </TabsContent>

                <TabsContent value="images" className="space-y-4">

                  <motion.div

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

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

                    transition={{ duration: 0.3 }}

                    className="space-y-6"

                  >

                    <div className="glassmorphism-panel">

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

                        <div className="h-8 w-8 rounded-full bg-gradient-to-r from-teal-500/40 to-teal-600/40 flex items-center justify-center backdrop-blur-md">

                          <User className="h-4 w-4 text-white" />

                        </div>

                        <Label className="text-slate-100 text-lg">Profile Photo</Label>

                      </div>

                      <FileUpload

                        label="Upload your profile photo"

                        onChange={handleProfileImageChange}

                        value={formData.profileImage}

                        accept="image/\*"

                        maxSizeMB={2}

                      />

                      <p className="text-sm text-slate-400 mt-2">

                        This image will be displayed as your profile photo. Square images work best.

                      </p>

                    </div>

                    <div className="glassmorphism-panel">

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

                        <div className="h-8 w-8 rounded-full bg-gradient-to-r from-purple-500/40 to-purple-600/40 flex items-center justify-center backdrop-blur-md">

                          <FolderKanban className="h-4 w-4 text-white" />

                        </div>

                        <Label className="text-slate-100 text-lg">Project Images (Optional)</Label>

                      </div>

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

                        {formData.projectImages.map((image, index) => (

                          <motion.div

                            key={index}

                            className="glassmorphism-card rounded-xl p-4"

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

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

                            transition={{ duration: 0.3, delay: index \* 0.1 }}

                          >

                            <FileUpload

                              label={`Project Image ${index + 1}`}

                              onChange={(base64) => handleProjectImageChange(index, base64)}

                              value={image}

                              accept="image/\*"

                              maxSizeMB={2}

                            />

                          </motion.div>

                        ))}

                      </div>

                      <p className="text-sm text-slate-400 mt-2">

                        These images will be displayed in your projects section.

                      </p>

                    </div>

                  </motion.div>

                </TabsContent>

                <TabsContent value="theme" className="space-y-4">

                  <motion.div

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

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

                    transition={{ duration: 0.3 }}

                    className="space-y-6"

                  >

                    <div className="glassmorphism-panel">

                      <Label className="text-slate-100 text-lg flex items-center gap-2 mb-6">

                        <div className="h-8 w-8 rounded-full bg-gradient-to-r from-teal-500/40 to-purple-600/40 flex items-center justify-center backdrop-blur-md">

                          <Palette className="h-4 w-4 text-white" />

                        </div>

                        Select a Theme

                      </Label>

                      <RadioGroup

                        value={formData.theme}

                        onValueChange={handleThemeChange}

                        className="grid grid-cols-1 md:grid-cols-3 gap-4"

                      >

                        <motion.div

                          whileHover={{ scale: 1.02, transition: { duration: 0.2 } }}

                          whileTap={{ scale: 0.98 }}

                        >

                          <RadioGroupItem value="creative" id="creative" className="peer sr-only" />

                          <Label

                            htmlFor="creative"

                            className="group flex flex-col items-center justify-between rounded-xl border border-slate-700/50 bg-black/30 backdrop-blur-md p-4 hover:bg-black/40 hover:border-teal-500/50 hover:text-slate-100 peer-data-[state=checked]:border-teal-500 peer-data-[state=checked]:ring-1 peer-data-[state=checked]:ring-teal-500/50 [&:has([data-state=checked])]:border-teal-500 transition-all duration-200"

                          >

                            <div className="mb-3 h-16 w-16 rounded-md bg-gradient-to-br from-purple-500 to-pink-500 flex items-center justify-center text-white font-bold group-hover:shadow-md group-hover:shadow-purple-500/20 transition-all">

                              C

                            </div>

                            <div className="space-y-1 text-center">

                              <p className="text-sm font-medium leading-none text-slate-200">Creative</p>

                              <p className="text-sm text-slate-400">Vibrant colors with modern animations</p>

                            </div>

                          </Label>

                        </motion.div>

                        <motion.div

                          whileHover={{ scale: 1.02, transition: { duration: 0.2 } }}

                          whileTap={{ scale: 0.98 }}

                        >

                          <RadioGroupItem value="minimal" id="minimal" className="peer sr-only" />

                          <Label

                            htmlFor="minimal"

                            className="group flex flex-col items-center justify-between rounded-xl border border-slate-700/50 bg-black/30 backdrop-blur-md p-4 hover:bg-black/40 hover:border-teal-500/50 hover:text-slate-100 peer-data-[state=checked]:border-teal-500 peer-data-[state=checked]:ring-1 peer-data-[state=checked]:ring-teal-500/50 [&:has([data-state=checked])]:border-teal-500 transition-all duration-200"

                          >

                            <div className="mb-3 h-16 w-16 rounded-md bg-gray-700 flex items-center justify-center text-white font-bold group-hover:shadow-md group-hover:shadow-gray-500/20 transition-all">

                              M

                            </div>

                            <div className="space-y-1 text-center">

                              <p className="text-sm font-medium leading-none text-slate-200">Minimal</p>

                              <p className="text-sm text-slate-400">Clean and simple design</p>

                            </div>

                          </Label>

                        </motion.div>

                        <motion.div

                          whileHover={{ scale: 1.02, transition: { duration: 0.2 } }}

                          whileTap={{ scale: 0.98 }}

                        >

                          <RadioGroupItem value="modern" id="modern" className="peer sr-only" />

                          <Label

                            htmlFor="modern"

                            className="group flex flex-col items-center justify-between rounded-xl border border-slate-700/50 bg-black/30 backdrop-blur-md p-4 hover:bg-black/40 hover:border-teal-500/50 hover:text-slate-100 peer-data-[state=checked]:border-teal-500 peer-data-[state=checked]:ring-1 peer-data-[state=checked]:ring-teal-500/50 [&:has([data-state=checked])]:border-teal-500 transition-all duration-200"

                          >

                            <div className="mb-3 h-16 w-16 rounded-md bg-gradient-to-r from-blue-500 to-cyan-400 flex items-center justify-center text-white font-bold group-hover:shadow-md group-hover:shadow-blue-500/20 transition-all">

                              M

                            </div>

                            <div className="space-y-1 text-center">

                              <p className="text-sm font-medium leading-none text-slate-200">Modern</p>

                              <p className="text-sm text-slate-400">Professional with subtle animations</p>

                            </div>

                          </Label>

                        </motion.div>

                      </RadioGroup>

                    </div>

                    <div className="glassmorphism-panel mt-8">

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

                        <div className="h-8 w-8 rounded-full bg-gradient-to-r from-teal-500/40 to-purple-600/40 flex items-center justify-center backdrop-blur-md">

                          <Palette className="h-4 w-4 text-white" />

                        </div>

                        <h3 className="font-medium text-slate-100">Theme Preview</h3>

                      </div>

                      <motion.div

                        className="rounded-xl overflow-hidden border border-white/10"

                        whileHover={{ scale: 1.01 }}

                        transition={{ duration: 0.3 }}

                      >

                        <div

                          className={`h-48 flex items-center justify-center overflow-hidden relative ${

                            formData.theme === "creative"

                              ? "bg-gradient-to-br from-purple-500 to-pink-500 text-white"

                              : formData.theme === "minimal"

                                ? "bg-gray-800 text-white border border-gray-700"

                                : "bg-gradient-to-r from-blue-500 to-cyan-400 text-white"

                          }`}

                        >

                          {formData.profileImage && (

                            <img

                              src={formData.profileImage || "/placeholder.svg"}

                              alt="Preview"

                              className="absolute inset-0 w-full h-full object-cover opacity-20 scale-105 blur-sm"

                            />

                          )}

                          <div className="absolute inset-0 bg-black/30 backdrop-blur-sm" />

                          <div className="relative z-10 flex flex-col items-center gap-4">

                            {formData.profileImage && (

                              <div className="w-20 h-20 rounded-full overflow-hidden border-2 border-white/50">

                                <img

                                  src={formData.profileImage || "/placeholder.svg"}

                                  alt="Preview"

                                  className="w-full h-full object-cover"

                                />

                              </div>

                            )}

                            <motion.div

                              initial={{ opacity: 0 }}

                              animate={{ opacity: 1 }}

                              transition={{ duration: 0.5 }}

                            >

                              <p className="font-bold text-xl">{formData.fullName || "Your Name"}</p>

                              <p className="text-center opacity-80">{formData.jobTitle || "Your Profession"}</p>

                            </motion.div>

                          </div>

                        </div>

                      </motion.div>

                      <p className="text-sm text-slate-400 mt-4">

                        This is a simplified preview. The actual portfolio will have a complete design with all your

                        information.

                      </p>

                    </div>

                  </motion.div>

                </TabsContent>

              </Tabs>

            </CardContent>

            <CardFooter className="flex justify-between p-8">

              <Button

                type="button"

                variant="outline"

                onClick={() => {

                  const prevTab = {

                    basic: "basic",

                    contact: "basic",

                    content: "contact",

                    images: "content",

                    theme: "images",

                  }[activeTab]

                  if (prevTab !== activeTab) {

                    setActiveTab(prevTab)

                  }

                }}

                disabled={activeTab === "basic"}

                className="glassmorphism-button-outline"

              >

                Previous

              </Button>

              {activeTab !== "theme" ? (

                <Button

                  type="button"

                  onClick={() => {

                    const nextTab = {

                      basic: "contact",

                      contact: "content",

                      content: "images",

                      images: "theme",

                      theme: "theme",

                    }[activeTab]

                    setActiveTab(nextTab)

                  }}

                  className="glassmorphism-button"

                >

                  Next

                </Button>

              ) : (

                <motion.div whileHover={{ scale: 1.03 }} whileTap={{ scale: 0.97 }}>

                  <Button

                    type="submit"

                    disabled={isLoading}

                    className="relative group glassmorphism-button overflow-hidden"

                  >

                    <motion.div

                      className="absolute inset-0 bg-gradient-to-r from-teal-500/20 via-purple-500/20 to-teal-500/20"

                      animate={{

                        x: ["-100%", "100%"],

                      }}

                      transition={{ duration: 3, repeat: Number.POSITIVE\_INFINITY, ease: "linear" }}

                    />

                    <span className="flex items-center gap-2 relative z-10">

                      {isLoading ? (

                        <>

                          <Loader2 className="h-4 w-4 animate-spin" /> Creating...

                        </>

                      ) : (

                        <>

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

                        </>

                      )}

                    </span>

                  </Button>

                </motion.div>

              )}

            </CardFooter>

          </motion.div>

        </form>

      </div>

    </DashboardShell>

  )

}

Dashboard

"use client"

import { useEffect, useState } from "react"

import { useRouter } from "next/navigation"

import Link from "next/link"

import { motion } from "framer-motion"

import { Plus, Settings, LogOut, User, Palette, Briefcase } from "lucide-react"

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

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

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

import { DashboardHeader } from "@/components/dashboard-header"

import { DashboardShell } from "@/components/dashboard-shell"

import { EmptyPlaceholder } from "@/components/empty-placeholder"

interface Portfolio {

  id: string

  title: string

  description: string

  theme: string

  createdAt: string

}

export default function DashboardPage() {

  const [portfolios, setPortfolios] = useState<Portfolio[]>([])

  const [isLoading, setIsLoading] = useState(true)

  const [user, setUser] = useState<{ name?: string; email: string } | null>(null)

  const router = useRouter()

  const { toast } = useToast()

  useEffect(() => {

    // Check if user is logged in

    const userData = localStorage.getItem("user")

    if (!userData) {

      router.push("/login")

      return

    }

    setUser(JSON.parse(userData))

    // Load portfolios from localStorage

    const storedPortfolios = localStorage.getItem("portfolios")

    if (storedPortfolios) {

      setPortfolios(JSON.parse(storedPortfolios))

    }

    setIsLoading(false)

  }, [router])

  const handleLogout = () => {

    localStorage.removeItem("user")

    toast({

      title: "Logged out",

      description: "You have been logged out successfully.",

    })

    router.push("/")

  }

  const handleDeletePortfolio = (id: string) => {

    const updatedPortfolios = portfolios.filter((portfolio) => portfolio.id !== id)

    setPortfolios(updatedPortfolios)

    localStorage.setItem("portfolios", JSON.stringify(updatedPortfolios))

    toast({

      title: "Portfolio deleted",

      description: "Your portfolio has been deleted successfully.",

    })

  }

  if (isLoading) {

    return (

      <div className="flex h-screen items-center justify-center bg-slate-950">

        <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2 border-teal-500"></div>

      </div>

    )

  }

  return (

    <DashboardShell>

      <DashboardHeader

        heading="Portfolios"

        text="Create and manage your professional portfolios."

        user={user}

        onLogout={handleLogout}

      >

        <Button

          asChild

          className="bg-gradient-to-r from-teal-500 to-teal-600 hover:from-teal-600 hover:to-teal-700 text-white border-0"

        >

          <Link href="/create-portfolio" className="gap-2">

            <Plus className="h-4 w-4" /> New Portfolio

          </Link>

        </Button>

      </DashboardHeader>

      <div className="grid gap-8">

        {portfolios.length === 0 ? (

          <EmptyPlaceholder>

            <EmptyPlaceholder.Icon>

              <Briefcase className="h-8 w-8 text-slate-400" />

            </EmptyPlaceholder.Icon>

            <EmptyPlaceholder.Title>No portfolios created</EmptyPlaceholder.Title>

            <EmptyPlaceholder.Description>

              You haven&apos;t created any portfolios yet. Start by creating your first portfolio.

            </EmptyPlaceholder.Description>

            <Button

              asChild

              className="bg-gradient-to-r from-teal-500 to-teal-600 hover:from-teal-600 hover:to-teal-700 text-white border-0"

            >

              <Link href="/create-portfolio" className="gap-2">

                <Plus className="h-4 w-4" /> Create Portfolio

              </Link>

            </Button>

          </EmptyPlaceholder>

        ) : (

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

            {portfolios.map((portfolio, index) => (

              <motion.div

                key={portfolio.id}

                initial={{ opacity: 0, y: 20 }}

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

                transition={{ delay: index \* 0.1, duration: 0.5 }}

              >

                <Card className="overflow-hidden bg-slate-800 border-slate-700 text-slate-200">

                  <div

                    className={`h-3 ${

                      portfolio.theme === "modern"

                        ? "bg-blue-500"

                        : portfolio.theme === "minimal"

                          ? "bg-gray-500"

                          : "bg-purple-500"

                    }`}

                  />

                  <CardHeader>

                    <CardTitle className="text-slate-100">{portfolio.title}</CardTitle>

                    <CardDescription className="line-clamp-2 text-slate-400">{portfolio.description}</CardDescription>

                  </CardHeader>

                  <CardContent>

                    <div className="flex items-center gap-2 text-sm text-slate-400">

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

                      <span className="capitalize">{portfolio.theme} Theme</span>

                    </div>

                    <div className="flex items-center gap-2 text-sm text-slate-400 mt-2">

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

                      <span>Created on {new Date(portfolio.createdAt).toLocaleDateString()}</span>

                    </div>

                  </CardContent>

                  <CardFooter className="flex justify-between">

                    <Button

                      variant="outline"

                      size="sm"

                      asChild

                      className="border-slate-700 text-slate-300 hover:bg-slate-700 hover:text-slate-100"

                    >

                      <Link href={`/portfolio/${portfolio.id}`}>View</Link>

                    </Button>

                    <div className="flex gap-2">

                      <Button

                        variant="outline"

                        size="sm"

                        asChild

                        className="border-slate-700 text-slate-300 hover:bg-slate-700 hover:text-slate-100"

                      >

                        <Link href={`/edit-portfolio/${portfolio.id}`}>

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

                        </Link>

                      </Button>

                      <Button variant="destructive" size="sm" onClick={() => handleDeletePortfolio(portfolio.id)}>

                        <LogOut className="h-4 w-4 rotate-180" />

                      </Button>

                    </div>

                  </CardFooter>

                </Card>

              </motion.div>

            ))}

          </div>

        )}

      </div>

    </DashboardShell>

  )

}

Edit portfolio

"use client"

import type React from "react"

import { useState, useEffect } from "react"

import { useRouter, useParams } from "next/navigation"

import Link from "next/link"

import { motion } from "framer-motion"

import {

  ArrowLeft,

  Loader2,

  Save,

  User,

  Briefcase,

  Mail,

  Phone,

  MapPin,

  Github,

  Linkedin,

  Twitter,

  Globe,

  GraduationCap,

  Building,

  FolderKanban,

  Palette,

  Sparkles,

} from "lucide-react"

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 { CardContent, CardFooter } from "@/components/ui/card"

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

import { RadioGroup, RadioGroupItem } from "@/components/ui/radio-group"

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

import { DashboardHeader } from "@/components/dashboard-header"

import { DashboardShell } from "@/components/dashboard-shell"

import { FileUpload } from "@/components/ui/file-upload"

interface Portfolio {

  id: string

  title: string

  description: string

  fullName: string

  jobTitle: string

  email: string

  phone: string

  location: string

  bio: string

  skills: string

  github: string

  linkedin: string

  twitter: string

  website: string

  education: string

  experience: string

  projects: string

  theme: string

  createdAt: string

  profileImage?: string

  projectImages?: string[]

}

export default function EditPortfolioPage() {

  const [formData, setFormData] = useState<Portfolio>({

    id: "",

    title: "",

    description: "",

    fullName: "",

    jobTitle: "",

    email: "",

    phone: "",

    location: "",

    bio: "",

    skills: "",

    github: "",

    linkedin: "",

    twitter: "",

    website: "",

    education: "",

    experience: "",

    projects: "",

    theme: "creative",

    createdAt: "",

    profileImage: "",

    projectImages: ["", "", ""],

  })

  const [isLoading, setIsLoading] = useState(false)

  const [user, setUser] = useState<{ name?: string; email: string } | null>(null)

  const [activeTab, setActiveTab] = useState("basic")

  const router = useRouter()

  const params = useParams()

  const { toast } = useToast()

  const portfolioId = params.id as string

  useEffect(() => {

    // Check if user is logged in

    const userData = localStorage.getItem("user")

    if (!userData) {

      router.push("/login")

      return

    }

    setUser(JSON.parse(userData))

    // Load portfolio data

    const storedPortfolios = localStorage.getItem("portfolios")

    if (storedPortfolios) {

      const portfolios = JSON.parse(storedPortfolios)

      const foundPortfolio = portfolios.find((p: Portfolio) => p.id === portfolioId)

      if (foundPortfolio) {

        // Ensure projectImages array exists

        if (!foundPortfolio.projectImages) {

          foundPortfolio.projectImages = ["", "", ""]

        }

        setFormData(foundPortfolio)

      } else {

        toast({

          title: "Portfolio not found",

          description: "The requested portfolio could not be found.",

          variant: "destructive",

        })

        router.push("/dashboard")

      }

    }

  }, [portfolioId, router, toast])

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

    const { name, value } = e.target

    setFormData((prev) => ({

      ...prev,

      [name]: value,

    }))

  }

  const handleThemeChange = (value: string) => {

    setFormData((prev) => ({

      ...prev,

      theme: value,

    }))

  }

  const handleProfileImageChange = (base64: string) => {

    setFormData((prev) => ({

      ...prev,

      profileImage: base64,

    }))

  }

  const handleProjectImageChange = (index: number, base64: string) => {

    setFormData((prev) => {

      const newProjectImages = [...(prev.projectImages || ["", "", ""])]

      newProjectImages[index] = base64

      return {

        ...prev,

        projectImages: newProjectImages,

      }

    })

  }

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

    e.preventDefault()

    // Validate form (basic validation)

    if (!formData.title || !formData.fullName || !formData.jobTitle) {

      toast({

        title: "Missing required fields",

        description: "Please fill in all required fields.",

        variant: "destructive",

      })

      return

    }

    setIsLoading(true)

    // Get existing portfolios

    const storedPortfolios = localStorage.getItem("portfolios")

    if (storedPortfolios) {

      const portfolios = JSON.parse(storedPortfolios)

      // Update the portfolio

      const updatedPortfolios = portfolios.map((p: Portfolio) => {

        if (p.id === portfolioId) {

          return { ...formData }

        }

        return p

      })

      // Save to localStorage

      localStorage.setItem("portfolios", JSON.stringify(updatedPortfolios))

      setTimeout(() => {

        toast({

          title: "Portfolio updated",

          description: "Your portfolio has been updated successfully.",

        })

        router.push(`/portfolio/${portfolioId}`)

        setIsLoading(false)

      }, 1500)

    }

  }

  const handleLogout = () => {

    localStorage.removeItem("user")

    toast({

      title: "Logged out",

      description: "You have been logged out successfully.",

    })

    router.push("/")

  }

  if (!formData.id) {

    return (

      <div className="flex h-screen items-center justify-center bg-slate-950">

        <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2 border-teal-500"></div>

      </div>

    )

  }

  const glowVariants = {

    initial: { opacity: 0.5, scale: 1 },

    animate: {

      opacity: [0.5, 0.8, 0.5],

      scale: [1, 1.05, 1],

      transition: {

        duration: 3,

        repeat: Number.POSITIVE\_INFINITY,

        repeatType: "reverse",

      },

    },

  }

  const borderVariants = {

    initial: { backgroundPosition: "0% 50%" },

    animate: {

      backgroundPosition: ["0% 50%", "100% 50%", "0% 50%"],

      transition: {

        duration: 10,

        repeat: Number.POSITIVE\_INFINITY,

        ease: "linear",

      },

    },

  }

  return (

    <DashboardShell>

      <motion.div

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

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

        transition={{ duration: 0.5 }}

        className="absolute top-0 right-0 h-96 w-96 rounded-full blur-[100px] opacity-20 bg-purple-500 z-0"

      />

      <motion.div

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

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

        transition={{ duration: 0.5, delay: 0.2 }}

        className="absolute bottom-0 left-0 h-96 w-96 rounded-full blur-[100px] opacity-20 bg-teal-500 z-0"

      />

      <DashboardHeader

        heading="Edit Portfolio"

        text="Update your portfolio information."

        user={user}

        onLogout={handleLogout}

      >

        <Button

          variant="outline"

          asChild

          className="relative backdrop-blur-sm border-slate-700/60 bg-slate-800/50 text-slate-300 hover:bg-slate-800/70 hover:text-slate-100"

        >

          <Link href={`/portfolio/${portfolioId}`} className="gap-2">

            <ArrowLeft className="h-4 w-4" /> Back to Portfolio

          </Link>

        </Button>

      </DashboardHeader>

      <div className="grid gap-8 relative z-10">

        <form onSubmit={handleSubmit}>

          <motion.div

            initial="initial"

            animate="animate"

            variants={glowVariants}

            className="absolute inset-0 -z-10 opacity-30 blur-[80px] bg-gradient-to-r from-teal-500/40 via-purple-500/40 to-teal-500/40"

          />

          <motion.div

            className="relative rounded-2xl overflow-hidden border border-white/10 shadow-2xl bg-black/20 backdrop-blur-xl"

            initial={{ opacity: 0, y: 20 }}

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

            transition={{ duration: 0.6 }}

          >

            <motion.div

              className="absolute inset-0 -z-10 bg-gradient-to-r from-teal-500/10 via-purple-500/10 to-teal-500/10"

              initial="initial"

              animate="animate"

              variants={borderVariants}

            />

            <motion.div

              className="absolute inset-0 border-2 border-transparent rounded-2xl"

              style={{

                background:

                  "linear-gradient(to right, rgba(56, 189, 248, 0.1), rgba(168, 85, 247, 0.1), rgba(56, 189, 248, 0.1))",

                backgroundClip: "padding-box, border-box",

                WebkitBackgroundClip: "padding-box, border-box",

                padding: "2px",

                mask: "linear-gradient(#fff 0 0) content-box, linear-gradient(#fff 0 0)",

                WebkitMask: "linear-gradient(#fff 0 0) content-box, linear-gradient(#fff 0 0)",

                maskComposite: "exclude",

                WebkitMaskComposite: "xor",

              }}

              initial="initial"

              animate="animate"

              variants={borderVariants}

            />

            {/\* Small Sparkles \*/}

            <motion.div

              className="absolute top-16 right-16 text-purple-500 opacity-70"

              animate={{

                scale: [1, 1.2, 1],

                opacity: [0.7, 1, 0.7],

                transition: { duration: 2, repeat: Number.POSITIVE\_INFINITY },

              }}

            >

              <Sparkles size={24} />

            </motion.div>

            <motion.div

              className="absolute bottom-16 left-16 text-teal-500 opacity-70"

              animate={{

                scale: [1, 1.2, 1],

                opacity: [0.7, 1, 0.7],

                transition: { duration: 2, repeat: Number.POSITIVE\_INFINITY, delay: 0.5 },

              }}

            >

              <Sparkles size={24} />

            </motion.div>

            <motion.div

              className="absolute top-1/2 left-8 text-teal-500 opacity-70"

              animate={{

                scale: [1, 1.2, 1],

                opacity: [0.7, 1, 0.7],

                transition: { duration: 2, repeat: Number.POSITIVE\_INFINITY, delay: 1 },

              }}

            >

              <Sparkles size={20} />

            </motion.div>

            <div className="relative pt-8 px-8 pb-2">

              <motion.div

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

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

                transition={{ duration: 0.5, delay: 0.2 }}

                className="flex flex-col gap-2"

              >

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

                  <span className="h-8 w-8 rounded-full bg-gradient-to-r from-teal-600 to-purple-600 flex items-center justify-center shadow-lg shadow-teal-500/20">

                    <Briefcase className="h-4 w-4 text-white" />

                  </span>

                  <h2 className="text-2xl font-bold bg-gradient-to-r from-white to-slate-300 bg-clip-text text-transparent">

                    Portfolio Information

                  </h2>

                </div>

                <p className="text-slate-400 ml-10">Update the details for your portfolio</p>

              </motion.div>

            </div>

            <CardContent className="p-8">

              <Tabs value={activeTab} onValueChange={setActiveTab} className="w-full">

                <TabsList className="relative grid grid-cols-5 mb-8 bg-slate-800/40 backdrop-blur-md border border-white/10 rounded-xl p-1">

                  <motion.div

                    className="absolute inset-0 rounded-xl"

                    style={{

                      background: "radial-gradient(circle at center, rgba(45, 212, 191, 0.15), transparent 70%)",

                    }}

                    animate={{

                      scale: [1, 1.02, 1],

                      opacity: [0.5, 0.7, 0.5],

                    }}

                    transition={{ duration: 3, repeat: Number.POSITIVE\_INFINITY }}

                  />

                  <TabsTrigger

                    value="basic"

                    className="relative data-[state=active]:bg-gradient-to-r data-[state=active]:from-teal-500/20 data-[state=active]:to-purple-500/20 data-[state=active]:text-white data-[state=active]:backdrop-blur-md data-[state=active]:shadow-md data-[state=active]:shadow-teal-500/10 rounded-lg transition-all duration-300"

                  >

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

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

                      <span className="hidden sm:inline">Basic Info</span>

                    </div>

                  </TabsTrigger>

                  <TabsTrigger

                    value="contact"

                    className="relative data-[state=active]:bg-gradient-to-r data-[state=active]:from-teal-500/20 data-[state=active]:to-purple-500/20 data-[state=active]:text-white data-[state=active]:backdrop-blur-md data-[state=active]:shadow-md data-[state=active]:shadow-teal-500/10 rounded-lg transition-all duration-300"

                  >

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

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

                      <span className="hidden sm:inline">Contact</span>

                    </div>

                  </TabsTrigger>

                  <TabsTrigger

                    value="content"

                    className="relative data-[state=active]:bg-gradient-to-r data-[state=active]:from-teal-500/20 data-[state=active]:to-purple-500/20 data-[state=active]:text-white data-[state=active]:backdrop-blur-md data-[state=active]:shadow-md data-[state=active]:shadow-teal-500/10 rounded-lg transition-all duration-300"

                  >

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

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

                      <span className="hidden sm:inline">Content</span>

                    </div>

                  </TabsTrigger>

                  <TabsTrigger

                    value="images"

                    className="relative data-[state=active]:bg-gradient-to-r data-[state=active]:from-teal-500/20 data-[state=active]:to-purple-500/20 data-[state=active]:text-white data-[state=active]:backdrop-blur-md data-[state=active]:shadow-md data-[state=active]:shadow-teal-500/10 rounded-lg transition-all duration-300"

                  >

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

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

                      <span className="hidden sm:inline">Images</span>

                    </div>

                  </TabsTrigger>

                  <TabsTrigger

                    value="theme"

                    className="relative data-[state=active]:bg-gradient-to-r data-[state=active]:from-teal-500/20 data-[state=active]:to-purple-500/20 data-[state=active]:text-white data-[state=active]:backdrop-blur-md data-[state=active]:shadow-md data-[state=active]:shadow-teal-500/10 rounded-lg transition-all duration-300"

                  >

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

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

                      <span className="hidden sm:inline">Theme</span>

                    </div>

                  </TabsTrigger>

                </TabsList>

                <TabsContent value="basic" className="space-y-4">

                  <motion.div

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

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

                    transition={{ duration: 0.3 }}

                    className="space-y-6"

                  >

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

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

                        <Label htmlFor="title" className="text-slate-300 flex items-center gap-2">

                          <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                            1

                          </span>

                          Portfolio Title <span className="text-teal-400">\*</span>

                        </Label>

                        <div className="relative">

                          <Input

                            id="title"

                            name="title"

                            placeholder="My Professional Portfolio"

                            value={formData.title}

                            onChange={handleChange}

                            required

                            className="glassmorphism-input"

                          />

                          <Briefcase className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                        </div>

                      </div>

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

                        <Label htmlFor="fullName" className="text-slate-300 flex items-center gap-2">

                          <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                            2

                          </span>

                          Full Name <span className="text-teal-400">\*</span>

                        </Label>

                        <div className="relative">

                          <Input

                            id="fullName"

                            name="fullName"

                            placeholder="John Doe"

                            value={formData.fullName}

                            onChange={handleChange}

                            required

                            className="glassmorphism-input"

                          />

                          <User className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                        </div>

                      </div>

                    </div>

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

                      <Label htmlFor="description" className="text-slate-300 flex items-center gap-2">

                        <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                          3

                        </span>

                        Short Description

                      </Label>

                      <Textarea

                        id="description"

                        name="description"

                        placeholder="A brief description of your portfolio"

                        value={formData.description}

                        onChange={handleChange}

                        rows={2}

                        className="glassmorphism-textarea resize-none"

                      />

                    </div>

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

                      <Label htmlFor="jobTitle" className="text-slate-300 flex items-center gap-2">

                        <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                          4

                        </span>

                        Job Title <span className="text-teal-400">\*</span>

                      </Label>

                      <div className="relative">

                        <Input

                          id="jobTitle"

                          name="jobTitle"

                          placeholder="Frontend Developer"

                          value={formData.jobTitle}

                          onChange={handleChange}

                          required

                          className="glassmorphism-input"

                        />

                        <Briefcase className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                      </div>

                    </div>

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

                      <Label htmlFor="bio" className="text-slate-300 flex items-center gap-2">

                        <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                          5

                        </span>

                        Bio

                      </Label>

                      <Textarea

                        id="bio"

                        name="bio"

                        placeholder="Tell us about yourself"

                        value={formData.bio}

                        onChange={handleChange}

                        rows={5}

                        className="glassmorphism-textarea"

                      />

                    </div>

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

                      <Label htmlFor="skills" className="text-slate-300 flex items-center gap-2">

                        <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                          6

                        </span>

                        Skills (comma separated)

                      </Label>

                      <Input

                        id="skills"

                        name="skills"

                        placeholder="React, JavaScript, CSS, HTML"

                        value={formData.skills}

                        onChange={handleChange}

                        className="glassmorphism-input"

                      />

                    </div>

                  </motion.div>

                </TabsContent>

                <TabsContent value="contact" className="space-y-4">

                  <motion.div

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

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

                    transition={{ duration: 0.3 }}

                    className="space-y-6"

                  >

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

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

                        <Label htmlFor="email" className="text-slate-300 flex items-center gap-2">

                          <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                            @

                          </span>

                          Email

                        </Label>

                        <div className="relative">

                          <Input

                            id="email"

                            name="email"

                            type="email"

                            placeholder="you@example.com"

                            value={formData.email}

                            onChange={handleChange}

                            className="glassmorphism-input"

                          />

                          <Mail className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                        </div>

                      </div>

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

                        <Label htmlFor="phone" className="text-slate-300 flex items-center gap-2">

                          <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                            #

                          </span>

                          Phone

                        </Label>

                        <div className="relative">

                          <Input

                            id="phone"

                            name="phone"

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

                            value={formData.phone}

                            onChange={handleChange}

                            className="glassmorphism-input"

                          />

                          <Phone className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                        </div>

                      </div>

                    </div>

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

                      <Label htmlFor="location" className="text-slate-300 flex items-center gap-2">

                        <span className="h-5 w-5 rounded-full bg-gradient-to-r from-teal-500/20 to-teal-500/30 flex items-center justify-center text-xs font-bold backdrop-blur-sm">

                          <MapPin className="h-3 w-3" />

                        </span>

                        Location

                      </Label>

                      <div className="relative">

                        <Input

                          id="location"

                          name="location"

                          placeholder="New York, NY"

                          value={formData.location}

                          onChange={handleChange}

                          className="glassmorphism-input"

                        />

                        <MapPin className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                      </div>

                    </div>

                    <div className="relative p-6 border border-white/10 rounded-xl backdrop-blur-md bg-white/5">

                      <motion.div

                        className="absolute inset-0 rounded-xl opacity-20 bg-gradient-to-r from-teal-500/10 via-purple-500/10 to-teal-500/10"

                        animate={{

                          backgroundPosition: ["0% 0%", "100% 100%", "0% 0%"],

                        }}

                        transition={{ duration: 10, repeat: Number.POSITIVE\_INFINITY, ease: "linear" }}

                      />

                      <h3 className="text-lg font-medium text-white mb-4 flex items-center gap-2">

                        <Globe className="h-5 w-5 text-teal-400" />

                        Social Links

                      </h3>

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

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

                          <Label htmlFor="github" className="text-slate-300">

                            GitHub URL

                          </Label>

                          <div className="relative">

                            <Input

                              id="github"

                              name="github"

                              placeholder="https://github.com/yourusername"

                              value={formData.github}

                              onChange={handleChange}

                              className="glassmorphism-input"

                            />

                            <Github className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                          </div>

                        </div>

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

                          <Label htmlFor="linkedin" className="text-slate-300">

                            LinkedIn URL

                          </Label>

                          <div className="relative">

                            <Input

                              id="linkedin"

                              name="linkedin"

                              placeholder="https://linkedin.com/in/yourusername"

                              value={formData.linkedin}

                              onChange={handleChange}

                              className="glassmorphism-input"

                            />

                            <Linkedin className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                          </div>

                        </div>

                      </div>

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

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

                          <Label htmlFor="twitter" className="text-slate-300">

                            Twitter URL

                          </Label>

                          <div className="relative">

                            <Input

                              id="twitter"

                              name="twitter"

                              placeholder="https://twitter.com/yourusername"

                              value={formData.twitter}

                              onChange={handleChange}

                              className="glassmorphism-input"

                            />

                            <Twitter className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                          </div>

                        </div>

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

                          <Label htmlFor="website" className="text-slate-300">

                            Personal Website

                          </Label>

                          <div className="relative">

                            <Input

                              id="website"

                              name="website"

                              placeholder="https://yourwebsite.com"

                              value={formData.website}

                              onChange={handleChange}

                              className="glassmorphism-input"

                            />

                            <Globe className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-slate-500" />

                          </div>

                        </div>

                      </div>

                    </div>

                  </motion.div>

                </TabsContent>

                <TabsContent value="content" className="space-y-4">

                  <motion.div

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

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

                    transition={{ duration: 0.3 }}

                    className="space-y-6"

                  >

                    <div className="glassmorphism-panel">

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

                        <div className="h-8 w-8 rounded-full bg-gradient-to-r from-teal-500/40 to-teal-600/40 flex items-center justify-center backdrop-blur-md">

                          <GraduationCap className="h-4 w-4 text-white" />

                        </div>

                        <Label htmlFor="education" className="text-slate-100 text-lg">

                          Education

                        </Label>

                      </div>

                      <Textarea

                        id="education"

                        name="education"

                        placeholder="Your educational background"

                        value={formData.education}

                        onChange={handleChange}

                        rows={4}

                        className="glassmorphism-textarea"

                      />

                      <p className="text-sm text-slate-400 mt-2">Format: School Name, Degree, Year (one per line)</p>

                    </div>

                    <div className="glassmorphism-panel">

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

                        <div className="h-8 w-8 rounded-full bg-gradient-to-r from-purple-500/40 to-purple-600/40 flex items-center justify-center backdrop-blur-md">

                          <Building className="h-4 w-4 text-white" />

                        </div>

                        <Label htmlFor="experience" className="text-slate-100 text-lg">

                          Work Experience

                        </Label>

                      </div>

                      <Textarea

                        id="experience"

                        name="experience"

                        placeholder="Your work experience"

                        value={formData.experience}

                        onChange={handleChange}

                        rows={6}

                        className="glassmorphism-textarea"

                      />

                      <p className="text-sm text-slate-400 mt-2">

                        Format: Company, Position, Duration, Description (one entry per paragraph)

                      </p>

                    </div>

                    <div className="glassmorphism-panel">

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

                        <div className="h-8 w-8 rounded-full bg-gradient-to-r from-teal-500/40 to-purple-600/40 flex items-center justify-center backdrop-blur-md">

                          <FolderKanban className="h-4 w-4 text-white" />

                        </div>

                        <Label htmlFor="projects" className="text-slate-100 text-lg">

                          Projects

                        </Label>

                      </div>

                      <Textarea

                        id="projects"

                        name="projects"

                        placeholder="Your projects"

                        value={formData.projects}

                        onChange={handleChange}

                        rows={6}

                        className="glassmorphism-textarea"

                      />

                      <p className="text-sm text-slate-400 mt-2">

                        Format: Project Name, Technologies, Description (one project per paragraph)

                      </p>

                    </div>

                  </motion.div>

                </TabsContent>

                <TabsContent value="images" className="space-y-4">

                  <motion.div

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

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

                    transition={{ duration: 0.3 }}

                    className="space-y-6"

                  >

                    <div className="glassmorphism-panel">

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

                        <div className="h-8 w-8 rounded-full bg-gradient-to-r from-teal-500/40 to-teal-600/40 flex items-center justify-center backdrop-blur-md">

                          <User className="h-4 w-4 text-white" />

                        </div>

                        <Label className="text-slate-100 text-lg">Profile Photo</Label>

                      </div>

                      <FileUpload

                        label="Upload your profile photo"

                        onChange={handleProfileImageChange}

                        value={formData.profileImage}

                        accept="image/\*"

                        maxSizeMB={2}

                      />

                      <p className="text-sm text-slate-400 mt-2">

                        This image will be displayed as your profile photo. Square images work best.

                      </p>

                    </div>

                    <div className="glassmorphism-panel">

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

                        <div className="h-8 w-8 rounded-full bg-gradient-to-r from-purple-500/40 to-purple-600/40 flex items-center justify-center backdrop-blur-md">

                          <FolderKanban className="h-4 w-4 text-white" />

                        </div>

                        <Label className="text-slate-100 text-lg">Project Images (Optional)</Label>

                      </div>

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

                        {(formData.projectImages || ["", "", ""]).map((image, index) => (

                          <motion.div

                            key={index}

                            className="glassmorphism-card rounded-xl p-4"

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

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

                            transition={{ duration: 0.3, delay: index \* 0.1 }}

                          >

                            <FileUpload

                              label={`Project Image ${index + 1}`}

                              onChange={(base64) => handleProjectImageChange(index, base64)}

                              value={image}

                              accept="image/\*"

                              maxSizeMB={2}

                            />

                          </motion.div>

                        ))}

                      </div>

                      <p className="text-sm text-slate-400 mt-2">

                        These images will be displayed in your projects section.

                      </p>

                    </div>

                  </motion.div>

                </TabsContent>

                <TabsContent value="theme" className="space-y-4">

                  <motion.div

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

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

                    transition={{ duration: 0.3 }}

                    className="space-y-6"

                  >

                    <div className="glassmorphism-panel">

                      <Label className="text-slate-100 text-lg flex items-center gap-2 mb-6">

                        <div className="h-8 w-8 rounded-full bg-gradient-to-r from-teal-500/40 to-purple-600/40 flex items-center justify-center backdrop-blur-md">

                          <Palette className="h-4 w-4 text-white" />

                        </div>

                        Select a Theme

                      </Label>

                      <RadioGroup

                        value={formData.theme}

                        onValueChange={handleThemeChange}

                        className="grid grid-cols-1 md:grid-cols-3 gap-4"

                      >

                        <motion.div

                          whileHover={{ scale: 1.02, transition: { duration: 0.2 } }}

                          whileTap={{ scale: 0.98 }}

                        >

                          <RadioGroupItem value="creative" id="creative" className="peer sr-only" />

                          <Label

                            htmlFor="creative"

                            className="group flex flex-col items-center justify-between rounded-xl border border-slate-700/50 bg-black/30 backdrop-blur-md p-4 hover:bg-black/40 hover:border-teal-500/50 hover:text-slate-100 peer-data-[state=checked]:border-teal-500 peer-data-[state=checked]:ring-1 peer-data-[state=checked]:ring-teal-500/50 [&:has([data-state=checked])]:border-teal-500 transition-all duration-200"

                          >

                            <div className="mb-3 h-16 w-16 rounded-md bg-gradient-to-br from-purple-500 to-pink-500 flex items-center justify-center text-white font-bold group-hover:shadow-md group-hover:shadow-purple-500/20 transition-all">

                              C

                            </div>

                            <div className="space-y-1 text-center">

                              <p className="text-sm font-medium leading-none text-slate-200">Creative</p>

                              <p className="text-sm text-slate-400">Vibrant colors with modern animations</p>

                            </div>

                          </Label>

                        </motion.div>

                        <motion.div

                          whileHover={{ scale: 1.02, transition: { duration: 0.2 } }}

                          whileTap={{ scale: 0.98 }}

                        >

                          <RadioGroupItem value="minimal" id="minimal" className="peer sr-only" />

                          <Label

                            htmlFor="minimal"

                            className="group flex flex-col items-center justify-between rounded-xl border border-slate-700/50 bg-black/30 backdrop-blur-md p-4 hover:bg-black/40 hover:border-teal-500/50 hover:text-slate-100 peer-data-[state=checked]:border-teal-500 peer-data-[state=checked]:ring-1 peer-data-[state=checked]:ring-teal-500/50 [&:has([data-state=checked])]:border-teal-500 transition-all duration-200"

                          >

                            <div className="mb-3 h-16 w-16 rounded-md bg-gray-700 flex items-center justify-center text-white font-bold group-hover:shadow-md group-hover:shadow-gray-500/20 transition-all">

                              M

                            </div>

                            <div className="space-y-1 text-center">

                              <p className="text-sm font-medium leading-none text-slate-200">Minimal</p>

                              <p className="text-sm text-slate-400">Clean and simple design</p>

                            </div>

                          </Label>

                        </motion.div>

                        <motion.div

                          whileHover={{ scale: 1.02, transition: { duration: 0.2 } }}

                          whileTap={{ scale: 0.98 }}

                        >

                          <RadioGroupItem value="modern" id="modern" className="peer sr-only" />

                          <Label

                            htmlFor="modern"

                            className="group flex flex-col items-center justify-between rounded-xl border border-slate-700/50 bg-black/30 backdrop-blur-md p-4 hover:bg-black/40 hover:border-teal-500/50 hover:text-slate-100 peer-data-[state=checked]:border-teal-500 peer-data-[state=checked]:ring-1 peer-data-[state=checked]:ring-teal-500/50 [&:has([data-state=checked])]:border-teal-500 transition-all duration-200"

                          >

                            <div className="mb-3 h-16 w-16 rounded-md bg-gradient-to-r from-blue-500 to-cyan-400 flex items-center justify-center text-white font-bold group-hover:shadow-md group-hover:shadow-blue-500/20 transition-all">

                              M

                            </div>

                            <div className="space-y-1 text-center">

                              <p className="text-sm font-medium leading-none text-slate-200">Modern</p>

                              <p className="text-sm text-slate-400">Professional with subtle animations</p>

                            </div>

                          </Label>

                        </motion.div>

                      </RadioGroup>

                    </div>

                    <div className="glassmorphism-panel mt-8">

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

                        <div className="h-8 w-8 rounded-full bg-gradient-to-r from-teal-500/40 to-purple-600/40 flex items-center justify-center backdrop-blur-md">

                          <Palette className="h-4 w-4 text-white" />

                        </div>

                        <h3 className="font-medium text-slate-100">Theme Preview</h3>

                      </div>

                      <motion.div

                        className="rounded-xl overflow-hidden border border-white/10"

                        whileHover={{ scale: 1.01 }}

                        transition={{ duration: 0.3 }}

                      >

                        <div

                          className={`h-48 flex items-center justify-center overflow-hidden relative ${

                            formData.theme === "creative"

                              ? "bg-gradient-to-br from-purple-500 to-pink-500 text-white"

                              : formData.theme === "minimal"

                                ? "bg-gray-800 text-white border border-gray-700"

                                : "bg-gradient-to-r from-blue-500 to-cyan-400 text-white"

                          }`}

                        >

                          {formData.profileImage && (

                            <img

                              src={formData.profileImage || "/placeholder.svg"}

                              alt="Preview"

                              className="absolute inset-0 w-full h-full object-cover opacity-20 scale-105 blur-sm"

                            />

                          )}

                          <div className="absolute inset-0 bg-black/30 backdrop-blur-sm" />

                          <div className="relative z-10 flex flex-col items-center gap-4">

                            {formData.profileImage && (

                              <div className="w-20 h-20 rounded-full overflow-hidden border-2 border-white/50">

                                <img

                                  src={formData.profileImage || "/placeholder.svg"}

                                  alt="Preview"

                                  className="w-full h-full object-cover"

                                />

                              </div>

                            )}

                            <motion.div

                              initial={{ opacity: 0 }}

                              animate={{ opacity: 1 }}

                              transition={{ duration: 0.5 }}

                            >

                              <p className="font-bold text-xl">{formData.fullName || "Your Name"}</p>

                              <p className="text-center opacity-80">{formData.jobTitle || "Your Profession"}</p>

                            </motion.div>

                          </div>

                        </div>

                      </motion.div>

                      <p className="text-sm text-slate-400 mt-4">

                        This is a simplified preview. The actual portfolio will have a complete design with all your

                        information.

                      </p>

                    </div>

                  </motion.div>

                </TabsContent>

              </Tabs>

            </CardContent>

            <CardFooter className="flex justify-between p-8">

              <Button

                type="button"

                variant="outline"

                onClick={() => {

                  const prevTab = {

                    basic: "basic",

                    contact: "basic",

                    content: "contact",

                    images: "content",

                    theme: "images",

                  }[activeTab]

                  if (prevTab !== activeTab) {

                    setActiveTab(prevTab)

                  }

                }}

                disabled={activeTab === "basic"}

                className="glassmorphism-button-outline"

              >

                Previous

              </Button>

              {activeTab !== "theme" ? (

                <Button

                  type="button"

                  onClick={() => {

                    const nextTab = {

                      basic: "contact",

                      contact: "content",

                      content: "images",

                      images: "theme",

                      theme: "theme",

                    }[activeTab]

                    setActiveTab(nextTab)

                  }}

                  className="glassmorphism-button"

                >

                  Next

                </Button>

              ) : (

                <motion.div whileHover={{ scale: 1.03 }} whileTap={{ scale: 0.97 }}>

                  <Button

                    type="submit"

                    disabled={isLoading}

                    className="relative group glassmorphism-button overflow-hidden"

                  >

                    <motion.div

                      className="absolute inset-0 bg-gradient-to-r from-teal-500/20 via-purple-500/20 to-teal-500/20"

                      animate={{

                        x: ["-100%", "100%"],

                      }}

                      transition={{ duration: 3, repeat: Number.POSITIVE\_INFINITY, ease: "linear" }}

                    />

                    <span className="flex items-center gap-2 relative z-10">

                      {isLoading ? (

                        <>

                          <Loader2 className="h-4 w-4 animate-spin" /> Saving...

                        </>

                      ) : (

                        <>

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

                        </>

                      )}

                    </span>

                  </Button>

                </motion.div>

              )}

            </CardFooter>

          </motion.div>

        </form>

      </div>

    </DashboardShell>

  )

}

Login

"use client"

import type React from "react"

import { useState } from "react"

import Link from "next/link"

import { useRouter } from "next/navigation"

import { motion } from "framer-motion"

import { ArrowLeft, Loader2 } from "lucide-react"

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

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

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

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

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

export default function LoginPage() {

  const [email, setEmail] = useState("")

  const [password, setPassword] = useState("")

  const [isLoading, setIsLoading] = useState(false)

  const [errors, setErrors] = useState<{ email?: string; password?: string }>({})

  const router = useRouter()

  const { toast } = useToast()

  // Hardcoded credentials for demo

  const validCredentials = {

    email: "demo@example.com",

    password: "password123",

  }

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

    e.preventDefault()

    // Reset errors

    setErrors({})

    // Validate form

    const newErrors: { email?: string; password?: string } = {}

    let isValid = true

    if (!email) {

      newErrors.email = "Email is required"

      isValid = false

    } else if (!/\S+@\S+\.\S+/.test(email)) {

      newErrors.email = "Email is invalid"

      isValid = false

    }

    if (!password) {

      newErrors.password = "Password is required"

      isValid = false

    }

    if (!isValid) {

      setErrors(newErrors)

      return

    }

    // Simulate login

    setIsLoading(true)

    setTimeout(() => {

      if (email === validCredentials.email && password === validCredentials.password) {

        // Store user in localStorage

        localStorage.setItem("user", JSON.stringify({ email, isLoggedIn: true }))

        toast({

          title: "Login successful",

          description: "Welcome back to PortfolioPro!",

        })

        router.push("/dashboard")

      } else {

        toast({

          title: "Login failed",

          description: "Invalid email or password. Try demo@example.com / password123",

          variant: "destructive",

        })

      }

      setIsLoading(false)

    }, 1500)

  }

  return (

    <div className="min-h-screen flex items-center justify-center bg-gradient-to-b from-slate-950 to-slate-900 p-4 text-slate-200">

      {/\* Glow effects \*/}

      <div className="absolute top-20 left-1/4 w-96 h-96 bg-teal-500/10 rounded-full blur-3xl" />

      <div className="absolute bottom-20 right-1/4 w-96 h-96 bg-purple-500/10 rounded-full blur-3xl" />

      <div className="w-full max-w-md z-10">

        <motion.div

          initial={{ opacity: 0, y: 20 }}

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

          transition={{ duration: 0.5 }}

          className="mb-8"

        >

          <Button

            variant="ghost"

            size="sm"

            asChild

            className="mb-8 text-slate-300 hover:text-slate-100 hover:bg-slate-800"

          >

            <Link href="/" className="gap-2">

              <ArrowLeft className="h-4 w-4" /> Back to home

            </Link>

          </Button>

        </motion.div>

        <motion.div

          initial={{ opacity: 0, y: 20 }}

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

          transition={{ duration: 0.5, delay: 0.1 }}

        >

          <Card className="border-slate-700 bg-slate-800/50 backdrop-blur-sm text-slate-200">

            <CardHeader>

              <CardTitle className="text-2xl text-slate-100">Login</CardTitle>

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

                Enter your credentials to access your account

              </CardDescription>

            </CardHeader>

            <CardContent>

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

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

                  <Label htmlFor="email" className="text-slate-300">

                    Email

                  </Label>

                  <Input

                    id="email"

                    type="email"

                    placeholder="you@example.com"

                    value={email}

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

                    className={`bg-slate-900 border-slate-700 text-slate-200 ${errors.email ? "border-red-500" : ""}`}

                  />

                  {errors.email && (

                    <motion.p

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

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

                      className="text-red-400 text-sm"

                    >

                      {errors.email}

                    </motion.p>

                  )}

                </div>

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

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

                    <Label htmlFor="password" className="text-slate-300">

                      Password

                    </Label>

                    <Link href="#" className="text-sm text-teal-400 hover:text-teal-300">

                      Forgot password?

                    </Link>

                  </div>

                  <Input

                    id="password"

                    type="password"

                    value={password}

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

                    className={`bg-slate-900 border-slate-700 text-slate-200 ${errors.password ? "border-red-500" : ""}`}

                  />

                  {errors.password && (

                    <motion.p

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

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

                      className="text-red-400 text-sm"

                    >

                      {errors.password}

                    </motion.p>

                  )}

                </div>

                <Button

                  type="submit"

                  className="w-full bg-gradient-to-r from-teal-500 to-teal-600 hover:from-teal-600 hover:to-teal-700 text-white border-0"

                  disabled={isLoading}

                >

                  {isLoading ? (

                    <>

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

                    </>

                  ) : (

                    "Login"

                  )}

                </Button>

              </form>

            </CardContent>

            <CardFooter className="flex flex-col space-y-4">

              <div className="text-sm text-center text-slate-400">

                <span>Demo credentials: demo@example.com / password123</span>

              </div>

              <div className="text-sm text-center text-slate-400">

                <span>Don&apos;t have an account? </span>

                <Link href="/register" className="text-teal-400 hover:text-teal-300">

                  Register

                </Link>

              </div>

            </CardFooter>

          </Card>

        </motion.div>

      </div>

    </div>

  )

}

Portfolio page

"use client"

import { useState, useEffect } from "react"

import { useRouter, useParams } from "next/navigation"

import Link from "next/link"

import { motion, AnimatePresence } from "framer-motion"

import { ArrowLeft, Copy, Check, Edit, Share2 } from "lucide-react"

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

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

import { Tabs, TabsList, TabsTrigger } from "@/components/ui/tabs"

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

import { CreativeTheme } from "@/components/portfolio-themes/creative-theme"

import { MinimalTheme } from "@/components/portfolio-themes/minimal-theme"

import { ModernTheme } from "@/components/portfolio-themes/modern-theme"

interface Portfolio {

  id: string

  title: string

  description: string

  fullName: string

  jobTitle: string

  email: string

  phone: string

  location: string

  bio: string

  skills: string

  github: string

  linkedin: string

  twitter: string

  website: string

  education: string

  experience: string

  projects: string

  theme: string

  createdAt: string

}

export default function PortfolioPage() {

  const [portfolio, setPortfolio] = useState<Portfolio | null>(null)

  const [isLoading, setIsLoading] = useState(true)

  const [currentTheme, setCurrentTheme] = useState<string>("")

  const [copied, setCopied] = useState(false)

  const router = useRouter()

  const params = useParams()

  const { toast } = useToast()

  const portfolioId = params.id as string

  useEffect(() => {

    // Load portfolio from localStorage

    const storedPortfolios = localStorage.getItem("portfolios")

    if (storedPortfolios) {

      const portfolios = JSON.parse(storedPortfolios)

      const foundPortfolio = portfolios.find((p: Portfolio) => p.id === portfolioId)

      if (foundPortfolio) {

        setPortfolio(foundPortfolio)

        setCurrentTheme(foundPortfolio.theme)

      } else {

        toast({

          title: "Portfolio not found",

          description: "The requested portfolio could not be found.",

          variant: "destructive",

        })

        router.push("/dashboard")

      }

    } else {

      toast({

        title: "No portfolios found",

        description: "You don't have any portfolios yet.",

        variant: "destructive",

      })

      router.push("/dashboard")

    }

    setIsLoading(false)

  }, [portfolioId, router, toast])

  const handleThemeChange = (theme: string) => {

    setCurrentTheme(theme)

    if (portfolio) {

      // Update portfolio theme in localStorage

      const storedPortfolios = localStorage.getItem("portfolios")

      if (storedPortfolios) {

        const portfolios = JSON.parse(storedPortfolios)

        const updatedPortfolios = portfolios.map((p: Portfolio) => {

          if (p.id === portfolioId) {

            return { ...p, theme }

          }

          return p

        })

        localStorage.setItem("portfolios", JSON.stringify(updatedPortfolios))

        toast({

          title: "Theme updated",

          description: `Portfolio theme changed to ${theme}.`,

        })

      }

    }

  }

  const copyPortfolioLink = () => {

    const link = `${window.location.origin}/portfolio/${portfolioId}`

    navigator.clipboard.writeText(link)

    setCopied(true)

    toast({

      title: "Link copied",

      description: "Portfolio link copied to clipboard.",

    })

    setTimeout(() => setCopied(false), 2000)

  }

  if (isLoading || !portfolio) {

    return (

      <div className="flex h-screen items-center justify-center bg-slate-950">

        <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2 border-teal-500"></div>

      </div>

    )

  }

  return (

    <div className="min-h-screen bg-slate-950 text-slate-200">

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

        <div className="flex flex-col md:flex-row justify-between items-start md:items-center gap-4 mb-8">

          <div>

            <Button

              variant="ghost"

              size="sm"

              asChild

              className="mb-2 text-slate-300 hover:text-slate-100 hover:bg-slate-800"

            >

              <Link href="/dashboard" className="gap-2">

                <ArrowLeft className="h-4 w-4" /> Back to Dashboard

              </Link>

            </Button>

            <h1 className="text-3xl font-bold text-slate-100">{portfolio.title}</h1>

            <p className="text-slate-400">{portfolio.description}</p>

          </div>

          <div className="flex flex-wrap gap-2">

            <Button

              variant="outline"

              size="sm"

              asChild

              className="border-slate-700 text-slate-300 hover:bg-slate-800 hover:text-slate-100"

            >

              <Link href={`/edit-portfolio/${portfolioId}`} className="gap-2">

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

              </Link>

            </Button>

            <Button

              variant="outline"

              size="sm"

              onClick={copyPortfolioLink}

              className="gap-2 border-slate-700 text-slate-300 hover:bg-slate-800 hover:text-slate-100"

            >

              {copied ? (

                <>

                  <Check className="h-4 w-4" /> Copied

                </>

              ) : (

                <>

                  <Copy className="h-4 w-4" /> Copy Link

                </>

              )}

            </Button>

            <Button

              size="sm"

              className="gap-2 bg-gradient-to-r from-teal-500 to-teal-600 hover:from-teal-600 hover:to-teal-700 text-white border-0"

            >

              <Share2 className="h-4 w-4" /> Share

            </Button>

          </div>

        </div>

        <Card className="mb-8 bg-slate-800 border-slate-700 text-slate-200">

          <CardContent className="p-6">

            <h2 className="text-xl font-semibold mb-4 text-slate-100">Theme Selection</h2>

            <Tabs value={currentTheme} onValueChange={handleThemeChange} className="w-full">

              <TabsList className="grid grid-cols-3 mb-4 bg-slate-900">

                <TabsTrigger

                  value="creative"

                  className="data-[state=active]:bg-slate-800 data-[state=active]:text-slate-100"

                >

                  Creative

                </TabsTrigger>

                <TabsTrigger

                  value="minimal"

                  className="data-[state=active]:bg-slate-800 data-[state=active]:text-slate-100"

                >

                  Minimal

                </TabsTrigger>

                <TabsTrigger

                  value="modern"

                  className="data-[state=active]:bg-slate-800 data-[state=active]:text-slate-100"

                >

                  Modern

                </TabsTrigger>

              </TabsList>

              <div className="text-sm text-slate-400 mb-4">Select a theme to preview and apply to your portfolio.</div>

            </Tabs>

          </CardContent>

        </Card>

        <div className="border rounded-lg overflow-hidden bg-slate-900 border-slate-700">

          <div className="p-4 bg-slate-800 border-b border-slate-700 flex justify-between items-center">

            <h2 className="font-medium text-slate-200">Portfolio Preview</h2>

            <div className="flex gap-2">

              <Button

                variant="outline"

                size="sm"

                onClick={copyPortfolioLink}

                className="gap-1 border-slate-700 text-slate-300 hover:bg-slate-700 hover:text-slate-100"

              >

                {copied ? <Check className="h-3 w-3" /> : <Copy className="h-3 w-3" />}

                <span className="text-xs">Copy Link</span>

              </Button>

            </div>

          </div>

          <div className="bg-slate-900">

            <AnimatePresence mode="wait">

              <motion.div

                key={currentTheme}

                initial={{ opacity: 0 }}

                animate={{ opacity: 1 }}

                exit={{ opacity: 0 }}

                transition={{ duration: 0.3 }}

              >

                {currentTheme === "creative" && <CreativeTheme portfolio={portfolio} />}

                {currentTheme === "minimal" && <MinimalTheme portfolio={portfolio} />}

                {currentTheme === "modern" && <ModernTheme portfolio={portfolio} />}

              </motion.div>

            </AnimatePresence>

          </div>

        </div>

      </div>

    </div>

  )

}

Register

"use client"

import type React from "react"

import { useState } from "react"

import Link from "next/link"

import { useRouter } from "next/navigation"

import { motion } from "framer-motion"

import { ArrowLeft, Loader2 } from "lucide-react"

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

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

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

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

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

export default function RegisterPage() {

  const [name, setName] = useState("")

  const [email, setEmail] = useState("")

  const [password, setPassword] = useState("")

  const [confirmPassword, setConfirmPassword] = useState("")

  const [isLoading, setIsLoading] = useState(false)

  const [errors, setErrors] = useState<{

    name?: string

    email?: string

    password?: string

    confirmPassword?: string

  }>({})

  const router = useRouter()

  const { toast } = useToast()

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

    e.preventDefault()

    // Reset errors

    setErrors({})

    // Validate form

    const newErrors: {

      name?: string

      email?: string

      password?: string

      confirmPassword?: string

    } = {}

    let isValid = true

    if (!name) {

      newErrors.name = "Name is required"

      isValid = false

    }

    if (!email) {

      newErrors.email = "Email is required"

      isValid = false

    } else if (!/\S+@\S+\.\S+/.test(email)) {

      newErrors.email = "Email is invalid"

      isValid = false

    }

    if (!password) {

      newErrors.password = "Password is required"

      isValid = false

    } else if (password.length < 8) {

      newErrors.password = "Password must be at least 8 characters"

      isValid = false

    }

    if (!confirmPassword) {

      newErrors.confirmPassword = "Please confirm your password"

      isValid = false

    } else if (password !== confirmPassword) {

      newErrors.confirmPassword = "Passwords do not match"

      isValid = false

    }

    if (!isValid) {

      setErrors(newErrors)

      return

    }

    // Simulate registration

    setIsLoading(true)

    setTimeout(() => {

      // Store user in localStorage

      localStorage.setItem("user", JSON.stringify({ name, email, isLoggedIn: true }))

      toast({

        title: "Registration successful",

        description: "Welcome to PortfolioPro!",

      })

      router.push("/dashboard")

      setIsLoading(false)

    }, 1500)

  }

  return (

    <div className="min-h-screen flex items-center justify-center bg-gradient-to-b from-slate-950 to-slate-900 p-4 text-slate-200">

      {/\* Glow effects \*/}

      <div className="absolute top-20 right-1/4 w-96 h-96 bg-teal-500/10 rounded-full blur-3xl" />

      <div className="absolute bottom-20 left-1/4 w-96 h-96 bg-purple-500/10 rounded-full blur-3xl" />

      <div className="w-full max-w-md z-10">

        <motion.div

          initial={{ opacity: 0, y: 20 }}

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

          transition={{ duration: 0.5 }}

          className="mb-8"

        >

          <Button

            variant="ghost"

            size="sm"

            asChild

            className="mb-8 text-slate-300 hover:text-slate-100 hover:bg-slate-800"

          >

            <Link href="/" className="gap-2">

              <ArrowLeft className="h-4 w-4" /> Back to home

            </Link>

          </Button>

        </motion.div>

        <motion.div

          initial={{ opacity: 0, y: 20 }}

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

          transition={{ duration: 0.5, delay: 0.1 }}

        >

          <Card className="border-slate-700 bg-slate-800/50 backdrop-blur-sm text-slate-200">

            <CardHeader>

              <CardTitle className="text-2xl text-slate-100">Create an account</CardTitle>

              <CardDescription className="text-slate-400">Enter your information to create an account</CardDescription>

            </CardHeader>

            <CardContent>

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

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

                  <Label htmlFor="name" className="text-slate-300">

                    Full Name

                  </Label>

                  <Input

                    id="name"

                    placeholder="John Doe"

                    value={name}

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

                    className={`bg-slate-900 border-slate-700 text-slate-200 ${errors.name ? "border-red-500" : ""}`}

                  />

                  {errors.name && (

                    <motion.p

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

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

                      className="text-red-400 text-sm"

                    >

                      {errors.name}

                    </motion.p>

                  )}

                </div>

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

                  <Label htmlFor="email" className="text-slate-300">

                    Email

                  </Label>

                  <Input

                    id="email"

                    type="email"

                    placeholder="you@example.com"

                    value={email}

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

                    className={`bg-slate-900 border-slate-700 text-slate-200 ${errors.email ? "border-red-500" : ""}`}

                  />

                  {errors.email && (

                    <motion.p

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

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

                      className="text-red-400 text-sm"

                    >

                      {errors.email}

                    </motion.p>

                  )}

                </div>

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

                  <Label htmlFor="password" className="text-slate-300">

                    Password

                  </Label>

                  <Input

                    id="password"

                    type="password"

                    value={password}

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

                    className={`bg-slate-900 border-slate-700 text-slate-200 ${errors.password ? "border-red-500" : ""}`}

                  />

                  {errors.password && (

                    <motion.p

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

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

                      className="text-red-400 text-sm"

                    >

                      {errors.password}

                    </motion.p>

                  )}

                </div>

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

                  <Label htmlFor="confirmPassword" className="text-slate-300">

                    Confirm Password

                  </Label>

                  <Input

                    id="confirmPassword"

                    type="password"

                    value={confirmPassword}

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

                    className={`bg-slate-900 border-slate-700 text-slate-200 ${errors.confirmPassword ? "border-red-500" : ""}`}

                  />

                  {errors.confirmPassword && (

                    <motion.p

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

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

                      className="text-red-400 text-sm"

                    >

                      {errors.confirmPassword}

                    </motion.p>

                  )}

                </div>

                <Button

                  type="submit"

                  className="w-full bg-gradient-to-r from-teal-500 to-teal-600 hover:from-teal-600 hover:to-teal-700 text-white border-0"

                  disabled={isLoading}

                >

                  {isLoading ? (

                    <>

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

                    </>

                  ) : (

                    "Register"

                  )}

                </Button>

              </form>

            </CardContent>

            <CardFooter>

              <div className="text-sm text-center w-full text-slate-400">

                <span>Already have an account? </span>

                <Link href="/login" className="text-teal-400 hover:text-teal-300">

                  Login

                </Link>

              </div>

            </CardFooter>

          </Card>

        </motion.div>

      </div>

    </div>

  )

}

"use client"

import Link from "next/link"

import { motion } from "framer-motion"

import { ArrowRight, Code, Palette, Share2, Sparkles } from "lucide-react"

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

export default function LandingPage() {

  const features = [

    {

      icon: <Code className="h-6 w-6" />,

      title: "Custom Portfolio Creation",

      description: "Build your professional portfolio in minutes with our intuitive editor.",

    },

    {

      icon: <Palette className="h-6 w-6" />,

      title: "Multiple Themes",

      description: "Choose from a variety of professionally designed themes to match your style.",

    },

    {

      icon: <Share2 className="h-6 w-6" />,

      title: "Easy Sharing",

      description: "Generate a unique link to share your portfolio with potential employers.",

    },

    {

      icon: <Sparkles className="h-6 w-6" />,

      title: "Animated Elements",

      description: "Make your portfolio stand out with beautiful animations and transitions.",

    },

  ]

  return (

    <div className="min-h-screen bg-gradient-to-b from-slate-950 to-slate-900 text-slate-200 overflow-hidden">

      {/\* Hero Section \*/}

      <div className="relative">

        {/\* Glow effects \*/}

        <div className="absolute top-20 left-1/4 w-96 h-96 bg-teal-500/10 rounded-full blur-3xl" />

        <div className="absolute top-40 right-1/4 w-96 h-96 bg-purple-500/10 rounded-full blur-3xl" />

        <div className="container mx-auto px-4 pt-32 pb-20 relative z-10">

          <motion.div

            initial={{ opacity: 0, y: 20 }}

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

            transition={{ duration: 0.8 }}

            className="text-center max-w-4xl mx-auto"

          >

            <motion.div

              initial={{ scale: 0.9, opacity: 0 }}

              animate={{ scale: 1, opacity: 1 }}

              transition={{ delay: 0.2, duration: 0.8 }}

              className="mb-6 inline-block"

            >

              <span className="px-4 py-2 rounded-full bg-teal-500/20 text-teal-300 text-sm font-medium">

                Launch Your Professional Portfolio Today

              </span>

            </motion.div>

            <motion.h1

              initial={{ opacity: 0, y: 20 }}

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

              transition={{ delay: 0.3, duration: 0.8 }}

              className="text-5xl md:text-6xl font-bold mb-6 bg-clip-text text-transparent bg-gradient-to-r from-teal-400 to-purple-400"

            >

              Create Stunning Portfolios in Minutes

            </motion.h1>

            <motion.p

              initial={{ opacity: 0, y: 20 }}

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

              transition={{ delay: 0.4, duration: 0.8 }}

              className="text-xl text-slate-400 mb-10"

            >

              Showcase your work with our powerful portfolio generator. Choose from multiple themes, add animations, and

              share your professional story with the world.

            </motion.p>

            <motion.div

              initial={{ opacity: 0, y: 20 }}

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

              transition={{ delay: 0.5, duration: 0.8 }}

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

            >

              <Button

                asChild

                size="lg"

                className="rounded-full px-8 gap-2 bg-gradient-to-r from-teal-500 to-teal-600 hover:from-teal-600 hover:to-teal-700 text-white border-0"

              >

                <Link href="/register">

                  Get Started <ArrowRight className="h-4 w-4" />

                </Link>

              </Button>

              <Button

                asChild

                variant="outline"

                size="lg"

                className="rounded-full px-8 border-slate-700 text-slate-300 hover:bg-slate-800 hover:text-slate-200"

              >

                <Link href="/login">Login</Link>

              </Button>

            </motion.div>

          </motion.div>

        </div>

      </div>

      {/\* Features Section \*/}

      <div className="container mx-auto px-4 py-24">

        <motion.div

          initial={{ opacity: 0, y: 20 }}

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

          transition={{ duration: 0.8 }}

          viewport={{ once: true }}

          className="text-center mb-16"

        >

          <h2 className="text-3xl md:text-4xl font-bold mb-4 text-slate-100">Powerful Features</h2>

          <p className="text-xl text-slate-400 max-w-2xl mx-auto">

            Everything you need to create and manage professional portfolios

          </p>

        </motion.div>

        <div className="grid md:grid-cols-2 lg:grid-cols-4 gap-8">

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

            <motion.div

              key={index}

              initial={{ opacity: 0, y: 20 }}

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

              transition={{ delay: index \* 0.1, duration: 0.5 }}

              viewport={{ once: true }}

              className="bg-slate-800/50 backdrop-blur-sm p-6 rounded-xl border border-slate-700"

            >

              <div className="bg-teal-500/10 p-3 rounded-lg w-fit mb-4 text-teal-300">{feature.icon}</div>

              <h3 className="text-xl font-semibold mb-2 text-slate-100">{feature.title}</h3>

              <p className="text-slate-400">{feature.description}</p>

            </motion.div>

          ))}

        </div>

      </div>

      {/\* How It Works Section \*/}

      <div className="container mx-auto px-4 py-24 relative">

        <div className="absolute top-1/2 left-1/3 w-96 h-96 bg-purple-500/10 rounded-full blur-3xl -z-10" />

        <motion.div

          initial={{ opacity: 0, y: 20 }}

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

          transition={{ duration: 0.8 }}

          viewport={{ once: true }}

          className="text-center mb-16"

        >

          <h2 className="text-3xl md:text-4xl font-bold mb-4 text-slate-100">How It Works</h2>

          <p className="text-xl text-slate-400 max-w-2xl mx-auto">Create your portfolio in three simple steps</p>

        </motion.div>

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

          {[

            { step: "1", title: "Create an account", description: "Sign up and access your personal dashboard" },

            { step: "2", title: "Fill in your details", description: "Add your work experience, skills, and projects" },

            { step: "3", title: "Choose a theme", description: "Select a theme and customize your portfolio" },

          ].map((step, index) => (

            <motion.div

              key={index}

              initial={{ opacity: 0, y: 20 }}

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

              transition={{ delay: index \* 0.2, duration: 0.5 }}

              viewport={{ once: true }}

              className="relative"

            >

              <div className="bg-slate-800/50 backdrop-blur-sm p-8 rounded-xl border border-slate-700 h-full">

                <div className="absolute -top-5 left-1/2 -translate-x-1/2 bg-gradient-to-r from-teal-500 to-teal-600 text-white w-10 h-10 rounded-full flex items-center justify-center font-bold">

                  {step.step}

                </div>

                <h3 className="text-xl font-semibold mb-2 mt-4 text-slate-100">{step.title}</h3>

                <p className="text-slate-400">{step.description}</p>

              </div>

            </motion.div>

          ))}

        </div>

      </div>

      {/\* CTA Section \*/}

      <div className="container mx-auto px-4 py-24">

        <motion.div

          initial={{ opacity: 0, scale: 0.95 }}

          whileInView={{ opacity: 1, scale: 1 }}

          transition={{ duration: 0.8 }}

          viewport={{ once: true }}

          className="bg-gradient-to-r from-slate-800/80 to-slate-900/80 rounded-2xl p-12 text-center max-w-4xl mx-auto backdrop-blur-sm border border-slate-700/50"

        >

          <h2 className="text-3xl md:text-4xl font-bold mb-4 text-slate-100">Ready to Showcase Your Work?</h2>

          <p className="text-xl text-slate-400 mb-8 max-w-2xl mx-auto">

            Join thousands of professionals who have created stunning portfolios with our platform.

          </p>

          <Button

            asChild

            size="lg"

            className="rounded-full px-8 gap-2 bg-gradient-to-r from-teal-500 to-teal-600 hover:from-teal-600 hover:to-teal-700 text-white border-0"

          >

            <Link href="/register">

              Get Started Now <ArrowRight className="h-4 w-4" />

            </Link>

          </Button>

        </motion.div>

      </div>

      {/\* Footer \*/}

      <footer className="border-t border-slate-800 py-12">

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

          <div className="flex flex-col md:flex-row justify-between items-center">

            <div className="mb-6 md:mb-0">

              <h3 className="text-2xl font-bold text-slate-100">

                Portfolio<span className="text-teal-400">Pro</span>

              </h3>

              <p className="text-slate-400">Create stunning portfolios in minutes</p>

            </div>

            <div className="flex gap-8">

              <Link href="/login" className="text-slate-400 hover:text-teal-300 transition-colors">

                Login

              </Link>

              <Link href="/register" className="text-slate-400 hover:text-teal-300 transition-colors">

                Register

              </Link>

              <Link href="#" className="text-slate-400 hover:text-teal-300 transition-colors">

                Features

              </Link>

            </div>

          </div>

          <div className="mt-8 pt-8 border-t border-slate-800 text-center text-slate-500">

            <p>© {new Date().getFullYear()} PortfolioPro. All rights reserved.</p>

          </div>

        </div>

      </footer>

    </div>

  )

}

BACKEND

Configuration

Package com.portfolio\_pro.app.configs;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.CorsRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration

public class GlobalCorsConfig {

    @Bean

    WebMvcConfigurer corsConfigurer() {

        return new WebMvcConfigurer() {

            @Override

            public void addCorsMappings(CorsRegistry registry) {

                registry.addMapping("/\*\*")

                        .allowedOrigins("http://localhost:3000", "http://localhost:5173")

                        .allowedMethods("GET", "POST", "PUT", "DELETE", "PATCH")

                        .allowedHeaders("\*")

                        .allowCredentials(true);

            }

        };

    }

}

package com.portfolio\_pro.app.configs;

import java.io.IOException;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

import com.portfolio\_pro.app.services.JwtService;

import com.portfolio\_pro.app.utils.AuthUtil;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.Cookie;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

@Component

public class JwtAuthenticationFilter extends OncePerRequestFilter {

    @Autowired

    JwtService jwtService;

    @Autowired

    UserDetailsService userDetailsService;

    @Autowired

    AuthUtil authUtil;

    @Override

    protected void doFilterInternal(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, FilterChain filterChain)

            throws ServletException, IOException {

        Cookie[] cookies = httpServletRequest.getCookies();

        if(cookies == null) {

            filterChain.doFilter(httpServletRequest, httpServletResponse);

            return;

        }

        String authCookie = "";

        for(Cookie cookie: cookies) {

            if(cookie.getName().equals("AuthCookie")) {

                authCookie = cookie.getValue();

            }

        }

        if (authCookie.isEmpty()) {

            filterChain.doFilter(httpServletRequest, httpServletResponse);

            System.out.println("Lol");

            return;

        }

        final String jwt = authCookie;

        final String username = jwtService.extractUsername(jwt);

        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

        if (username != null && authentication == null) {

            UserDetails userDetails = userDetailsService.loadUserByUsername(username);

            System.out.println(userDetails.getUsername());

            if (jwtService.isTokenValid(jwt, userDetails)) {

                UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(

                        userDetails, null, userDetails.getAuthorities());

                authenticationToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(httpServletRequest));

                SecurityContextHolder.getContext().setAuthentication(authenticationToken);

            }

        }

        filterChain.doFilter(httpServletRequest, httpServletResponse);

    }

}

package com.portfolio\_pro.app.configs;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.AuthenticationProvider;

import org.springframework.security.authentication.dao.DaoAuthenticationProvider;

import org.springframework.security.config.Customizer;

import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import org.springframework.web.cors.CorsConfiguration;

import org.springframework.web.cors.reactive.CorsConfigurationSource;

import org.springframework.web.cors.reactive.UrlBasedCorsConfigurationSource;

@Configuration

@EnableWebSecurity

public class WebSecurityConfig {

    @Autowired

    UserDetailsService userDetailsService;

    @Autowired

    JwtAuthenticationFilter jwtAuthenticationFilter;

    @Bean

    SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception {

        httpSecurity.csrf(csrf -> csrf.disable()) // .cors(Customizer.withDefaults())

                .sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))

                .authorizeHttpRequests(request -> request.requestMatchers("/api/auth/signup", "/api/auth/signin")

                        .permitAll().requestMatchers("/api/public/\*\*", "/api/users/exists/\*\*").permitAll().anyRequest()

                        .authenticated())

                .httpBasic(Customizer.withDefaults())

                .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class);

        return httpSecurity.build();

    }

    /\*

    CorsConfigurationSource configurationSource() {

        CorsConfiguration corsConfiguration = new CorsConfiguration();

        corsConfiguration.setAllowedOrigins(List.of("http://localhost:3000","http://127.0.0.1:5173/", "http://vercel.app"));

        corsConfiguration.setAllowedMethods(List.of("GET", "POST", "PUT", "DELETE", "PATCH"));

        corsConfiguration.setAllowCredentials(true);

        corsConfiguration.setMaxAge(3600L);

        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();

        source.registerCorsConfiguration("/\*\*", corsConfiguration);

        return source;

    }

    \*/

    @Bean

    BCryptPasswordEncoder bCryptPasswordEncoder() {

        return new BCryptPasswordEncoder(14);

    }

    @Bean

    AuthenticationProvider authenticationProvider(UserDetailsService userDetailsService,

            PasswordEncoder passwordEncoder) {

        DaoAuthenticationProvider provider = new DaoAuthenticationProvider();

        provider.setUserDetailsService(userDetailsService);

        provider.setPasswordEncoder(passwordEncoder);

        return provider;

    }

    @Bean

    AuthenticationManager authenticationManager(AuthenticationConfiguration configuration) throws Exception {

        return configuration.getAuthenticationManager();

    }

}

Controller

package com.portfolio\_pro.app.controllers;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.ResponseStatus;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import com.portfolio\_pro.app.dtos.ResponseDto;

import com.portfolio\_pro.app.dtos.UserSigninDto;

import com.portfolio\_pro.app.dtos.UserSignupDto;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.User;

import com.portfolio\_pro.app.services.AuthServices;

import jakarta.servlet.http.HttpServletResponse;

@RestController

@RequestMapping("/api/auth")

public class AuthController {

    @Autowired

    AuthServices authServices;

    @PostMapping("/signup")

    @ResponseStatus(HttpStatus.CREATED)

    public ResponseEntity<User> signup(@RequestBody UserSignupDto userSignupDto) throws UserException {

        return new ResponseEntity<User>(authServices.signupUser(userSignupDto), HttpStatus.CREATED);

    }

    @PostMapping("/signin")

    @ResponseStatus(HttpStatus.OK)

    public ResponseEntity<ResponseDto> signin(@RequestBody UserSigninDto userSigninDto, HttpServletResponse httpServletResponse) throws UserException{

        return new ResponseEntity<ResponseDto>(authServices.signinUser(userSigninDto, httpServletResponse), HttpStatus.OK);

    }

}

package com.portfolio\_pro.app.controllers;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.ResponseStatus;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import com.portfolio\_pro.app.dtos.CreateFirstPortfolioDto;

import com.portfolio\_pro.app.dtos.ResponseDto;

import com.portfolio\_pro.app.exceptions.PortfolioException;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.Portfolio;

import com.portfolio\_pro.app.services.PortfolioServices;

import jakarta.servlet.http.HttpServletRequest;

@RestController

@RequestMapping("/api/portfolios")

public class PortfolioController {

    @Autowired

    PortfolioServices portfolioServices;

    @PostMapping("/create-first")

    @ResponseStatus(HttpStatus.CREATED)

    public ResponseEntity<Portfolio> createFirstPortfolio(@RequestBody CreateFirstPortfolioDto createFirstPortfolioDto,

            HttpServletRequest httpServletRequest) throws UserException, PortfolioException {

        return new ResponseEntity<>(portfolioServices.createFirstPortfolio(createFirstPortfolioDto, httpServletRequest),

                HttpStatus.CREATED);

    }

    @GetMapping("/all")

    @ResponseStatus(HttpStatus.OK)

    public ResponseEntity<List<Portfolio>> getAllPortfoliosByUsername(HttpServletRequest httpServletRequest)

            throws UserException, PortfolioException {

        return new ResponseEntity<List<Portfolio>>(portfolioServices.getPortfoliosByUsername(httpServletRequest),

                HttpStatus.OK);

    }

    @GetMapping("/one/{portfolioId}")

    @ResponseStatus(HttpStatus.OK)

    public ResponseEntity<Portfolio> getPortfoliobyUserAndPortfolioId(HttpServletRequest httpServletRequest,

            @PathVariable Long portfolioId) throws UserException, PortfolioException {

        return new ResponseEntity<Portfolio>(

                portfolioServices.getPortfolioByUserAndPorfolioId(portfolioId, httpServletRequest), HttpStatus.OK);

    }

    @DeleteMapping("/remove/{portfolioId}")

    @ResponseStatus(HttpStatus.ACCEPTED)

    public ResponseEntity<ResponseDto> deletePortfolioByUserAndPortfolioId(HttpServletRequest httpServletRequest,

            @PathVariable Long portfolioId) throws UserException, PortfolioException {

        return new ResponseEntity<ResponseDto>(

                portfolioServices.deletePortfolioByUserAndPortfolioId(portfolioId, httpServletRequest),

                HttpStatus.ACCEPTED);

    }

}

package com.portfolio\_pro.app.controllers;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.ResponseStatus;

import org.springframework.web.bind.annotation.RestController;

import com.portfolio\_pro.app.exceptions.PortfolioException;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.Portfolio;

import com.portfolio\_pro.app.services.PublicServices;

@RestController

@RequestMapping("/api/public")

public class PublicController {

    @Autowired

    PublicServices publicServices;

    @GetMapping("/{username}/{portfolioId}")

    @ResponseStatus(HttpStatus.OK)

    public ResponseEntity<Portfolio> getPortfolioByUsernameAndPortfolioId(@PathVariable String username, @PathVariable Long portfolioId) throws UserException, PortfolioException{

        return new ResponseEntity<Portfolio>(publicServices.getPortfolioByUsernameAndPortfolioId(username, portfolioId), HttpStatus.OK);

    }

}

package com.portfolio\_pro.app.controllers;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.ResponseStatus;

import org.springframework.web.bind.annotation.RestController;

import com.portfolio\_pro.app.dtos.UpdateUserDto;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.User;

import com.portfolio\_pro.app.services.UserServices;

import jakarta.servlet.http.HttpServletRequest;

@RestController

@RequestMapping("/api/users")

public class UserController {

    @Autowired

    UserServices userServices;

    @GetMapping("/exists/{username}")

    @ResponseStatus(HttpStatus.OK)

    public ResponseEntity<Boolean> isUsernameExists(@PathVariable String username) throws UserException {

        return new ResponseEntity<>(userServices.isUsernameExists(username), HttpStatus.OK);

    }

    @PostMapping("/update")

    @ResponseStatus(HttpStatus.ACCEPTED)

    public ResponseEntity<User> updateUser(@RequestBody UpdateUserDto updateUserDto, HttpServletRequest httpServletRequest) throws UserException {

        return new ResponseEntity<>(userServices.updateUser(updateUserDto, httpServletRequest), HttpStatus.ACCEPTED);

    }

    @GetMapping("/me")

    @ResponseStatus(HttpStatus.OK)

    public ResponseEntity<User> getCurrentUser(HttpServletRequest httpServletRequest) throws UserException{

        return new ResponseEntity<User>(userServices.currentUser(httpServletRequest), HttpStatus.OK);

    }

}

DTO

package com.portfolio\_pro.app.dtos;

import java.util.List;

public class CreateFirstPortfolioDto {

//  portfolio data

    private String portfolioTitle;

    private String about;

    private String bio;

    private String professionalIdentity;

    private String profilePhotoUrl;

    private String skills;

    List<ExperienceDto> experienceDtos;

    List<ProjectDto> projectDtos;

    public CreateFirstPortfolioDto() {}

    public CreateFirstPortfolioDto(String portfolioTitle, String about, String bio, String professionalIdentity,

            String profilePhotoUrl, String skills, List<ExperienceDto> experienceDtos, List<ProjectDto> projectDtos) {

        this.portfolioTitle = portfolioTitle;

        this.about = about;

        this.bio = bio;

        this.professionalIdentity = professionalIdentity;

        this.profilePhotoUrl = profilePhotoUrl;

        this.experienceDtos = experienceDtos;

        this.projectDtos = projectDtos;

        this.skills = skills;

    }

    public String getPortfolioTitle() {

        return portfolioTitle;

    }

    public void setPortfolioTitle(String portfolioTitle) {

        this.portfolioTitle = portfolioTitle;

    }

    public String getAbout() {

        return about;

    }

    public void setAbout(String about) {

        this.about = about;

    }

    public String getBio() {

        return bio;

    }

    public void setBio(String bio) {

        this.bio = bio;

    }

    public String getProfessionalIdentity() {

        return professionalIdentity;

    }

    public void setProfessionalIdentity(String professionalIdentity) {

        this.professionalIdentity = professionalIdentity;

    }

    public String getProfilePhotoUrl() {

        return profilePhotoUrl;

    }

    public void setProfilePhotoUrl(String profilePhotoUrl) {

        this.profilePhotoUrl = profilePhotoUrl;

    }

    public List<ExperienceDto> getExperienceDtos() {

        return experienceDtos;

    }

    public void setExperienceDtos(List<ExperienceDto> experienceDtos) {

        this.experienceDtos = experienceDtos;

    }

    public List<ProjectDto> getProjectDtos() {

        return projectDtos;

    }

    public void setProjectDtos(List<ProjectDto> projectDtos) {

        this.projectDtos = projectDtos;

    }

    public String getSkills() {

        return skills;

    }

    public void setSkills(String skills) {

        this.skills = skills;

    }

}

package com.portfolio\_pro.app.dtos;

import java.time.LocalDateTime;

public class ErrorDetails {

    private String errorMessage;

    private String errorDetails;

    private LocalDateTime errorTimestamp;

    public ErrorDetails() {}

    public ErrorDetails(String errorMessage, String errorDetails, LocalDateTime errorTimestamp) {

        this.errorMessage = errorMessage;

        this.errorDetails = errorDetails;

        this.errorTimestamp = errorTimestamp;

    }

    public String getErrorMessage() {

        return errorMessage;

    }

    public void setErrorMessage(String errorMessage) {

        this.errorMessage = errorMessage;

    }

    public String getErrorDetails() {

        return errorDetails;

    }

    public void setErrorDetails(String errorDetails) {

        this.errorDetails = errorDetails;

    }

    public LocalDateTime getErrorTimestamp() {

        return errorTimestamp;

    }

    public void setErrorTimestamp(LocalDateTime errorTimestamp) {

        this.errorTimestamp = errorTimestamp;

    }

}

package com.portfolio\_pro.app.dtos;

import java.time.LocalDate;

import com.fasterxml.jackson.annotation.JsonFormat;

public class ExperienceDto {

    private String jobTitle;

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd")

    private LocalDate startingDate;

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd")

    private LocalDate endingDate;

    private String description;

    private String company;

    public ExperienceDto() {}

    public ExperienceDto(String jobTitle, LocalDate startingDate, LocalDate endingDate, String description,

            String company) {

        this.jobTitle = jobTitle;

        this.startingDate = startingDate;

        this.endingDate = endingDate;

        this.description = description;

        this.company = company;

    }

    public String getJobTitle() {

        return jobTitle;

    }

    public void setJobTitle(String jobTitle) {

        this.jobTitle = jobTitle;

    }

    public LocalDate getStartingDate() {

        return startingDate;

    }

    public void setStartingDate(LocalDate startingDate) {

        this.startingDate = startingDate;

    }

    public LocalDate getEndingDate() {

        return endingDate;

    }

    public void setEndingDate(LocalDate endingDate) {

        this.endingDate = endingDate;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

    public String getCompany() {

        return company;

    }

    public void setCompany(String company) {

        this.company = company;

    }

}

package com.portfolio\_pro.app.dtos;

public class ProjectDto {

    private String projectName;

    private String description;

    private String link;

    private String technologies; // comma separated tech stack items

    public ProjectDto() {}

    public ProjectDto(String projectName, String description, String link, String technologies) {

        this.projectName = projectName;

        this.description = description;

        this.link = link;

        this.technologies = technologies;

    }

    public String getProjectName() {

        return projectName;

    }

    public void setProjectName(String projectName) {

        this.projectName = projectName;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

    public String getLink() {

        return link;

    }

    public void setLink(String link) {

        this.link = link;

    }

    public String getTechnologies() {

        return technologies;

    }

    public void setTechnologies(String technologies) {

        this.technologies = technologies;

    }

}

package com.portfolio\_pro.app.dtos;

public class ResponseDto {

    private String message;

    private Boolean success;

    public ResponseDto(String message, Boolean success) {

        this.message = message;

        this.success = success;

    }

    public ResponseDto() {}

    public String getMessage() {

        return message;

    }

    public void setMessage(String message) {

        this.message = message;

    }

    public Boolean getSuccess() {

        return success;

    }

    public void setSuccess(Boolean success) {

        this.success = success;

    }

}

package com.portfolio\_pro.app.dtos;

import java.util.List;

import com.portfolio\_pro.app.models.Education;

import com.portfolio\_pro.app.models.Portfolio;

import com.portfolio\_pro.app.models.SocialLink;

public class UpdateUserDto {

    private String username;

    private String email;

    private String password;

    private String fullName;

    private String phoneNo;

    private String location;

    private Portfolio activePortfolio;

    private List<Education> educations;

    private List<SocialLink> socialLinks;

    public UpdateUserDto() {

    }

    public UpdateUserDto(String username, String email, String password, String fullName, String phoneNo,

            String location, Portfolio activePortfolio, List<Education> educations, List<SocialLink> socialLinks) {

        this.username = username;

        this.email = email;

        this.password = password;

        this.fullName = fullName;

        this.phoneNo = phoneNo;

        this.location = location;

        this.activePortfolio = activePortfolio;

        this.educations = educations;

        this.socialLinks = socialLinks;

    }

    public String getUsername() {

        return username;

    }

    public void setUsername(String username) {

        this.username = username;

    }

    public String getEmail() {

        return email;

    }

    public void setEmail(String email) {

        this.email = email;

    }

    public String getPassword() {

        return password;

    }

    public void setPassword(String password) {

        this.password = password;

    }

    public String getFullName() {

        return fullName;

    }

    public void setFullName(String fullName) {

        this.fullName = fullName;

    }

    public String getPhoneNo() {

        return phoneNo;

    }

    public void setPhoneNo(String phoneNo) {

        this.phoneNo = phoneNo;

    }

    public String getLocation() {

        return location;

    }

    public void setLocation(String location) {

        this.location = location;

    }

    public List<Education> getEducations() {

        return educations;

    }

    public void setEducations(List<Education> educations) {

        this.educations = educations;

    }

    public List<SocialLink> getSocialLinks() {

        return socialLinks;

    }

    public void setSocialLinks(List<SocialLink> socialLinks) {

        this.socialLinks = socialLinks;

    }

    public Portfolio getActivePortfolio() {

        return activePortfolio;

    }

    public void setActivePortfolio(Portfolio activePortfolio) {

        this.activePortfolio = activePortfolio;

    }

}

Models

package com.portfolio\_pro.app.models;

import java.util.Collection;

import java.util.Collections;

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

import org.springframework.security.core.userdetails.UserDetails;

public class CustomUserDetails implements UserDetails{

    private static final long serialVersionUID = 1L;

    private final User user;

    public CustomUserDetails(User user) {

        this.user = user;

    }

    @Override

    public Collection<? extends GrantedAuthority> getAuthorities() {

        return Collections.singleton(new SimpleGrantedAuthority("USER"));

    }

    @Override

    public String getPassword() {

        return user.getPassword();

    }

    @Override

    public String getUsername() {

        return user.getUsername();

    }

}

package com.portfolio\_pro.app.models;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.JoinColumn;

import jakarta.persistence.ManyToOne;

import jakarta.persistence.Table;

@Entity

@Table(name = "educations")

public class Education {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    @Column(name = "education\_id")

    private Long educationId;

    @ManyToOne

    @JoinColumn(name = "username", nullable = false)

    private User user;

    @Column(name = "degree\_name")

    private String degreeName;

    @Column(name = "institute\_name")

    private String instituteName;

    @Column(name = "starting\_year")

    private Integer startingYear;

    @Column(name = "ending\_year")

    private Integer endingYear;

    @Column(columnDefinition = "TEXT", name = "description")

    private String description;

    public Education() {

    }

    public Long getEducationId() {

        return educationId;

    }

    public void setEducationId(Long educationId) {

        this.educationId = educationId;

    }

    public User getUser() {

        return user;

    }

    public void setUser(User user) {

        this.user = user;

    }

    public String getDegreeName() {

        return degreeName;

    }

    public void setDegreeName(String degreeName) {

        this.degreeName = degreeName;

    }

    public String getInstituteName() {

        return instituteName;

    }

    public void setInstituteName(String instituteName) {

        this.instituteName = instituteName;

    }

    public Integer getStartingYear() {

        return startingYear;

    }

    public void setStartingYear(Integer startingYear) {

        this.startingYear = startingYear;

    }

    public Integer getEndingYear() {

        return endingYear;

    }

    public void setEndingYear(Integer endingYear) {

        this.endingYear = endingYear;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

}

package com.portfolio\_pro.app.models;

import java.time.LocalDate;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.JoinColumn;

import jakarta.persistence.ManyToOne;

import jakarta.persistence.Table;

@Entity

@Table(name = "experiences")

public class Experience {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    @Column(name = "experience\_id")

    private Long experienceId;

    @ManyToOne

    @JoinColumn(name = "portfolio\_id", nullable = false)

    @JsonIgnore

    private Portfolio portfolio;

    @Column(name = "job\_title")

    private String jobTitle;

    @Column(name = "company")

    private String company;

    @Column(name = "starting\_date")

    private LocalDate startingDate;

    @Column(name = "ending\_date")

    private LocalDate endingDate;

    @Column(columnDefinition = "TEXT", name = "description")

    private String description;

    public Experience() {}

    public Long getExperienceId() {

        return experienceId;

    }

    public void setExperienceId(Long experienceId) {

        this.experienceId = experienceId;

    }

    public Portfolio getPortfolio() {

        return portfolio;

    }

    public void setPortfolio(Portfolio portfolio) {

        this.portfolio = portfolio;

    }

    public String getJobTitle() {

        return jobTitle;

    }

    public void setJobTitle(String jobTitle) {

        this.jobTitle = jobTitle;

    }

    public String getCompany() {

        return company;

    }

    public void setCompany(String company) {

        this.company = company;

    }

    public LocalDate getStartingDate() {

        return startingDate;

    }

    public void setStartingDate(LocalDate startingDate) {

        this.startingDate = startingDate;

    }

    public LocalDate getEndingDate() {

        return endingDate;

    }

    public void setEndingDate(LocalDate endingDate) {

        this.endingDate = endingDate;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

}

package com.portfolio\_pro.app.models;

import java.util.List;

import jakarta.persistence.CascadeType;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.GenerationType;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.Id;

import jakarta.persistence.JoinColumn;

import jakarta.persistence.ManyToOne;

import jakarta.persistence.OneToMany;

import jakarta.persistence.Table;

@Entity

@Table(name = "portfolios")

public class Portfolio {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    @Column(name = "portfolio\_id")

    private long portfolioId;

    @Column(name = "portfolio\_title")

    private String portfolioTitle;

    @Column(name = "bio", columnDefinition = "TEXT")

    private String bio;

    @Column(name = "about", columnDefinition = "LONGTEXT")

    private String about;

    @Column(name = "professional\_identity")

    private String professionalIdentity;

    @Column(name = "profile\_photo\_url", unique = true)

    private String profilePhotoUrl;

    @Column(name = "skills", columnDefinition = "TEXT")

    private String skills;

    @ManyToOne

    @JoinColumn(name = "username", nullable = false)

    private User user;

    @OneToMany(mappedBy = "portfolio", cascade = CascadeType.ALL, orphanRemoval = true)

    private List<Project> projects;

    @OneToMany(mappedBy = "portfolio", cascade = CascadeType.ALL, orphanRemoval = true)

    private List<Experience> experiences;

    public Portfolio() {

    }

    public long getPortfolioId() {

        return portfolioId;

    }

    public void setPortfolioId(long portfolioId) {

        this.portfolioId = portfolioId;

    }

    public String getPortfolioTitle() {

        return portfolioTitle;

    }

    public void setPortfolioTitle(String portfolioTitle) {

        this.portfolioTitle = portfolioTitle;

    }

    public String getBio() {

        return bio;

    }

    public void setBio(String bio) {

        this.bio = bio;

    }

    public String getAbout() {

        return about;

    }

    public void setAbout(String about) {

        this.about = about;

    }

    public String getProfessionalIdentity() {

        return professionalIdentity;

    }

    public void setProfessionalIdentity(String professionalIdentity) {

        this.professionalIdentity = professionalIdentity;

    }

    public String getProfilePhotoUrl() {

        return profilePhotoUrl;

    }

    public void setProfilePhotoUrl(String profilePhotoUrl) {

        this.profilePhotoUrl = profilePhotoUrl;

    }

    public User getUser() {

        return user;

    }

    public void setUser(User user) {

        this.user = user;

    }

    public List<Project> getProjects() {

        return projects;

    }

    public void setProjects(List<Project> projects) {

        this.projects = projects;

    }

    public List<Experience> getExperiences() {

        return experiences;

    }

    public void setExperiences(List<Experience> experiences) {

        this.experiences = experiences;

    }

    public String getSkills() {

        return skills;

    }

    public void setSkills(String skills) {

        this.skills = skills;

    }

}

package com.portfolio\_pro.app.models;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.JoinColumn;

import jakarta.persistence.ManyToOne;

import jakarta.persistence.Table;

@Entity

@Table(name = "projects")

public class Project {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long projectId;

    @ManyToOne

    @JoinColumn(name = "portfolio\_id", nullable = false)

    @JsonIgnore

    private Portfolio portfolio;

    @Column(columnDefinition = "TEXT", name = "project\_name")

    private String projectName;

    @Column(columnDefinition = "LONGTEXT", name = "description")

    private String description;

    @Column(unique = true, name = "link")

    private String link;

    @Column(columnDefinition = "TEXT", name = "technologies")

    private String technologies;

    public Project() {}

    public Long getProjectId() {

        return projectId;

    }

    public void setProjectId(Long projectId) {

        this.projectId = projectId;

    }

    public Portfolio getPortfolio() {

        return portfolio;

    }

    public void setPortfolio(Portfolio portfolio) {

        this.portfolio = portfolio;

    }

    public String getProjectName() {

        return projectName;

    }

    public void setProjectName(String projectName) {

        this.projectName = projectName;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String description) {

        this.description = description;

    }

    public String getLink() {

        return link;

    }

    public void setLink(String link) {

        this.link = link;

    }

    public String getTechnologies() {

        return technologies;

    }

    public void setTechnologies(String technologies) {

        this.technologies = technologies;

    }

}

package com.portfolio\_pro.app.models;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.JoinColumn;

import jakarta.persistence.ManyToOne;

import jakarta.persistence.Table;

@Entity

@Table(name = "social\_links")

public class SocialLink {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    @Column(name = "social\_id")

    private Long socialId;

    @ManyToOne

    @JoinColumn(name = "username", nullable = false)

    private User user;

    @Column(name = "platform\_name")

    private String platformName;

    @Column(unique = true, name = "social\_url")

    private String socialUrl;

    public SocialLink() {}

    public Long getSocialId() {

        return socialId;

    }

    public void setSocialId(Long socialId) {

        this.socialId = socialId;

    }

    public User getUser() {

        return user;

    }

    public void setUser(User user) {

        this.user = user;

    }

    public String getPlatformName() {

        return platformName;

    }

    public void setPlatformName(String platformName) {

        this.platformName = platformName;

    }

    public String getSocialUrl() {

        return socialUrl;

    }

    public void setSocialUrl(String socialUrl) {

        this.socialUrl = socialUrl;

    }

}

package com.portfolio\_pro.app.models;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.JoinColumn;

import jakarta.persistence.OneToMany;

import jakarta.persistence.OneToOne;

import jakarta.persistence.Table;

import java.util.List;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.CascadeType;

@Entity

@Table(name = "users")

public class User {

    @Id

    @Column(name = "username")

    private String username;

    @Column(name = "email", unique = true, nullable = false)

    private String email;

    @Column(name = "password", nullable = false)

    private String password;

    @Column(name = "full\_name", nullable = false)

    private String fullName;

    @Column(name = "phone\_no", unique = true)

    private String phoneNo;

    @Column(name = "location")

    private String location;

    @OneToOne

    @JoinColumn(name = "active\_portfolio\_id")

    @JsonIgnore

    private Portfolio activePortfolio;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval = true)

    @JsonIgnore

    private List<Portfolio> portfolios;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval = true)

    private List<Education> educations;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval = true)

    private List<SocialLink> socialLinks;

    public User() {

    }

    public String getUsername() {

        return username;

    }

    public void setUsername(String username) {

        this.username = username;

    }

    public String getEmail() {

        return email;

    }

    public void setEmail(String email) {

        this.email = email;

    }

    public String getPassword() {

        return password;

    }

    public void setPassword(String password) {

        this.password = password;

    }

    public String getFullName() {

        return fullName;

    }

    public void setFullName(String fullName) {

        this.fullName = fullName;

    }

    public String getPhoneNo() {

        return phoneNo;

    }

    public void setPhoneNo(String phoneNo) {

        this.phoneNo = phoneNo;

    }

    public String getLocation() {

        return location;

    }

    public void setLocation(String location) {

        this.location = location;

    }

    public Portfolio getActivePortfolio() {

        return activePortfolio;

    }

    public void setActivePortfolio(Portfolio activePortfolio) {

        this.activePortfolio = activePortfolio;

    }

    public List<Portfolio> getPortfolios() {

        return portfolios;

    }

    public void setPortfolios(List<Portfolio> portfolios) {

        this.portfolios = portfolios;

    }

    public List<Education> getEducations() {

        return educations;

    }

    public void setEducations(List<Education> educations) {

        this.educations = educations;

    }

    public List<SocialLink> getSocialLinks() {

        return socialLinks;

    }

    public void setSocialLinks(List<SocialLink> socialLinks) {

        this.socialLinks = socialLinks;

    }

}

Repositories

package com.portfolio\_pro.app.repositories;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.portfolio\_pro.app.models.Education;

@Repository

public interface EducationRepository extends JpaRepository<Education, Long>{

}

package com.portfolio\_pro.app.repositories;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.portfolio\_pro.app.models.Experience;

@Repository

public interface ExperienceRepositry extends JpaRepository<Experience, Long>{

    @SuppressWarnings("unchecked")

    public Experience save(Experience experience);

}

package com.portfolio\_pro.app.repositories;

import java.util.List;

import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.portfolio\_pro.app.models.Portfolio;

import com.portfolio\_pro.app.models.User;

@Repository

public interface PortfolioRepository extends JpaRepository<Portfolio, Long>{

    @SuppressWarnings("unchecked")

    public Portfolio save(Portfolio portfolio);

    public List<Portfolio> findByUser(User user);

    public Optional<Portfolio> findByUserAndPortfolioId(User user, Long portfolioId);

}

package com.portfolio\_pro.app.repositories;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.portfolio\_pro.app.models.Project;

@Repository

public interface ProjectRepository extends JpaRepository<Project, Long>{

    @SuppressWarnings("unchecked")

    public Project save(Project project);

}

package com.portfolio\_pro.app.repositories;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.portfolio\_pro.app.models.SocialLink;

@Repository

public interface SocialLinkRepository extends JpaRepository<SocialLink, Long>{

}

package com.portfolio\_pro.app.repositories;

import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.portfolio\_pro.app.models.User;

@Repository

public interface UserRepository extends JpaRepository<User, String>{

    public boolean existsByUsername(String username);

    public boolean existsByEmail(String email);

    public Optional<User> findByUsername(String username);

    @SuppressWarnings("unchecked")

    public User save(User user);

}

Serviceimplementation

package com.portfolio\_pro.app.serviceimps;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.stereotype.Service;

import com.portfolio\_pro.app.dtos.ResponseDto;

import com.portfolio\_pro.app.dtos.UserSigninDto;

import com.portfolio\_pro.app.dtos.UserSignupDto;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.User;

import com.portfolio\_pro.app.repositories.UserRepository;

import com.portfolio\_pro.app.services.AuthServices;

import com.portfolio\_pro.app.services.JwtService;

import com.portfolio\_pro.app.utils.AuthUtil;

import com.portfolio\_pro.app.utils.ModelMapper;

import jakarta.servlet.http.Cookie;

import jakarta.servlet.http.HttpServletResponse;

@Service

public class AuthServicesImp implements AuthServices {

    @Autowired

    BCryptPasswordEncoder bCryptPasswordEncoder;

    @Autowired

    UserRepository userRepository;

    @Autowired

    AuthenticationManager authenticationManager;

    @Autowired

    JwtService jwtService;

    @Autowired

    AuthUtil authUtil;

    @Override

    public User signupUser(UserSignupDto userSignupDto) throws UserException{

        if(userRepository.existsByEmail(userSignupDto.getEmail())) {

            throw new UserException("Email is already in use");

        }

        if(userRepository.existsByUsername(userSignupDto.getUsername())) {

            throw new UserException("Username is already taken");

        }

        User user = ModelMapper.toUserFromUserSignupDto(userSignupDto);

        user.setPassword(bCryptPasswordEncoder.encode(userSignupDto.getPassword()));

        return userRepository.save(user);

    }

    @Override

    public ResponseDto signinUser(UserSigninDto userSigninDto, HttpServletResponse httpServletResponse) throws UserException{

        Authentication authentication = authenticationManager

                .authenticate(new UsernamePasswordAuthenticationToken(userSigninDto.getUsername(), userSigninDto.getPassword()));

        if(authentication.isAuthenticated()) {

            Cookie cookie = authUtil.generateCookie(userSigninDto);

            httpServletResponse.addCookie(cookie);

            return new ResponseDto(authUtil.generateCookie(userSigninDto).getValue(), true);

        }

        return new ResponseDto(null, false);

    }

}

package com.portfolio\_pro.app.serviceimps;

import java.util.List;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.portfolio\_pro.app.dtos.CreateFirstPortfolioDto;

import com.portfolio\_pro.app.dtos.ExperienceDto;

import com.portfolio\_pro.app.dtos.ProjectDto;

import com.portfolio\_pro.app.dtos.ResponseDto;

import com.portfolio\_pro.app.exceptions.PortfolioException;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.Experience;

import com.portfolio\_pro.app.models.Portfolio;

import com.portfolio\_pro.app.models.Project;

import com.portfolio\_pro.app.models.User;

import com.portfolio\_pro.app.repositories.ExperienceRepositry;

import com.portfolio\_pro.app.repositories.PortfolioRepository;

import com.portfolio\_pro.app.repositories.ProjectRepository;

import com.portfolio\_pro.app.repositories.UserRepository;

import com.portfolio\_pro.app.services.JwtService;

import com.portfolio\_pro.app.services.PortfolioServices;

import com.portfolio\_pro.app.utils.AuthUtil;

import com.portfolio\_pro.app.utils.ModelMapper;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.transaction.Transactional;

@Service

public class PortfolioServicesImp implements PortfolioServices {

    @Autowired

    ProjectRepository projectRepository;

    @Autowired

    UserRepository userRepository;

    @Autowired

    PortfolioRepository portfolioRepository;

    @Autowired

    ExperienceRepositry experienceRepositry;

    @Autowired

    JwtService jwtService;

    @Autowired

    AuthUtil authUtil;

    @Override

    @Transactional

    public Portfolio createFirstPortfolio(CreateFirstPortfolioDto createFirstPortfolioDto,

            HttpServletRequest httpServletRequest) throws PortfolioException, UserException {

        String username = null;

        try {

            username = authUtil.extractUsernameFromRequest(httpServletRequest);

        } catch (ServletException e) {

            e.printStackTrace();

        }

        Optional<User> user = userRepository.findByUsername(username);

        if (user.isEmpty()) {

            throw new UserException("User not available by username " + username);

        }

        Portfolio portfolio = ModelMapper.toPortfolioFromCreateFirstPortfolioDto(createFirstPortfolioDto);

        portfolio.setUser(user.get());

        Portfolio newPortfolio = portfolioRepository.save(portfolio);

        User currUser = user.get();

        currUser.setActivePortfolio(newPortfolio);

        User updatedUser = userRepository.save(currUser);

        if (updatedUser == null) {

            throw new UserException("Failed to update active portfolio.");

        }

        List<ExperienceDto> experienceDtos = createFirstPortfolioDto.getExperienceDtos();

        List<ProjectDto> projectDtos = createFirstPortfolioDto.getProjectDtos();

        for (ExperienceDto experienceDto : experienceDtos) {

            Experience experience = ModelMapper.toExperienceFromExperienceDto(experienceDto);

            experience.setPortfolio(newPortfolio);

            experienceRepositry.save(experience);

        }

        for (ProjectDto projectDto : projectDtos) {

            Project project = ModelMapper.toProjectFromProjectDto(projectDto);

            project.setPortfolio(newPortfolio);

            projectRepository.save(project);

        }

        newPortfolio = portfolioRepository.save(newPortfolio);

        if (newPortfolio == null) {

            throw new PortfolioException("Failed to Create Portfolio");

        }

        return newPortfolio;

    }

    @Override

    public List<Portfolio> getPortfoliosByUsername(HttpServletRequest httpServletRequest)

            throws UserException, PortfolioException {

        String username = null;

        try {

            username = authUtil.extractUsernameFromRequest(httpServletRequest);

        } catch (ServletException e) {

            e.printStackTrace();

        }

        Optional<User> user = userRepository.findByUsername(username);

        if (user.isEmpty()) {

            throw new UserException("User Does Not Exist");

        }

        List<Portfolio> portfolios = portfolioRepository.findByUser(user.get());

        return portfolios;

    }

    @Override

    public Portfolio getPortfolioByUserAndPorfolioId(Long portfolioId, HttpServletRequest httpServletRequest)

            throws UserException, PortfolioException {

        String username = null;

        try {

            username = authUtil.extractUsernameFromRequest(httpServletRequest);

        } catch (ServletException e) {

            e.printStackTrace();

        }

        Optional<User> user = userRepository.findByUsername(username);

        if (user.isEmpty()) {

            throw new UserException("User Does Not Exist");

        }

        Optional<Portfolio> portfolio = portfolioRepository.findByUserAndPortfolioId(user.get(), portfolioId);

        if (portfolio.isEmpty()) {

            throw new PortfolioException("Portfolio Does Not Exist");

        }

        return portfolio.get();

    }

    @Override

    @Transactional

    public ResponseDto deletePortfolioByUserAndPortfolioId(Long portfolioId, HttpServletRequest httpServletRequest)

            throws UserException, PortfolioException {

        String username = null;

        try {

            username = authUtil.extractUsernameFromRequest(httpServletRequest);

        } catch (ServletException e) {

            e.printStackTrace();

        }

        Optional<User> user = userRepository.findByUsername(username);

        if (user.isEmpty()) {

            throw new UserException("User Does Not Exist");

        }

        Optional<Portfolio> portfolio = portfolioRepository.findById(portfolioId);

        if (portfolio.isEmpty()) {

            throw new PortfolioException("Portfolio Does Not Exists");

        }

        if (!portfolio.get().getUser().getUsername().equals(username)) {

            throw new UserException("User Not Authorized");

        }

        if (user.get().getActivePortfolio() != null

                && user.get().getActivePortfolio().getPortfolioId() == portfolio.get().getPortfolioId()) {

            user.get().setActivePortfolio(null);

            userRepository.save(user.get()); // update user to remove reference

        }

        portfolioRepository.delete(portfolio.get());

        return new ResponseDto("Portfolio Successfully Deleted", true);

    }

}

package com.portfolio\_pro.app.serviceimps;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.portfolio\_pro.app.exceptions.PortfolioException;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.Portfolio;

import com.portfolio\_pro.app.models.User;

import com.portfolio\_pro.app.repositories.PortfolioRepository;

import com.portfolio\_pro.app.repositories.UserRepository;

import com.portfolio\_pro.app.services.PublicServices;

@Service

public class PublicServiceImp implements PublicServices{

    @Autowired

    UserRepository userRepository;

    @Autowired

    PortfolioRepository portfolioRepository;

    @Override

    public Portfolio getPortfolioByUsernameAndPortfolioId(String username, Long portfolioId)

            throws UserException, PortfolioException {

        Optional<User> user = userRepository.findByUsername(username);

        if(user.isEmpty()) {

            throw new UserException("User Does Not Exists");

        }

        Optional<Portfolio> portfolio = portfolioRepository.findByUserAndPortfolioId(user.get(), portfolioId);

        if(portfolio.isEmpty()) {

            throw new UserException("Portfolio Does Not Exists");

        }

        return portfolio.get();

    }

}

package com.portfolio\_pro.app.serviceimps;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.portfolio\_pro.app.dtos.UpdateUserDto;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.User;

import com.portfolio\_pro.app.repositories.UserRepository;

import com.portfolio\_pro.app.services.UserServices;

import com.portfolio\_pro.app.utils.AuthUtil;

import com.portfolio\_pro.app.utils.ModelMapper;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

@Service

public class UserServicesImp implements UserServices{

    @Autowired

    UserRepository userRepository;

    @Autowired

    AuthUtil authUtil;

    @Override

    public boolean isUsernameExists(String username) {

        return userRepository.existsByUsername(username);

    }

    @Override

    public User updateUser(UpdateUserDto updateUserDto, HttpServletRequest httpServletRequest) throws UserException {

        String username = null;

        try {

            username = authUtil.extractUsernameFromRequest(httpServletRequest);

        } catch (ServletException e) {

            e.printStackTrace();

        }

        Optional<User> user = userRepository.findByUsername(username);

        if(user.isEmpty()) {

            throw new UserException("User Does Not Exists");

        }

        User updatedUserData = ModelMapper.mapUserDtoToUser(user.get(), updateUserDto);

        User updatedUser = userRepository.save(updatedUserData);

        if(updatedUser == null) {

            throw new UserException("User Update Failed");

        }

        return updatedUser;

    }

    @Override

    public User currentUser(HttpServletRequest httpServletRequest) throws UserException {

        String username = null;

        try {

            username = authUtil.extractUsernameFromRequest(httpServletRequest);

        } catch (ServletException e) {

            e.printStackTrace();

        }

        Optional<User> user = userRepository.findByUsername(username);

        if(user.isEmpty()) {

            throw new UserException("User Does Not Exists");

        }

        return user.get();

    }

}

Service

package com.portfolio\_pro.app.services;

import com.portfolio\_pro.app.dtos.ResponseDto;

import com.portfolio\_pro.app.dtos.UserSigninDto;

import com.portfolio\_pro.app.dtos.UserSignupDto;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.User;

import jakarta.servlet.http.HttpServletResponse;

public interface AuthServices {

    public User signupUser(UserSignupDto userSignupDto) throws UserException;

    public ResponseDto signinUser(UserSigninDto userSigninDto, HttpServletResponse httpServletResponse) throws UserException;

}

package com.portfolio\_pro.app.services;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.stereotype.Service;

import com.portfolio\_pro.app.models.CustomUserDetails;

import com.portfolio\_pro.app.models.User;

import com.portfolio\_pro.app.repositories.UserRepository;

@Service

public class CustomUserDetailsService implements UserDetailsService{

    @Autowired

    UserRepository userRepository;

    public CustomUserDetailsService(UserRepository userRepository) {

        this.userRepository = userRepository;

    }

    @Override

    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

        Optional<User> user = userRepository.findByUsername(username);

        if(user.isEmpty()) {

            System.out.println("User not available");

            throw new UsernameNotFoundException("User not found!!!");

        }

        return new CustomUserDetails(user.get());

    }

}

package com.portfolio\_pro.app.services;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import java.util.function.Function;

import javax.crypto.SecretKey;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.stereotype.Service;

import com.portfolio\_pro.app.dtos.UserSigninDto;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.io.Decoders;

import io.jsonwebtoken.security.Keys;

@Service

public class JwtService {

    @Value("${jwt.secret}")

    private String secretKey;

    public String generateToken(UserSigninDto userSigninDto) {

        Map<String, Object> claims = new HashMap<>();

        claims.put("role", "USER");

        return Jwts

                .builder()

                .claims()

                .add(claims)

                .subject(userSigninDto.getUsername())

                .issuer("XERK").issuedAt(new Date(System.currentTimeMillis()))

                .expiration(new Date(System.currentTimeMillis() + 24\*60\*60\*1000))

                .and()

                .signWith(generateKey())

                .compact();

    }

    private SecretKey generateKey() {

        byte[] decode = Decoders.BASE64.decode(getSecretKey());

        return Keys.hmacShaKeyFor(decode);

    }

    public String getSecretKey() {

        return secretKey;

    }

    public boolean isTokenValid(String jwt, UserDetails userDetails) {

        final String username = extractUsername(jwt);

        return (userDetails.getUsername().equals(username) && !isTokenExpired(jwt));

    }

    private boolean isTokenExpired(String jwt) {

        return extractExpiration(jwt).before(new Date());

    }

    private Date extractExpiration(String jwt) {

        return extractClaims(jwt, Claims::getExpiration);

    }

    public String extractUsername(String jwt) {

        return extractClaims(jwt, Claims::getSubject);

    }

    private <T> T extractClaims(String jwt, Function<Claims, T> claimResolver) {

        Claims claims = extractClaims(jwt);

        return claimResolver.apply(claims);

    }

    private Claims extractClaims(String jwt) {

        return Jwts.parser().verifyWith(generateKey()).build().parseSignedClaims(jwt).getPayload();

    }

}

package com.portfolio\_pro.app.services;

import java.util.List;

import com.portfolio\_pro.app.dtos.CreateFirstPortfolioDto;

import com.portfolio\_pro.app.dtos.ResponseDto;

import com.portfolio\_pro.app.exceptions.PortfolioException;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.Portfolio;

import jakarta.servlet.http.HttpServletRequest;

public interface PortfolioServices {

    public Portfolio createFirstPortfolio(CreateFirstPortfolioDto createPortfolioDto, HttpServletRequest httpServletRequest) throws PortfolioException, UserException;

    public List<Portfolio> getPortfoliosByUsername(HttpServletRequest httpServletRequest) throws UserException, PortfolioException;

    public Portfolio getPortfolioByUserAndPorfolioId(Long portfolioId, HttpServletRequest httpServletRequest) throws UserException, PortfolioException;

    public ResponseDto deletePortfolioByUserAndPortfolioId(Long portfolioId, HttpServletRequest httpServletRequest) throws UserException, PortfolioException;

}

package com.portfolio\_pro.app.services;

import com.portfolio\_pro.app.exceptions.PortfolioException;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.Portfolio;

public interface PublicServices {

    public Portfolio getPortfolioByUsernameAndPortfolioId(String username, Long portfolioId) throws UserException, PortfolioException;

}

package com.portfolio\_pro.app.services;

import com.portfolio\_pro.app.dtos.UpdateUserDto;

import com.portfolio\_pro.app.exceptions.UserException;

import com.portfolio\_pro.app.models.User;

import jakarta.servlet.http.HttpServletRequest;

public interface UserServices {

     public boolean isUsernameExists(String username);

//   public boolean changeUsername(String username) throws UserException;

     public User updateUser(UpdateUserDto updateUserDto, HttpServletRequest httpServletRequest) throws UserException;

     public User currentUser(HttpServletRequest httpServletRequest) throws UserException;

}

package com.portfolio\_pro.app;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class PortfolioProApplication {

    public static void main(String[] args) {

        SpringApplication.run(PortfolioProApplication.class, args);

    }

}